Федеральное государственное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований Российской академии наук»
(ФГУ ФНЦ НИИСИ РАН)

ТРУДЫ НИИСИ РАН

ТОМ 8 № 6

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2018

Тематика номера:
Исследование физических процессов и их моделирование, суперкомпьютерные информационные технологии, информационные технологии и электронные системы, информационные технологии в образовании и культуре.

Журнал публикует оригинальные статьи по следующим областям исследований: математическое и компьютерное моделирование, обработка изображений, визуализация, системный анализ, методы обработки сигналов, информационная безопасность, информационные технологии, высокопроизводительные вычисления, оптико-нейронные технологии, микро- и наноэлектроника, математические исследования и вопросы численного анализа, история науки и техники.

The topic of the issue:
Research of physical processes and their modeling, Supercomputer information technologies, Information technologies and electronic systems, Information technologies in education and culture.

The Journal publishes novel articles on the following research arias: mathematical and computer modeling, image processing, visualization, system analysis, signal processing, information security, information technologies, high-performance computing, optical-neural technologies, micro- and nanoelectronics, mathematical researches and problems of numerical analysis, history of science and of technique.

Заведующий редакцией: Ю.Н.Штейников

Издатель: ФГУ ФНЦ НИИСИ РАН,
117218, Москва, Нахимовский проспект 36, к. 1

СОДЕРЖАНИЕ

I. ИССЛЕДОВАНИЕ ФИЗИЧЕСКИХ ПРОЦЕССОВ И ИХ МОДЕЛИРОВАНИЕ

II. СУПЕРКОМПЬЮТЕРНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

III. ИНФОРМАЦИОННЕ ТЕХНОЛОГИИ И ЭЛЕКТРОННЫЕ СИСТЕМЫ

IV. ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ОБРАЗОВАНИИ И КУЛЬТУРЕ

Процессы диффузии элементов группы 3 и 5 в частице катализатора при росте нитевидных нанокристаллов A^3B^5 по механизму пар-кристалл-кристалл

А.А. Корякин[1], С.А. Кукушкин[2]

[1,2] Санкт-Петербургский национальный исследовательский Академический университет РАН, Санкт-Петербург, Россия, [1] СПбО МСЦ РАН - филиал ФГУ ФНЦ НИИСИ РАН, Санкт-Петербург, Россия, [2] Институт проблем машиноведения Российской академии наук, Санкт-Петербург, Россия

E-mail's: [1]alexkorya@gmail.com, [2]sergey.a.kukushkin@gmail.com

Аннотаци: Изучены процессы диффузии элементов группы 3 и 5 в частице катализатора при росте нитевидных нанокристаллов (ННК) A^3B^5 по механизму пар-кристалл-кристалл. Рассмотрен случай формирования золото-каталитических ННК InAs на подложке InAs(111). Вычислены профили концентрации индия и мышьяка в объеме частицы катализатора. Обнаружено, что на скорость роста островков A^3B^5 влияют диффузионные потоки индия как в объеме, так и на границе раздела катализатор-ННК. Однако, диффузия мышьяка происходит преимущественно в тонком диффузионном слое вблизи границы раздела катализатор-ННК. Фактором, лимитирующим рост островков A^3B^5, является транспорт частиц группы 5 на границе диффузионного слоя и газовой фазы.

Ключвые слова: нитевидные нанокристаллы, нуклеация, полупроводники

1. Введение

Нитевидные нанокристаллы (ННК) полупроводниковых соединений A^3B^5 являются перспективными наноматериалами для создания высокопроизводительных электронных и оптоэлектронных приборов [1, 2]. В настоящее время наибольшее распространение получили каталитические методы синтеза ННК [2]. Для реализации каталитического метода роста ННК на подготовительной стадии на подложку осаждают вещество катализатора (например, золото на подложку A^3B^5(111)). Затем производят отжиг подложки для формирования эвтектического сплава вещетва катализатора и элеметов группы 3. После начала осаждения потоков группы 3 и 5 происходит растворение элементов группы 3 и 5 в частицах катализатора и при достижении пересыщения относительно подложки пары A^3B^5 кристаллизуются на границе катализатор-подложка. В результате нуклеации двумерных зародышей (островков) происходит послойный рост ННК [3-5]. Скорость роста кристалла A^3B^5 на части подложки, активированной частицами катализатора, обычно на порядки больше скорости роста на неактивировнных участках [2]. Большинство моделей каталитического роста ННК (например, [3-5]) строится в предположении, что частица катализатора, расположенная на вершине ННК, во время роста находится в жидком состоянии (механизм роста пар-жидкость-кристалл). Однако, однозначно определить фазовое состояние катализатора часто затруднительно, так как рост ННК наблюдается при температурах как выше температуры плавления частицы катализатора, так и ниже. Например, золото-каталитические ННК InAs на подложке InAs(111) формируют в диапазоне температур 380-480 ^oC [6], причем температура наименьшей эвтектики сплава золота с индием равна 454.3 ^oC [7]. Влиянием содержания частиц группы 5 в катализаторе (мышьяка и фосфора, ~ 1-5 %) на фазовое состояние обычно пренебрегают [8]. Более того, методом просвечивающей электронной микроскопии *in situ* было показано, что одновременно на подложке может происходить рост ННК A^3B^5 с жидкими и твердыми частицами катализатора [8]. В последнем случае, рост происходит по механизму пар-кристалл-кристалл (ПКК).

В настоящей работе изучается процесс роста золото-каталитических ННК A^3B^5 по механизму ПКК. В качестве примера рассматривается рост ННК InAs на подложке InAs(111). Определяются возможные пути диффузии частиц группы 3 и 5 в катализаторе, вычисляются потоки, лимитирующие рост A^3B^5 островков, также оценивается величина характерного времени нуклеации при зарождении на границе катализатор-ННК. Оценки времени нуклеации проводятся в рамках модели роста ННК по механизум ПКК, построенной в работах [9, 10]. Отличием настоящего исследования является учет влияния потоков как элементов группы 3, так и элементов группы 5 на интенсивность зарождения A^3B^5 островков.

2. Теоретическая модель

Согласно теории нуклеации в твердом теле [11] энергия Гиббса образования островка A^3B^5 на границе раздела катализатор-ННК при механизме роста ПКК может быть представлена в виде [9]

$$\Delta F = \left(\alpha + b(\ln i + d)\right) i^{1/2} - \left(\Delta\mu - w_0\right) i, \quad (1)$$

где α – вклад, зависящий от межфазной поверхностной энергии боковых поверхностей островка γ и его формы; для островков, имеющих форму плоского цилиндра радиуса R и высотой h, равной одному монослою, $\alpha = 2\pi^{1/2}\gamma(h\Omega)^{1/2}$, Ω – объем пары A^3B^5 в твердом теле; $\Delta\mu$ – разность химических потенциалов для частиц в катализаторе и островке; i – число пар A^3B^5 в островке; сумма $w_0 + b(\ln i + d)/i^{1/2}$ равна энергии упругих напряжений, возникающих из-за разности атомных плотностей в катализаторе и островке. Разность химических потенциалов $\Delta\mu$ в формуле (1) является функцией концентраций элементов группы 3 и 5 в катализаторе, C_3 и C_5, и, следовательно, энергия Гиббса также есть функция концентраций. Используя выражение (1) для энергии образования островка, находим, что барьер нуклеации равен [9]

$$\Delta F^* = \frac{b^2\left(2W_j(z) + W_j^{\,2}(z)\right)}{\Delta\mu - w_0}, \quad (2)$$

где

$$z = -\frac{\Delta\mu - w_0}{b}\exp\left[-\frac{1}{2}\left(\frac{\alpha}{b} + d\right) - 1\right], \quad (3)$$

$W_j(z)$ – функция Ламберта, $j = -1$ при $b > 0$ и $j = 0$ при $b < 0$. Интенсивность нуклеации островков определяется по формуле [12]: $I = N_0 W^+ Z \exp\left[-\Delta F^*/k_B T\right]$, где $N_0 \sim 1/a^2$ – максимальная поверхностная плотность пар A^3B^5, a – постоянная решетки A^3B^5; W^+ – коэффициент диффузии в пространстве размеров; Z – фактор Зельдовича; k_B – постоянная Больцмана. Тогда характерное время между двумя актами нуклеации можно оценить по формуле [3]:

$$\tau_N \sim 1/\pi(D/2)^2 I. \quad (4)$$

Изучим зависимость величины барьера нуклеации ΔF^* и времени нуклеации τ_N от времени роста при стационарном режиме роста ННК. С этой целью определим вначале кинетику заполнения частицы катализатора потоками элементов группы 3 и 5 после формирования монослоя. Будем считать, что частица катализатора является монокристаллом, так как ее размеры малы $D \sim$ 20-80 нм [6]. Действительно, характерные размеры зерен в сплавах элемента группы 3 с золотом после отжига подложки равны $\sim$ 30-60 нм [13] и сравнимы с размерами частиц катализатора.

Рассмотрим два возможных пути диффузии элементов группы 3 и 5 к растущей поверхности ННК: в объеме катализатора и вдоль границы раздела катализатор-ННК. Граница раздела в данном случае является по определению границей раздела зерен: кристалла A^3-Au и кристалла A^3B^5 [14]. Толщина границы раздела δ обычно принимается равной 2-3 межатомным расстояниям [14]. Диффузия вдоль границы раздела зерен происходит быстрее, чем в объеме. Причем характерные коэффициенты диффузии в объеме D^v на несколько порядков меньше, чем коэффициенты диффузии вдоль границы раздела зерен D^b. Однако, глубина проникновения вещества при диффузия вдоль границы зерна может быть сравнимой с глубиной проникновения в объеме [14]. Это происходит из-за того, что когда диффундирующее вещество проходит вглубь границы, оно начинает диффундировать в направлениях, перпендикулярных границе раздела. Таким образом, граница раздела зерен питает слои кристаллов по обе ее стороны. Поэтому в общем случае необходимо

рассматривать как объемные потоки, так и потоки вдоль границы раздела катализатор-ННК.

Для моделирования диффузионных потоков вещества будем считать, что частица катализатора имеет форму полусферы (рисунок 1). Коэффициенты диффузии

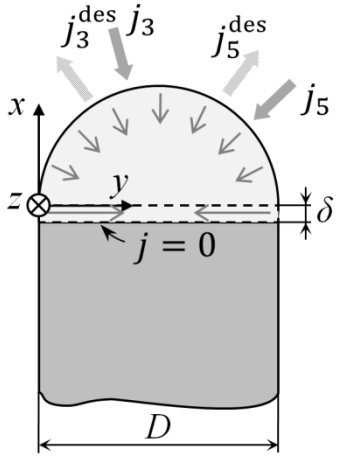


Рис. 1. Модельная геометрия системы. Потоки вещества в газовой фазе и в катализаторе показаны серыми стрелками.

элементов группы 3 и 5 в объеме равны, соответственно, D_3^v и D_5^v.

На границе катализатор-ННК находится диффузионный слой толщиной δ, в котором коэффициенты диффузии равны, соответственно, D_3^b и D_5^b. Начало отсчета системы координат и направление осей показаны на рисунке 1. Граничные условия для уравнений диффузии следующие.

На поверхности частицы катализатора заданы потоки j_3 и j_5, определяющие количество частиц группы 3 и 5 поступающих в объем частицы катализатора.

В общем случае, в потоки j_3 и j_5 дают вклад прямые потоки из газовой фазы, отраженные потоки от соседних ННК и подложки и диффузионные потоки с боковых стенок ННК.

Также в модели учитываются десорбционные потоки, $j_3^{des} = k_3 C_3$ и $j_5^{des} = k_5 C_5^2$, где k_3 и k_5 – коэффициенты десорбции и учтено, что частицы группы 3 поступают на поверхность катализатора в виде атомов, а частицы группы 5 в виде молекул $B^5{}_2$ [5, 6]. Коэффициенты десорбции могут быть оценены, используя формулы для давления насыщенных паров элементов группы 3 и 5 [9]:

$$k_3 = \frac{0.1\,\text{МПа}}{(2\pi m_3 k_B T)^{1/2}} \exp\left[\frac{\mu_3^s - \mu_3^g}{k_B T}\right], \qquad (5)$$

$$k_5 = \frac{2\times 0.1\,\text{МПа}}{(2\pi m_5 k_B T)^{1/2}} \exp\left[\frac{2\mu_5^s - \mu_5^g}{k_B T}\right], \qquad (6)$$

где μ^s и μ^g – химические потенциалы частиц в твердом и газообразном состоянии, являются известными функциями температуры [15]; m – масса частиц. Диффузией элементов группы 3 и 5 в объем кристалла A^3B^5 пренебрегается, поэтому величина потоков через поверхность $x = -\delta$ полагается равной нулю, $j_3 = j_5 = 0$. В начальный момент времени концентрации элементов 3 и 5 равны нулю, $C_3(t=0) = C_5(t=0) = 0$. Данное условие соответствует формированию первого монослоя. Однако, оно может быть также использовано для оценки порядка величины времени нуклеации при формировании последующих монослоев.

Далее рассмотрение проведем на примере формирования золото-каталитических ННК InAs на подложке InAs(111) [6]. После отжига на подложке формируются золото-индиевые частицы катализатора. Согласно работе [16], посвященной изучению диффузии индия в золотой пленке, индий вступает в химическую реакцию с золотом даже при низких температурах ($\sim$ 60-150 °C). Причем, формирование стабильных соединений (например, $AuIn_2$ [16]) сопровождается диффузией индия в объеме золото-индиевого кристалла. Важно отметить, что как только в некоторой области формируется стабильное соединение, концентрация индия перестает существенно меняться. Однако, через данную область и в дальнейшем могут протекать диффузионные потоки индия. Поэтому, мы предполагаем, что концентрация индия в диффузионных потоках в катализаторе много меньше его суммарной концентрации. Разность химических потенциалов оценивается по формуле $\Delta\mu = k_B T \ln\left(C_{In} C_{As} / K^{eq}\right)$ [12], где K^{eq} – константа равновесия химической реакции In + As $\rightarrow$ InAs. Параметры, определяющие упругую энергию в случае зарождения островка InAs в матрице AuIn были определены в работе [10]: $w_0 = 28.8$ (мэВ), $b = 112$ (мэВ), $d = 1.55$. Межфазная энергия γ при нуклеации когерентных с матрицей зародышей равна по порядку ~ 0.01 Дж м$^{-2}$ [11],

поэтому величиной α можно пренебречь по сравнению с вкладом $b(\ln i + d)$ в энергию Гиббса (формула 1).

В результате решения уравнений диффузии находим профиль состава в различные моменты времени. На рисунке 2 показана зависимость концентрации мышьяка в объеме катализаторе от расстояния вдоль оси x ($y = D/2$) и зависимость концентрации мышьяка на границе раздела катализатор-ННК от расстояния вдоль оси y ($x = 0$).

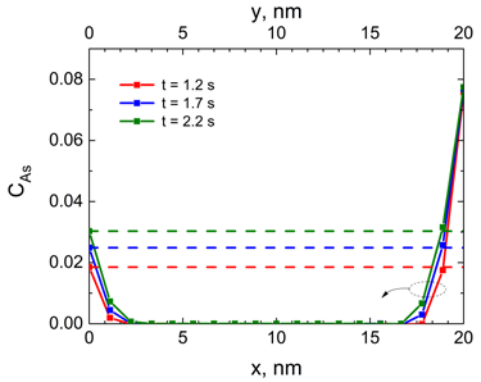


Рис. 2. Профиль концентрации мышьяка в частице катализатора.

Видно, что диффузионными потоками мышьяка в объеме катализатора можно пренебречь, так как диффузионный фронт перемещается со скоростью ~ 1 нм с^{-1}. Однако, диффузия мышьяка вдоль границы раздела происходит значительно быстрее, с характерным временем диффузии $\sim 10^{-4}$ с. Поэтому градиент концентрации вдоль оси y пренебрежимо мал. Диффузия индия происходит одинаково быстро вдоль границы раздела и в объеме катализатора. Градиент концентрации индия во всем объеме катализатора близок к нулю в любой момент времени. На рисунке 3 показана зависимость концентрации мышьяка и индия вблизи границы раздела катализатор-ННК ($y = D/4$) от времени. При больших временах в отсутствии зарождающихся островков концентрация мышьяка достигает максимально значения $C_{As}^{\max}$, зависящего от константы десорбции k_{As} (формула 6) и потока j_{As} (в данном расчете $C_{As}^{\max} = 0.081$). Концентрация индия может достигать десятков процентов, так как константа десорбции k_{In} много меньше j_{In}. Расчет произведен для температуры T = 400 °C

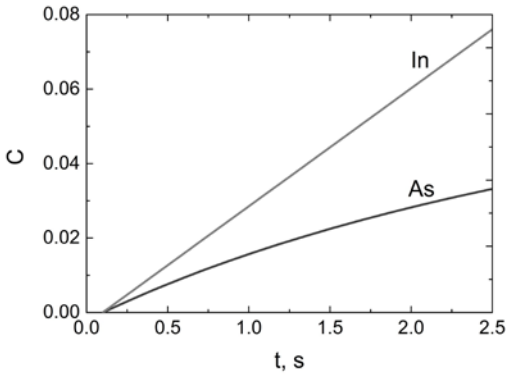


Рис. 3. Зависимость концентрации индия и мышьяка вблизи границы раздела катализатор-ННК от времени.

при следующих значениях параметров модели: $D_{In}^{v} = 1.5\times10^{-12}$ $\text{м}^2\text{с}^{-1}$ [16], $D_{In}^{b} = 1.5\times10^{-11}$ $\text{м}^2\text{с}^{-1}$ [16], $D_{As}^{v} = 4.2\times10^{-19}$ $\text{м}^2\text{с}^{-1}$ [17], $D_{As}^{b} = 1.9\times10^{-12}$ $\text{м}^2\text{с}^{-1}$ [17], D = 40 нм, δ = 0.82 нм, $j_{In} = 1\times10^{19}$ $\text{с}^{-1}\text{м}^{-2}$, $j_{As} = 1\times10^{19}$ $\text{с}^{-1}\text{м}^{-2}$ [5], $k_{In} = 2.6\times10^{7}$ $\text{с}^{-1}\text{м}^{-2}$, $k_{As} = 1.5\times10^{21}$ $\text{с}^{-1}\text{м}^{-2}$. Как только происходит акт нуклеации, на границе раздела катализатор-ННК резко понижается пересыщение. Учитывая быстрый транспорт мышьяка в диффузионном слое толщиной δ, мы полагаем, что все количество мышьяка, содержавшееся в нем после акта нуклеации за время $\sim 10^{-4}$ с потребляется растущим островком. Данного количества достаточно для формирования островка радиусом ~ 5 нм. Количества же индия, содержащегося в объеме катализатора будет с избытком хватать для формирования монослоя. Поэтому рост островка ограничен потоком мышьяка вдоль границы раздела, который, в свою очередь, определяется граничным потоком j_{As}.

Для оценок времени нуклеации по формуле 4 необходимо получить выражения, определяющие W^+ и Z. Предполагая, что рост островка ограничен потоками группы 5 на границе раздела катализатор-ННК, коэффициент диффузии в пространстве размеров может быть найден в виде [9]

$$W^+ = \frac{2\pi N_0 D_5^b C_5}{\ln \lambda_5 / R_c}, \qquad (7)$$

где λ_5 – характерный размер области, приходящийся на один островок, в расчетах

полагается равным $\sim D/2$, R_c – радиус островка критического размера, равный

$$R_c = \left(\frac{\Omega}{\pi h}\right)^{1/2} \frac{|b| W_j(z)}{\Delta\mu - w_0}. \quad (8)$$

Фактор Зельдовича с учетом выражения (1) находится в виде [9]

$$Z = \frac{1}{2|b|}\sqrt{\frac{(\Delta\mu - w_0)^3 (1 + W_j(z))}{\pi k_B T\, W_j{}^3(z)}}. \quad (9)$$

Зависимость барьера нуклеации ΔF^* (формула 2) и времени нуклеации (формула 4) от времени представлены на рисунке 4. В расчетах использовалось значение константы равновесиях химической реакции, определенное на основе справочных данных [15], $K^{eq} = 2.25 \times 10^{-4}$. Видно, что при $t \sim 1$ с выполнено условие $\tau_N \approx t$ [9], т.е. в этот момент времени высока вероятность формирования островка InAs на границе раздела катализатор-ННК, причем характерное время нуклеации составляет $\tau_N \sim 1$ с. Полученное значение времени нуклеации согласуется с характерным временем роста монослоя, оцененным на основе измерения скорости роста ННК [6].

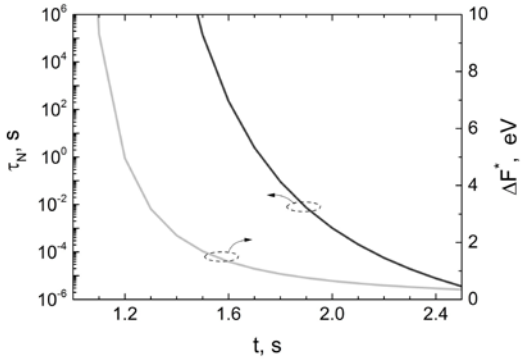


Рис. 4. Зависимость барьера нуклеации и характерного времени нуклеации от времени роста.

Отметим также, что в работах [9, 10] учитывались диффузионные потоки только группы 5, концентрация элементов группы 3 считалась заданной (~ 0.4).

В этом случае рассчитанное время нуклеации будет на несколько порядков меньше. Однако, характерное время роста монослоя остается прежним, так как определяется граничным потоком группы 5.

3. Заключение

Изучены процессы диффузии элементов группы 3 и 5 в частице катализатора при механизме роста пар-кристалл-кристалл ННК A^3B^5. Рассмотрен случай формирования золото-каталитических ННК InAs на подложке InAs(111).

Вычислены профили концентрации индия и мышьяка в объеме частицы катализатора в различные моменты времени роста монослоя.

Обнаружено, что из-за быстрой диффузии индия градиет концентрация индия в объме катализатора и на границе раздела катализатор-ННК близок к нулю.

На скорость роста островков A^3B^5 влияют диффузионные потоки индия как в объме, так и на границе раздела катализатор-ННК.

Диффузия мышьяка происходит преимущественно в тонком диффузионном слое вблизи границы раздела катализатор-ННК. Фактором, лимитирующим рост островков A^3B^5, является транспорт частиц группы 5 на границе диффузионного слоя и газовой фазы. Полученные оценки времени нуклеации островков A^3B^5 согласуются с экспериментальными данными.

Работа выполнена при поддержке НИР 0065-2018-0016 (номер гос. регистрации АААА-А18-118041890053-2).

Diffusion processes of elements of group 3 and 5 in catalyst particle during the vapor-solid-solid growth of A^3B^5 nanowires

A.A. Koryakin, S.A. Kukushkin

Abstract: The diffussion processes of elements of group 3 and 5 in catalyst particle during the vapor-solid-solid growth of A^3B^5 nanowires have been investigated. The formation of gold-cataluzed InAs nanowires on InAs(111) has been considered. The concentration profiles of indium and arsenic in catalyst particle have been calculated. It was foud that the volume diffusion flux and interfacial diffusion flux on the catalyst-nanowire interface both influence the A^3B^5 island growth rate. The arsenic diffusion occurs mainly in the thin diffusion laeyr nearby the catalyst-nanowire interface. The growth rate of A^3B^5 islands is limited by the material transport of arsenic species on the boundary between the diffusion layer and the gas phase.

Key words: nanowires, nucleation, semiconductors

Литература

1. В.Г. Дубровский, Г.Э. Цырлин, В.М. Устинов. Полупроводниковые нитевидные нанокристаллы: синтез, свойства, применения. «Физика и техника полупроводников». т. 43 (2009), № 12, 1585-1628.
2. V.G. Dubrovskii. Nucleation theory and growth of nanostructures. Heidelberg, Springer, 2014.
3. V.G. Dubrovskii, N.V. Sibirev. Growth rate of a crystal facet of arbitrary size and growth kinetics of vertical nanowires. «Phys. Rev. E». v. 70 (2004), 031604.
4. J. Johansson, M. Ghasemi. Composition of gold alloy seeded InGaAs nanowires in the nucleation limited regime. «Cryst. Growth Des.». v. 17 (2017), № 4, 1630–1635.
5. F. Glas, M. R. Ramdani, G. Patriarche, J.C. Harmand. Predictive modeling of self-catalyzed III-V nanowire growth. «Phys. Rev. B». v. 88 (2013), 195304.
6. M. Tchernycheva, L. Travers, G. Patriarche, F. Glas, J.C. Harmand, G.E. Cirlin, V.G. Dubrovskii. Au-assisted molecular beam epitaxy of InAs nanowires: growth and theoretical analysis. «J. Appl. Phys.». v. 102 (2007), 094313.
7. S.E.R. Hiscocks, W. Hume-Rothery. The equilibrium diagram of the system gold-indium. «Proc. R. Soc. Lond. A». v. 282 (1964), 318-330.
8. K.A. Dick. A review of nanowire growth promoted by alloys and non-alloying elements with emphasis on Au-assisted III-V nanowires. «Prog. Cryst. Growth Charact. Mater.». v. 54 (2008), 138-173.
9. А.А.Корякин, С.А.Кукушкин, Н.В.Сибирев. Механизм роста пар-кристалл-кристалл Аuкаталитических GaAs нитевидных нанокристаллов. «Физика и техника полупроводников». (2018) (в печати).
10. A.A. Koryakin, S.A. Kukushkin, K.P. Kotlyar, E.D. Leshchenko, E.V. Ubyivovk, R.R. Reznik, G.E. Cirlin, I.V. Shtrom, A.D. Bouravleuv. Catalytic growth of III-V nanowires with axial heterostructures. «VI International scientific conference STRANN 2018 – M., 17-19 октября 2018 г. (abstracts)», M. 2018, 28-30.
11. A.G. Khachaturyan. Theory of structural transformations in solids. New York, Dover, 2008.
12. С.А. Кукушкин, А.В. Осипов. Процессы конденсации тонких пленок. «Успехи физических наук». т. 168 (1998), № 10, 1083-1116.
13. Y. Cai, S.K. Chan, I.K. Sou,Y.F. Chan, D.S. Su, N. Wang. The size-dependent growth direction of ZnSe nanowires. «Adv. Mater.». v. 18 (2006), 109-114.
14. D. McLean. Grain boundaries in metals. Oxford, Clarendon Press, 1957.
15. I. Ansara, C. Chatillon, H.L. Lukas, T. Nishizawa, H. Ohtani, K. Ishida, M. Hillert, B. Sundman, B.B. Argent, A. Watson, T.G. Chart, T. Anderson. «CALPHAD». v. 18 (1994), № 2, 177-222.
16. Y. Hasumi. Lateral diffusion of In and formation of AuIn2 in AuIn thin films. «J. Appl. Phys.». v. 58 (1985), 3081-3086.
17. D.B. Butrymowicz, J.R. Manning, M.E. Read. Diffusion in copper and copper alloys. Part V. Diffusion in systems involving elements of group VA. «J. Phys. Chem. Ref. Data». v. 6 (1977), № 1, 1-50.

Численное моделирование гетерогенного течения расплава минерального сырья в пористой среде.

Ю.А.Куракин

СПбО МСЦ РАН – филиал ФГУ ФНЦ НИИСИ РАН и ФТИ им. А.Ф. Иоффе РАН, Санкт-Петербург, Россия

E-mail: yurii.kurakin@mail.ioffe.ru

Аннотация: Представлена математическая модель и результаты расчётов гетерогенного течения расплава минерального сырья в пористой среде. С помощью модели исследовались процессы в коксовой плавильной печи-вагранке. Рассмотрены процессы, определяющие конечную температуру расплава на выходе из печи. Изучено влияние на температуру расплава изменения режима работы печи, в частности, содержания кокса в шихте и температуры подаваемого в печь дутьевого воздуха.

Ключевые слова: численное моделирование, течение жидкости через пористую среду, плавильная печь для минерального сырья.

Введение

В предыдущей работе авторов [1] были представлены математическая модель и результаты расчетов процессов тепло-массообмена, связанных с горением кокса, химическими превращениями и плавлением минерального сырья в плавильной печи. Однако картина происходящего в печи не будет полной без рассмотрения эволюции жидкой фазы – расплава базальта.

Основная часть работ по моделированию течения расплавов в вагранках посвящено металлургическим печам [2-5]. Такие же подходы справедливы и для плавильных вагранок. И в том, и в другом случае течение расплава по поверхности кокса является не смачиваемым, а объёмная доля расплава составляет несколько процентов.

Область плавления базальта находится выше уровня ванны с расплавом в донной области печи, и отделена от него слоем чистого кокса. Из области плавления расплав стекает вниз в виде отдельных струек, заполняя лишь частично объём пор между частицами кокса. Такая картина течения подтверждается многочисленными наблюдениями работающих печей, как металлургических, так и печей для плавления минерального сырья. По этим причинам модели сплошных сред для расплавов не годятся, требуется рассматривать динамику отдельных капель.

Математическая модель

Баланс сил, действующих на отдельные струи расплава выглядит так:

$$\vec{F}_g + \vec{F}_{sl} + \vec{F}_{gl} = 0$$

где $F_g = V_l \rho_l g \vec{z}$ - сила тяжести, V_l - суммарный объём струй в единице объёма, ρ_l - плотность расплава, g - ускорение свободного падения, z - вертикальный единичной вектор, $\vec{F}_{sl}$ - сила сопротивления пористого слоя, $\vec{F}_{gl}$ - сила трения газового потока, проходящего через поры:

$$\vec{F}_{gl} = C_D S_{gl} \tfrac{1}{2} \rho_g \vec{u}_g \left| u_g \right| ,$$

где C_D - коэффициент сопротивления, ρ_g, u_g - плотность и скорость газа, а S_{gl} - площадь поверхности контакта струй и газа в единице объёма. Коэффициент сопротивления C_D рассчитывается по известной формуле, полученной Эргуном [6]:

$$C_D = \begin{cases} \dfrac{24}{\mathrm{Re}}\left(1 + 0.15\,\mathrm{Re}^{0.687}\right) & \mathrm{Re} \le 1000 \\ 0.44 & \mathrm{Re} > 1000 \end{cases},$$

число Рейнольдса Re определяется по скорости газа и характерному размеру пор.

С определением силы сопротивления пористого слоя $\vec{F}_{sl}$ существуют определенные трудности, поскольку строгая теория движения струй жидкости через поры к настоящему времени не разработана. Обычно используют одно из уравнений движения сплошной среды через пористые материалы (Козени-Кармана, Дарси или Эргуна) с поправкой на то, что контакт с жидкостью занимает только часть поверхности пор. Законность такого подхода проверялась экспериментально [2,3].

В форме уравнения Эргуна сила сопротивления $\vec{F}_{sl}$ будет иметь выражение:

$$\vec{F}_{sl} = k_1 S_{sl}^2 \mu_l \vec{u}_l + k_2 S_{sl} \rho_l \vec{u}_l |u_l| ,$$

где u_l – скорость расплава, μ_l – вязкость раплава, S_{sl} – площадь контакта жидкости и поверхности пор в единице объёма.

Коэффициенты k_1 и k_2 определяются либо напрямую эмпирически [2,3], либо через коэффициент удержания жидкости пористой средой [4,5], для которого также существуют эмпирические выражения.

Другой открытый вопрос – определение характерного размера d_l капель или струек расплава, через который рассчитывается площадь поверхности расплава, и который участвует в уравнениях для k_1 и k_2. Например, в расчётах [7] размер капель напрямую брался из собственного эксперимента, в котором печь в момент работы “захолаживалась” азотом, затем разрезалась, и по срезам определялся характерный размер всех составляющих шихты.

В работах [2,3] коэффициент k_2 определялся из эксперимента, а первым членом уравнения Эргуна пренебрегалось, поскольку течение расплава считалось турбулентным и не зависящим от его вязкости, что может быть справедливо для расплава железа, но не годится для расплава базальта, обладающего большей вязкостью.

В описываемой здесь модели использованы результаты работы [5], где коэффициенты k_1 и k_2, а также размер струй d_l рассчитывались на основе коэффициента удержания жидкости пористым слоем и законов поведения капель жидкости на не смачиваемых поверхностях. Там же приводятся данные для металлургических вагранок по размерам струй расплавов шлаков, имеющих сходные с базальтами вязкость и коэффициент поверхностного натяжения. При сравнении термодинамических свойств шлаков и базальтов, данные для базальтов брались из работы [8].

Система уравнений динамики расплава состоит из уравнения неразрывности и уравнения баланса энергии:

$$\oint_V \rho_l' \, u_l dV = Q ,$$

$$\oint_V \rho_l' \, u_l C_l T_l dV = E_{\lg} + E_r ,$$

где ρ_l' - масса расплава в единице объёма пористой среды, V - объём, Q - массовая скорость образования расплава в единице объёма, T_l - температура расплава, C_l - теплоёмкость расплава, $E_{\lg}$ - интенсивность контактного теплообмена расплава с газом [1], E_r - интенсивность радиационного теплообмена расплава с газом и коксом. Распределение скорости расплава u_l определяется из решения уравнения баланса сил, приведенного выше.

Модель радиационного теплообмена взята из работы [7]. В модели использованы экспериментальные измерения по излучательной способности компонентов в печи

$$E_r = \sigma\left[\varepsilon_g S_{\lg}\left(T_g^4 - T_l^4\right) + S_l\left(\varepsilon_c T_c^4 - \varepsilon_l T_l^4\right)\right],$$

где T_g, T_c - температура газа и кокса, $\varepsilon_g = 0.01$, $\varepsilon_c = 1$, $\varepsilon_l = 0.85$ - излучательная способность газа, кокса и расплава, S_l - площадь поверхности расплава в единице объёма. У этой модели имеются следующие допущения: объёмное содержание расплава настолько мало, что радиационного теплообмена между каплями расплава не происходит, контакт капель с поверхностью кокса настолько мал, что контактным теплообменом расплава и кокса пренебрегается.

Численный метод.

Расчеты показывают, что энергия, расходуемая на нагрев расплава, составляет менее 1% в общем тепловом балансе печи. Это даёт основание пренебречь обратным влиянием расплава на газ и шихту. В расчетах течения расплава использованы готовые распределения параметров газа и шихты, полученные на предыдущем этапе численных исследований [1].

Задача осесимметричная. Уравнения переноса записываются для объёма ячеек регулярной расчетной сетки.

На первом этапе находится распределение скорости расплава из решения уравнения баланса сил.

На втором этапе решаются уравнения переноса, причем, потоки массы и энергии через грани ячеек рассчитываются по противопоточной схеме. На гранях ячеек скорость расплава аппроксимируется из центров ячеек, а значения плотности и температуры расплава берутся из той ячейки, откуда поток исходит, согласно направлению скорости.

Поскольку расплав движется преимущественно вниз, уравнения решались маршевым методом по вертикальной координате.

Результаты моделирования плавильной печи.

Математическая модель применялась для исследования процессов в плавильной печи, установленной на предприятии ООО “ИЗОМИН” (Московская область).

Печь представляет собой вертикальную трубу переменного сечения (Рис.1):

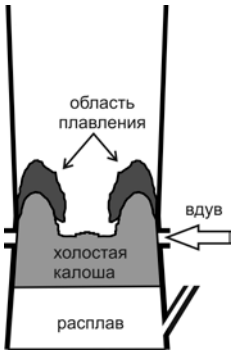


Рис.1. Изображение печи и распределение характерных областей во время работы, полученное в расчётах. На рисунке показана область наиболее интенсивного плавления базальта.

Характерный радиус печи ~ 1 м, высота ~ 4 м. В печь сверху подается смесь кокса с сырьём (базальт + доломит). Воздух вдувается через фурмы. Параметры работы печи в основном режиме, используемом на производстве следующие: температура воздуха – 200 С, содержание кокса в шихте – 20%.

В процессе работы внизу печи образуется холостая калоша, состоящая из чистого кокса, причем соотношение загружаемого в печь кокса и сырья подобрано таким образом, чтобы размер калоши сохранялся.

Получаемый в расчетах уровень калоши у стенок печи выше, чем в центре (Рис.1.), что также наблюдается в эксперименте [7]. Объясняется это следующим образом. Кислород подаваемого в печь воздуха выгорает в непосредственной близости фурм [1]. Горячий газ из зоны горения кокса поднимается вверх, в основном, вдоль стенок печи и почти не проникает в её центр. Вблизи стенок над холостой калошей образуется область интенсивного плавления базальта, где происходит высвобождение чистого кокса.

Протяженность полученной в расчётах области наиболее интенсивного плавления составляет 0.23-0.9 м от уровня фурм. Для сравнения, в эксперименте [7] протяженность области составила 0.35-1 м.

Расплав базальта из зоны плавления протекает вниз через холостую калошу и скапливается в ванне в нижней части печи. Уровень расплава в ванне определяется положением сливного отверстия и находится ниже уровня фурм.

Температура плавления базальтов лежит в пределах 1250-1450 С. При плавлении получаемый базальтовый расплав содержит различные включения в виде пузырьков и свилей. Обычно в технологии производства после плавления расплав сначала нагревают на несколько сот градусов для интенсификации дегазации и частичного растворения свилей, а затем остужают до температуры 1500-1600 С, оптимальной для волокнообразования [9].

Подобная картина эволюции расплава наблюдается и в представленном расчёте. На Рис.2. показаны линии тока расплава:

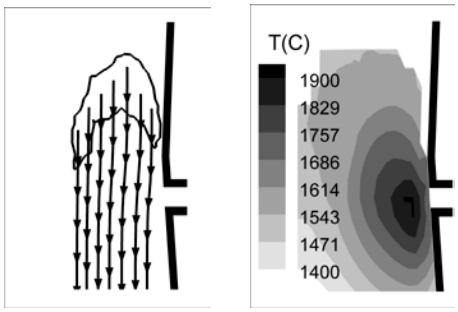


Рис.2. Линии тока и распределение температуры расплава в основном режиме.

Вблизи фурм наблюдается незначительное отклонение потока расплава из-за влияния струи воздуха от вдува, но основная часть расплава проходит через фурменную область, в которой температура кокса и газа максимальна. Там расплав нагревается (Рис.2.), а затем, двигаясь вниз от фурменной области, остывает.

Замеры пирометром температуры расплава после выхода из печи в отводящем лотке показали значение ~1550 С. Очевидно, что температура в ванне выше этой величины на ~100-200 градусов, поскольку дойдя до лотка, расплав успевает остыть.

В расчетах получена температура расплава в ванне ~1540 С. Это значение несколько ниже реальной температуры. Объяснить это можно чрезмерным охлаждением расплава в области, расположенной ниже фурм, что, в свою очередь, обусловлено заниженной расчетной температурой кокса в этой области. Это говорит о том, что обратное влияние жидкой фазы на твердую, в данном случае это – разогрев кокса горячим расплавом, в модели нужно учитывать.

Тем не менее, сравнение расчетов с экспериментом по температуре расплава даёт

расхождение всего в 5-10%, что можно считать удовлетворительным результатом.

Влияние изменения режимов работы печи на температуру расплава.

Ранее в [1] исследовалось влияние снижения содержания кокса в шихте и повышения температуры дутьевого воздуха на картину процессов в печи.

Расчеты показывают, что в обоих случаях температура расплава в ванне растёт (Рис.3, 4), но причины этого роста в каждом случае разные.

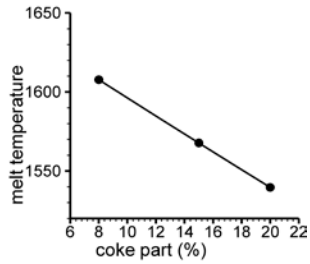


Рис.3. Зависимость температуры расплава от содержания кокса в шихте.

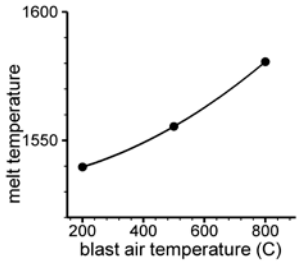


Рис.4. Зависимость температуры расплава от температуры дутьевого воздуха.

Сравнивая форму холостой калоши на Рис.4 и Рис.2, можно видеть, что после снижения процента кокса в шихте происходит вырождение калоши, её объём уменьшается, а в область фурм вдоль стенок печи начинает поступать базальт. При этом область горения кокса отодвигается от стенки печи и захватывает большую часть потока расплава по сравнению с основным режимом. Это хорошо видно при сравнении распределений температуры расплава на Рис.4 и Рис.2. Область максимальной температуры на этих рисунках совпадает с областью горения кокса.

При увеличении температуры дутьевого воздуха положение области горения такое же, как и в основном режиме (сравнение Рис.6 и Рис.2), однако температура горения выше, что является следствием горячего дутья. Соответственно растёт и температура расплава (Рис.6) .

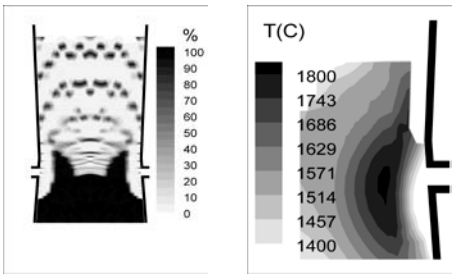


Рис.5. Состояние холостой калоши (слева) и распределение температуры расплава (справа) при содержании кокса в шихте 10%.

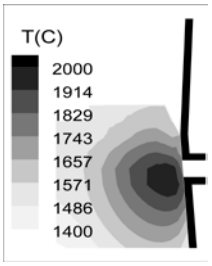


Рис.6. Распределение температуры расплава при температуре дутьевого воздуха 800С.

Заключение

Разработана математическая модель течения расплава базальта через пористую среду в плавильной печи-вагранке.

Проведены расчеты движения расплава для печи, используемой на производстве. Установлено, что главным фактором, определяющим распределение температуры расплава на выходе из печи, является взаимодействие потока расплава с областью горения кокса вблизи фурм. В область горения расплав попадает при просачивании из зоны плавления через холостую калошу в донную часть печи.

Изменение состава шихты и параметров дутьевого воздуха меняет положение и температуру области горения кокса, что сказывается на распределении параметров расплава. Температура расплава в ванне растет при снижении содержания кокса в шихте, и при увеличении температуры дутьевого воздуха.

Сравнение расчётов с замерами на работающей печи показали, что предложенная модель даёт заниженную на 5-10% температуру расплава в ванне. Предположительно это вызвано пренебрежением обратным влиянием расплава на шихту.

Работа выполнена в рамках НИР 0065-2018-0016.

Numerical simulation of heterogeneous flow of mineral raw melt in porous media.

Y.A.Kurakin

Abstracts: Mathematical model and numerical calculations of heterogeneous flow of mineral melt in porous media are presented. The model was applied for investigations of the processes in the coke cupola for mineral raw melting. The processes which govern the melt temperature in cupola outlet are considered. The effects of the cupola operation mode variation on the melt temperature were investigated. The mode variation includes the change of coke part in cupola charge material and the change of blast air temperature.

Keywords: numerical simulation, liquid flow in porous media, cupola for mineral raw melting.

Литература

[1] П.А.Войнович, Ю.А.Куракин. Численные исследования эффектов нагрева дутья в коксовой плавильной печи для минерального сырья. Труды НИИСИ РАН т.7 № 4, 2017.
[2] G.S.Gupta, J.D.Lister, V.R.Rudolph, E.T.White, A.Domanti. Model Studies of Liquid Flow in the Blast Furnace Lower Zone. ISIJ International, Vol. 36 (1996), No. 1, pp. 32-39
[3] G.S.Gupta, J.D.Lister, E.T.White, V.R.Rudolph. Nonwetting Flow of a Liquid through a Packed Bed with Gas Cross-Flow. Metallurgical and Materials Transactions B, Vol. 28B, Aug. 1997-597
[4] S.J.Chew, P.Zulli, A.Yu. Modelling of Liquid Flow in the Blast Furnace. Application in a Comprehensive Blast Furnace Model. ISIJ International, 2001, Vol.41, No.10, pp.1122-1130
[5] S.J.Chew, P.Zulli, A.Yu. Modelling of Liquid Flow in the Blast Furnace. Theoretical Analysis of the Effects of Gas, Liquid and Packing Properties. ISIJ International, 2001, Vol.41, No.10, pp.1112-1121
[6] Ergun S. Fluid flow through packed columns. Chemical Engineering Progress v.48, 1952, 89-94.
[7] R. Leth-Miller, A. D. Jensen, P. Glarborg, L. M. Jensen, P. B. Hansen, S. B. Jorgensen. Investigation of a Mineral Melting Cupola Furnace. Part I, II. Ind. Eng. Chem. Res. 2003, 42, 6872-6892
[8] О.С.Татаринцева, Д.Е.Зимин. Особенности плавления горных пород и волокнообразования из расплавов. Ползуновский вестник № 2, 2006.
[9] Д.Д.Джигирис, М.Ф.Махова. Основы производства базальтовых волокон и изделий. Москва, Теплоэнергетик, 2002.

О влиянии технологии гидродинамических исследований скважин на результаты интерпретации

А.А. Колеватов, Ю.М. Штейнберг, А.К. Пономарев, А.Г. Дяченко, Д.В. Солопов

ФГУ ФНЦ НИИСИ РАН, Москва, Россмя, E-mail: akolevatov@niisi.ras.ru

Аннотация: В статье рассматриваются вопросы влияния технологии гидродинамических исследований на результаты интерпретации. Приводится пример исследования, регистрировавшегося в условиях влияния ствола скважины, а также вероятные фильтрационно-емкостные свойства, которые могли иметь место в случае применения технического устройства исключающего влияние ствола скважины. Рассматриваются различия в величинах фильтрационно-емкостных свойств.

Ключевые слова*:* гидродинамические исследования, двойная проницаемость, влияние ствола скважины, забойный штуцер-отсекатель.

Введение

Корректность результатов гидродинамических исследований (ГДИ) зависит от множества факторов [1]. Условно эти факторы разделяются на технические и технологические. К техническим относят точность и частоту регистрации изменений давления. К технологическим относят время работы скважины на режимах до остановки, длительность остановки скважины на исследования, отсутствие изменений режимов соседних скважин. Также к технологическим факторам относится место установки глубинного манометра в стволе скважины и наличие или отсутствие изоляции трубного и затрубного пространства насосно-компрессорной трубы (НКТ). Наличие или отсутствие изоляции затрубного пространства влияет на регистрируемое глубинным манометром забойное давление в скважине. В зависимости от свойств пласта и нефти длительность такого влияния может составлять до нескольких часов. Это связано с тем, что после закрытия скважины приток нефти в ствол скважины не приводит к соответствующему росту давления в точке регистрации. Вместо этого нефть преимущественно поступает в затрубное пространство и регистрируемое давление искажается процессами в трубном и затрубном пространстве (дегазация, остывание, вертикальная конвекция). Степень влияния ствола скважины (ВСС) не играет большой роли при исследовании скважин в стандартных терригенных коллекторах, фильтрация в которых чаще всего описывается интерпретационной моделью «однородный пласт». Наиболее заметное ВСС оказывается на исследования карбонатных трещиноватых коллекторов, либо на исследования коллекторов сложного строения, таких как баженовская свита [2]. В этом случае ВСС может «скрыть» или исказить процессы перетоков флюидов между пластами с разными свойствами, либо между двумя фильтрующими средами - матрицей и высокопроницаемыми каналами (трещинами). Приведенные ниже результаты исследования рассматривают степень влияния ствола скважины на значения фильтрационно-емкостных свойств (ФЕС), определенных по результатам ГДИ в скважине, вскрывшей нефтенасыщенные отложения баженовской свиты.

Описание объекта исследования

На рисунке 1 приведен пример разреза отложений баженовской свиты и нижележащих пород, описывающий наиболее часто встречающуюся структуру продуктивного пласта.

Промышленная нефтеносность месторождений, вскрывших отложения баженовской свиты, связана с отложениями

баженовско-верхнеабалакского комплекса верхней юры, тюменской свиты средней юры и трещиновато-поровыми эффузивами доюрского комплекса. Основной интерес представляют порово-кавернозно-трещинные карбонаты пласта $Ю_0$, а также трещиноватые эффузивы доюрского возраста. Большие запасы нефти содержат алевролито-глинистые пласты тюменской свиты, однако они являются слабоизученными и, в первую очередь, требуют раздельного испытания в колонне.

Продуктивные коллектора пласта $Ю_0$ считаются наиболее перспективными для пробной эксплуатации на естественном режиме.

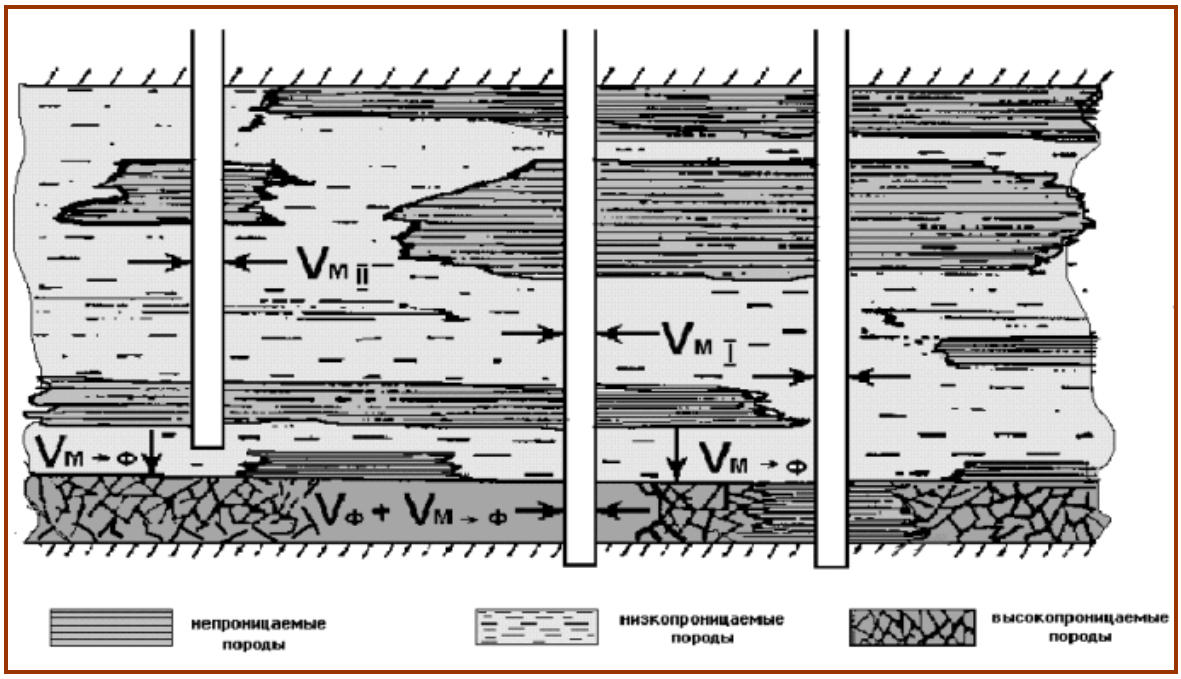


Рисунок 1. Пример разреза продуктивных отложений баженовско-верхнеабалакского комплекса верхней юры, тюменской свиты средней юры и трещиновато-поровых эффузивов доюрского комплекса.

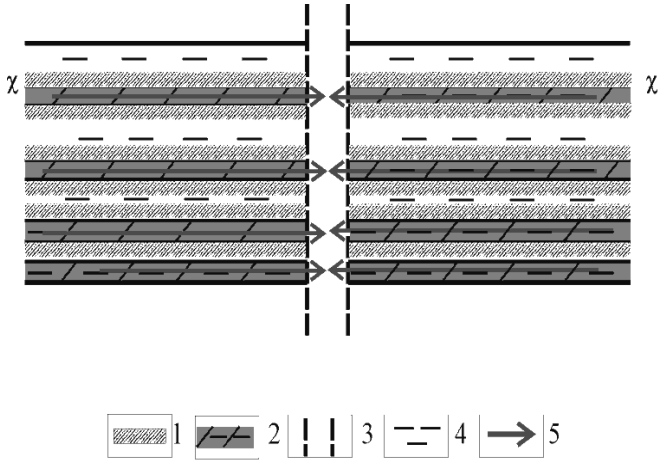


Рисунок 2. Пример схемы строения нижнетутлеймской подсвиты в пределах Галяновского и Средне-Назымского участков: 1 - слой χ нефтематеринской породы; 2 – коллекторский прослой; 3 - перфорированный ствол скважины; 4 – битуминозные глины; 5 - пути миграции нефти в скважину.

Верхнеюрский нефтегазоносный подкомплекс (баженовско-верхнеабалакский) является регионально нефтегазоносным. Комплекс выдержан по толщине и составу на огромной территории. Литологически продуктивная часть комплекса представлена переслаиванием кремнисто-глинистых пород, опок и черных глинистых известняков. Комплекс включает пласты $ЮК_1$ (верхнеабалакская подсвита) и $Ю_0$ (баженовская-нижнетутлеймская подсвита)

Приведенная на рисунке 2 иллюстрация структуры коллекторов в современном представлении получила название «двойной проницаемости». Это означает, что условно разобщенные, либо слабо сообщающиеся по системе трещин проницаемые прослои могут математически описываться как коллектор, состоящий из двух продуктивных пластов. При проведении ГДИ такие коллектора математически описываются [3, 4] уравнениями, включающими два пласта с разной проводимостью, и имеющими характерный вид диагностического графика (рис. 3)

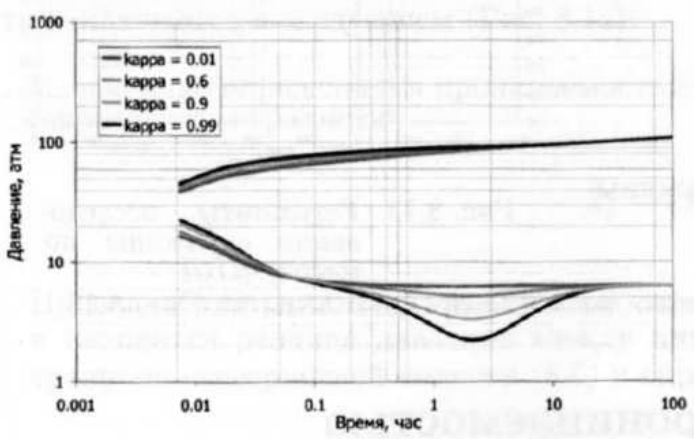


Рисунок 3. Диагностический график модели двойной проницаемости.
Первый случай — скважина вскрывает оба продуктивных пропластка [3]

В модели двойной проницаемости наиболее проницаемый пласт характеризуется величинами k_1 и h_1. Наименее проницаемый пласт характеризуется величинами k_2 и h_2.

Систему описывают 3 дополнительных параметра:

• доля наиболее проницаемого пласта к емкости системы:

$$\omega = \frac{(\phi c_t h)_1}{(\phi c_t h)_1 + (\phi c_t h)_2},$$

где:Ф – пористость; c_t – общая сжимаемость, h_1 и h_2 – мощности пластов

• удельный коэффициент проводимости между пластами:

$$\lambda = \frac{k_2 h_2}{k_1 h_1 + k_2 h_2} \alpha r_w^2,$$

где: r^2_w – приведенный радиус скважины, k_1 и k_2 – проницаемости пластов

• отношение kh (проводимость) наиболее проницаемого пласта к общей kh системы:

$$\kappa = \frac{k_1 h_1}{k_1 h_1 + k_2 h_2}$$

Первый случай - скважина вскрывает оба продуктивных пропластка. В начальный момент времени система действует как два однородных пласта и её можно охарактеризовать как однородный пласт с суммарной kh системы. Когда наиболее проницаемый пласт работает более активно, начинает проявляться перепад давления между пластами и, соответственно, переток между ними. Переходный процесс характеризуется отклонением производной вниз на диагностическом билогарифмическом графике (рис. 3).

Когда κ равно 1, то проницаемость менее проницаемого пласта равна 0, и поведение системы соответствует двойной пористости. Менее проницаемый слой представляет матрицу, приток из которой возможен при вскрытии наиболее проницаемого пласта (трещина гидроразрыва или естественная трещиноватость).

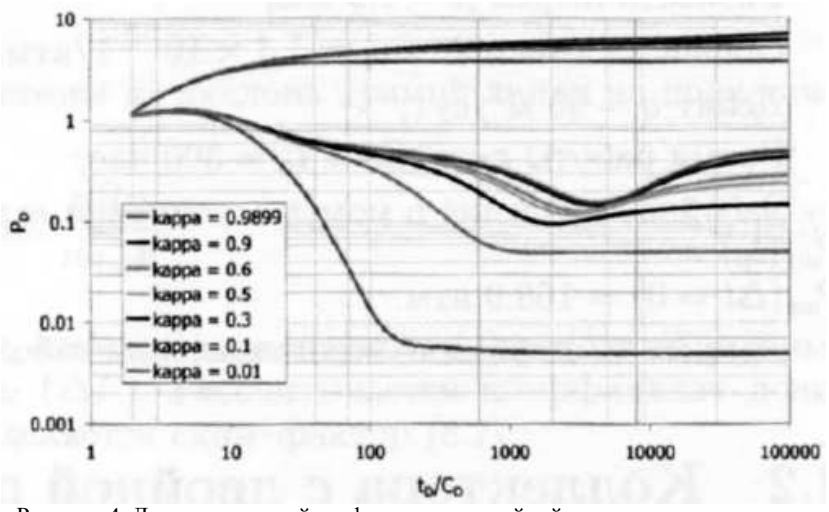


Рисунок 4. Диагностический график модели двойной проницаемости. Второй случай — скважина вскрывает только первый пропласток [3].

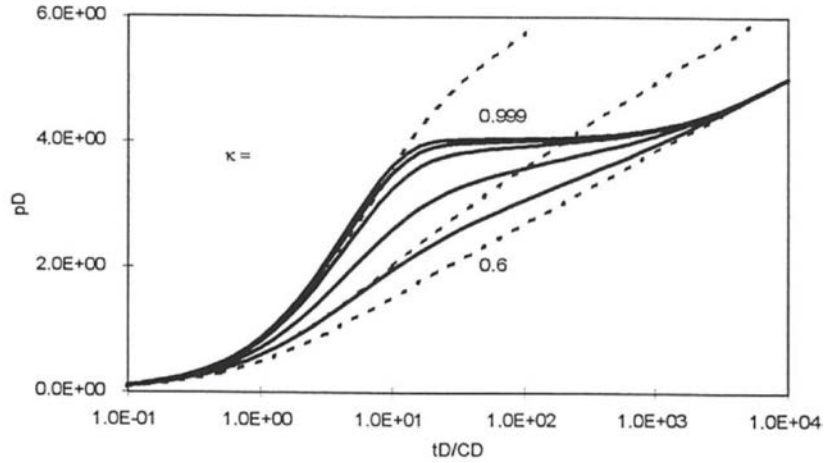


Рисунок 5. Диагностический график модели двойной проницаемости (рис. 4) в полулогарифмических координатах [4].

Гидродинамические исследования скважин, вскрывших отложения баженовской свиты

При вскрытии скважиной керогенсодержащих пород баженовской свиты (аналогично рис. 1 и 2) возможны оба варианта дренирования (рис. 3 и 4) подвижных запасов нефти. Это было подтверждено гидродинамическими исследованиями добывающих скважин разных месторождений посредством регистрации кривых восстановления давления (КВД). Результаты ГДИ подтвердили механизм фильтрации согласно интерпретационной модели «двойная проницаемость» (рис. 6 и 7).

Величины безразмерных параметров κ, λ и ω, характеризующих перетоки нефти между высоко- и низко-проницаемыми прослоями, соотношение запасов и соотношение проводимости пластов по исследованиям скважин №№ Х19 и ХХ19 составили 0.89-0.99; $5.49*10^{-8}$-$1.84*10^{-7}$ и 0.167 соответственно.

И в том и в другом случае скважины не имели в своей конструкции элементов, изолирующих затрубное пространство и позволяющих перекрывать в самом низу отверстие НКТ. Другими словами, на регистрируемые изменения давления в скважине после остановки имело место ВСС.

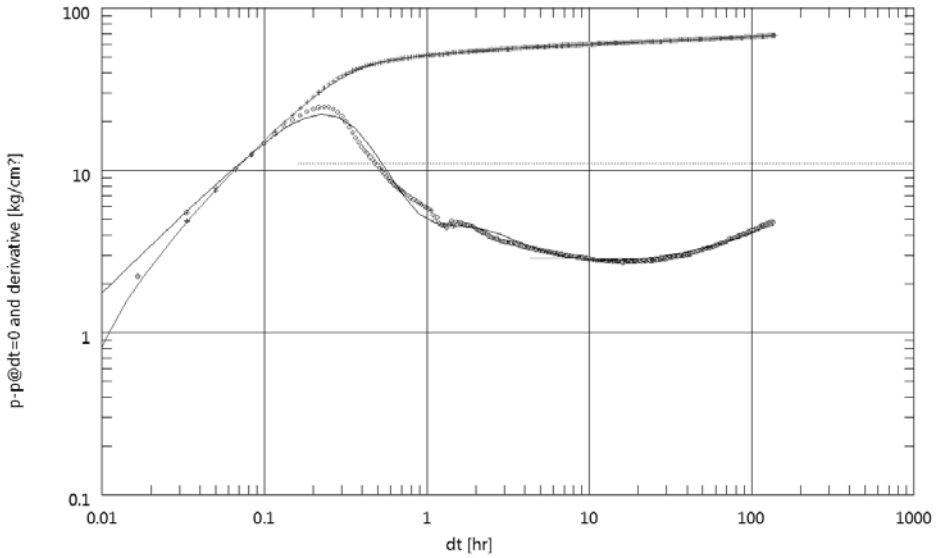


Рисунок 6. Диагностический график модели двойной проницаемости в билогарифмических координатах по скважине № Х19. Месторождение СН. Исследование 2005 года.

В рамках выполнения исследований (Программа ФНИ государственных академий наук на 2013-2020 годы. «Создание методики выявления невыработанных зон на нефтяных месторождениях и подсчёта остаточных запасов нефти на основе комплексирования математического моделирования, анализа разработки с исследованиями скважин и пластов») специалистами отдела ОГИМНГО ФГУ ФНЦ НИИСИ РАН выполнена разработка, позволяющая исключить влияние ствола скважины на регистрируемое забойное давление. Подана заявка на патентование с рег. № 2018138987 от 06.11.2018 «Скважинный штуцерный клапан отсекатель».

Основные преимущества разработки: - возможность управления притоком в скважину с поверхности (открытие и закрытие), возможность пошагового регулирования притока в скважину, возможность снижения ВСС до минимальных величин в случае установки пакера и изолирующего устройства в непосредственной близости от верхних дыр перфорации.

С учетом преимущества минимизации ВСС на регистрацию изменений забойного давления при остановке скважины для регистрации КВД было смоделировано исследование скважины №ХХ19 при отсутствии влияния ствола скважины (рис. 8). Смоделированная запись забойного давления при остановке скважины для регистрации КВД интерпретировалась с учетом следующих параметров: - не изменялся скин-фактор (величина характеризующая степень совершенства связи скважины с пластом); - не изменялась величина экстраполированного пластового давления для зоны дренирования скважины; - не изменялась величина проницаемости менее проницаемого пласта (участок графика характеризующий эту величину не подвергается ВСС). При определении ФЕС методом наилучшего совмещения для случая отсутствия ВСС производилась адаптация безразмерных параметров κ, λ и ω, характеризующих перетоки нефти между высоко- и низко-проницаемыми прослоями, соотношение запасов и соотношение проводимости пластов. Соотношение ФЕС для начального исследования скважины № 3019 и смоделированного исследования, не имеющего ВСС, приведено в таблице 1.

На рисунке №9 приведен совмещенный диагностический график с наложением исходной кривой и смоделированной для более наглядной иллюстрации влияния ствола скважины на регистрируемые данные по изменению давления в остановленной скважине.

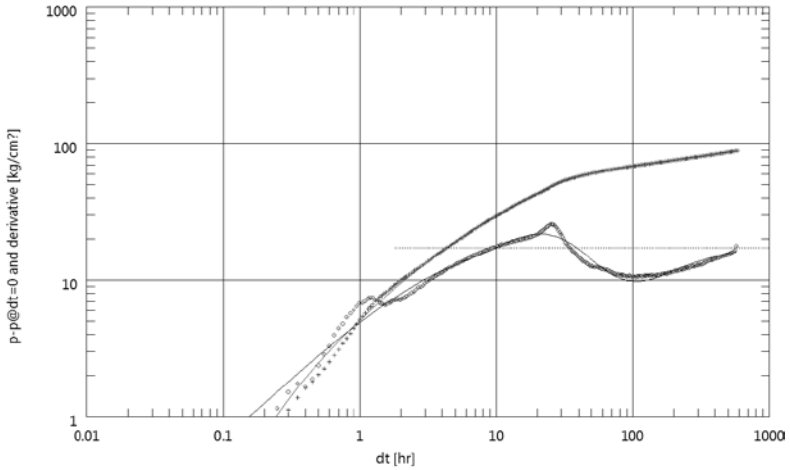


Рисунок 7. Диагностический график модели двойной проницаемости в билогарифмических координатах по скважине № ХХ19. Месторождение СН. Исследование 2017 года.

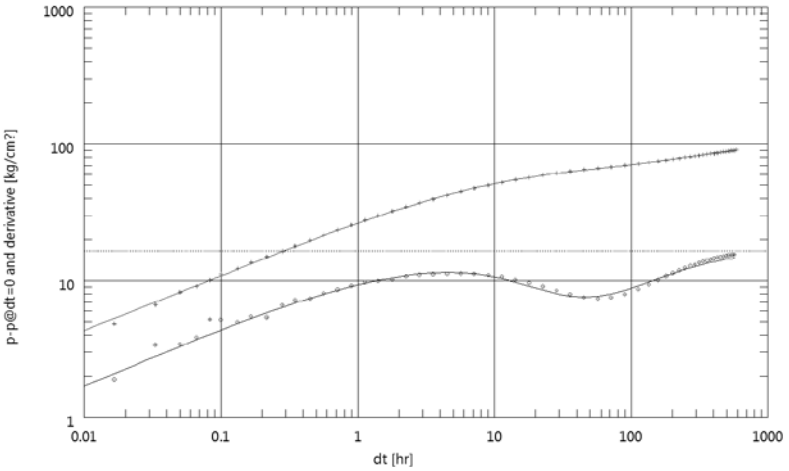


Рисунок 8. Диагностический график модели двойной проницаемости в билогарифмических координатах по скважине № ХХ19. Месторождение СН. Смоделированное исследование 2017 года с исключенным влиянием ствола скважины.

Таблица 1. Соотношение ФЕС по скважине №ХХ19 с учетом ВСС и в его отсутствии

Параметр, ед. изм	ФЕС с влиянием ВСС	ФЕС без влияния ВСС
Рпл экстраполированное, кгс/см^2	258.78	258.78
Скинфактор пласта №1, безразм	5.95	6.11
Скинфактор пласта №2, безразм	6.33	6.24
Продимость, kh mD/m	15.7	15.7
ВСС, м^3*см^2/кг	0.201	-
к, безразм	0.99	0.96
λ, безразм	$2.12*10^{-7}$	$1.33*10^{-7}$
ω, безразм	0.119	0.06

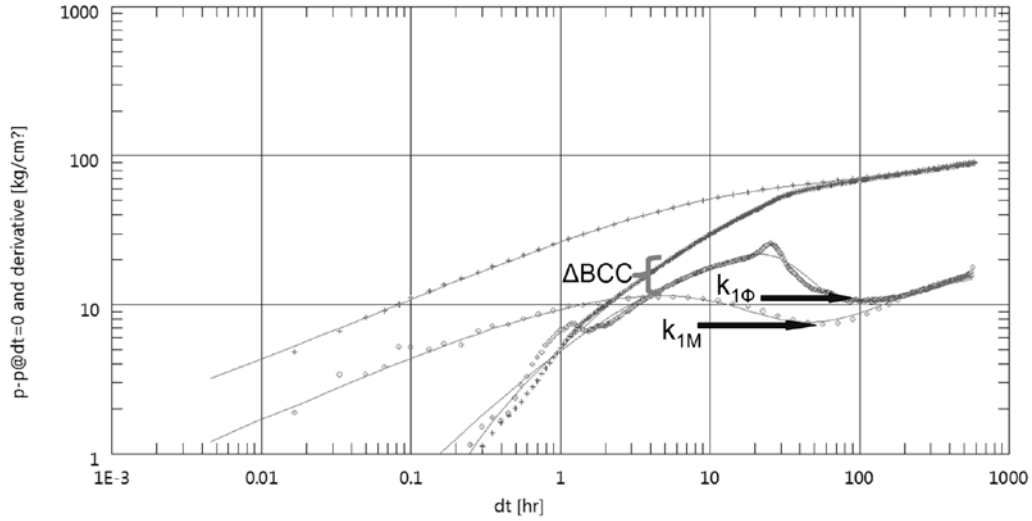


Рисунок 9. Совмещенный диагностический график модели двойной проницаемости в билогарифмических координатах по скважине № XX19. Месторождение СН. Фактические и смоделированные данные по исследованию 2017г. (ΔBCC – разница производных давления при наличии/отсутствии влияния ствола скважины; k_{1M} – положение участка характеризующего проницаемость наиболее проницаемого пропластка для смоделированного примера; $k_{1Ф}$ – положение участка характеризующего проницаемость наиболее проницаемого пропластка для фактических данных при наличии ВСС)

Заключение

Анализ величин ФЕС из Таблицы №1 и Рисунка №9 показал, что при практически неизменных величинах скин-фактора, пластового давления и параметра проводимости одно и тоже исследование в зависимости от технологии может иметь значительные отличия в результатах и дальнейших заключениях:

1. Параметр κ, характеризующий соотношение проводимости двух пластов, отличался на 0.03. При небольшой абсолютной разнице отличие этого параметра от 0.99 означает, что менее проницаемый пласт имеет дренируемые запасы нефти. И эти запасы возможно частично извлечь посредством дренирования через более проницаемый пропласток, т.к. он охватывает гораздо большую площадь.

2. Парметр ω, характеризующий соотношение емкостей в общих запасах системы двух пластов и отличающийся примерно на 50% для случая отсутствия влияния ствола скважины указывает на то, что наиболее проницаемый пропласток содержит небольшую часть запасов, а основным источником дренируемых запасов являются низкопроницаемые, но гораздо более мощные пласты.

3. Параметр λ, характеризующий наличие или отсутствие перетока между высоко и низкопроницаемым пластом, снизился на 30% относительно фактических данных, что также может свидетельствовать о более активном массообмене между пластами, чем предполагалось на основе фактических данных.

4. Изменения параметров в пунктах 1-3 Заключения, основанные на модели, свидетельствуют о целесообразности внедрения в производство упомянутой выше разработки управляемого устройства для отделения трубного пространства в момент остановки скважины на исследования, т.к. в этом случае значительно уменьшаются искажения на регистрируемые изменения забойного давления в остановленной скважине.

Работа выполнена при поддержке Программы ФНИ государственных академий наук на 2013-2020 гг, проект № 0065-2018-0118.

Well test interpretation results dependency on survey technology

A.A. Kolevatov, Y.M. Steinberg, , A.K. Ponomarev,

A.G. Dyachenko, D.V. Solopov

Abstract: Article overviews interpretation results dependency on well test technology. Survey example results presented for conditions with wellbore storage influence and adjusted hypothetically without wellbore storage influence in case of downhole operated choke application. Filtration-conductivity properties differences for both cases are discussed.

Keywords: well testing, dual-permeability reservoir, wellbore storage influence, downhole operated choke

Литература

1. П.В.Крыганов. Методы повышения достоверности результатов гидродинамических исследований нефтяных пластов и скважин: диссертация кандидата технических наук. Москва. 2012. 133с.
2. С.Г.Вольпин, О.В.Ломакина, И.В.Афанаскин. Особенности геологического строения и энергетического состояния залежи в отложениях баженовской свиты. Материалы международной научно-технической конференции Geopetrol 2014, Exploration and production of oil and natural gas reservoirs – new technologies, new challenges (Krakow, 15-18.09.2014) – с. 85-95.
3. Гидродинамические исследования скважин: анализ и интерпретация данных. Т.А.Деева, М.Р.Камартдинов, Т.Е.Кулагина, П.В.Мангазеев. – Томск 2009.
4. Интерпретация результатов исследования скважин. Д.Бурдэ.

Модели двойной пористости и двойной проницаемости для интерпретации исследований скважин методом двух режимов

И.В. Афанаскин[1], С.Г. Вольпин[2], Ю.М. Штейнберг[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]ivan@afanaskin.ru, [2]sergvolpin@gmail.com, [3]yurash22@gmail.com

Аннотация: Рассмотрены особенности интерпретации гидродинамических исследований скважин методом двух режимов по моделям двойной пористости и двойной проницаемости. Приведены соответствующие математические модели. Получены две приближенные формулы для связи специальных параметров модели двойной проницаемости. Предложена приближенная формула для связи общего скин-фактора и скин-факторов пропластков для модели двойной проницаемости. Рассматриваемые модели успешно апробированы на синтетических кривых забойного давления.

Ключевые слова: гидродинамические исследования скважин, метод двух режимов, модель двойной пористости, модель двойной проницаемости.

Введение

Гидродинамические исследования скважин являются одним из основных источников о фильтрационно-емкостных свойствах пластов, а так же важным способом контроля разработки месторождений.

Большинство методов гидродинамических исследований требует остановки скважин, что приводит к потерям в добыче нефти, поэтому промысловые службы стараются уменьшить объем таких работ.

Актуальным является развитие методов исследований без остановки скважин.

Одним из таких методов является рассматриваемый в настоящей работе метод двух режимов.

Он представляет из себя последовательную отработку скважины на двух режимах работы (при двух различных дебитах), причем второй дебит должен быть больше первого.

Для определения искомых параметров анализируется кривая забойного давления, зарегистрированная на втором режиме работы скважины, с учетом истории работы на первом режиме.

Поскольку более сложными с точки зрения разработки являются неоднородные по фильтрационно-емкостным свойствам (ФЕС) пласты, то методы определения степени их неоднородности являются актуальными.

Поэтому настоящая работа посвящена моделям двойной пористости и двойной проницаемости, как моделям, описывающим неоднородные по ФЕС пласты.

1. Модель двойной пористости

Модель двойной пористости описывает фильтрацию в сложной неоднородной среде, представленной блоками низкопроницаемой породы (матрицы), рассеченными высокопроницаемыми трещинами, рис. 1. С помощью этой модели описывают трещиновато-поровый коллектор.

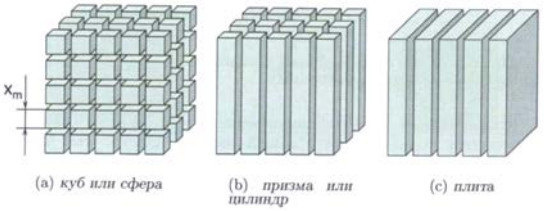


Рис. 1. Идеализированное геометрическое представление трещиноватых коллекторов при описании процессов фильтрации [3]

Существует две распространенные модели двойной пористости:

1. Модель псевдоустановившихся межпоровых перетоков [Warren и Root], пренебрегающая распределением давления внутри блоков матрицы.
2. Модель неустановившихся межпоровых перетоков [Swann], учитывающая распределение давления внутри блоков матрицы.

На первый взгляд кажется, что модель неустановившихся межпоровых перетоков

должна точнее отписывать реальные пласты. Однако практика показывает, что в большинстве случаев работу реальных трещиновато-поровых коллекторов лучше описывает модель псевдоустановившихся межпоровых перетоков. Поэтому ниже будет рассматриваться именно эта модель.

1. Ограничения и предположения, принимаемые в рассматриваемой модели:
2. Фильтрация однофазная. Флюид слабосжимаемый. Пласт упругий.
3. Проницаемость трещин много больше проницаемости матрицы.
4. Пористость матрицы много больше проницаемости трещин.
5. Притоком флюида из матрицы непосредственно в ствол скважины пренебрегается.
6. Переток флюида между матрицей и трещинами псевдоустановившийся.
7. Блоки матрицы однородные и одинаковы по форме и свойствам.
8. Трещины одинаковы по форме и свойствам.

Модель двойной пористости описывается двумя специфическими параметрами [1-4, 7, 8]:

1. «омега» – относительная емкость трещин

$$\omega = \frac{\left(h\phi c_t\right)_f}{\left(h\phi c_t\right)_f + \left(h\phi c_t\right)_m}, \qquad (1)$$

где h - толщина, м; ϕ - пористость, д.ед.; c_t - суммарная сжимаемость системы пласт-флюид, 1/бар; нижний индекс «f» - трещины, нижний индекс «m» - матрица и

2. «лямбда» – параметр, характеризующий переток жидкости между матрицей и трещинами

$$\lambda = \alpha r_w^2 \frac{k_m}{k_f}, \qquad (2)$$

где k - проницаемость, мД; r_w - радиус скважины, м; $\alpha = 4n(n+2)/x_m^2$ - параметр формы матричных блоков, 1/м^2; n=3 (куб или сфера), n=2 (призма или цилиндр), n=1 (плита), рис. 1; x_m - характерный размер блока матрицы, м.

Будем рассматривать интерпретацию исследования скважины методом двух режимов по модели двойной пористости на примере теоретической кривой, полученной с помощью моделирования в ПК Saphir Kappa Engineering [4], рис. 2. Исходные данные для моделирования: радиус скважины 0,1 м; толщина пласта 10 м.; общая пористость 0,12 м.; объемный коэффициент 1,0 м^3/м^3; вязкость флюида 1,0 мПа·с; общая сжимаемость 4,3·10^{-5} 1/бар; влияние ствола скважины отсутствует; общий скин-фактор +5,0 безразм.; начальное пластовое давление 350 бар; проницаемость трещин 100 мД; омега 0,1 д.ед.; лямбда 5,0·10^{-8} безразм.

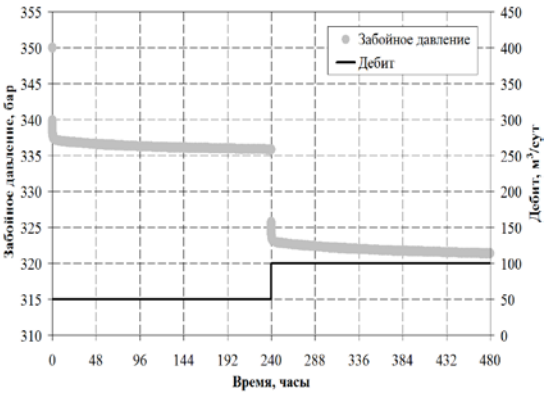


Рис. 2. Технологическая схема исследования методом двух режимов пласта с двойной пористостью

Для определения режима притока используется диагностический график Бурде [1, 3, 4, 6, 7] - семейство двух кривых: изменение забойного давления и его производная по функции суперпозиции, построенные от времени в часах. Производная вычисляется по формуле Бурде:

$$\frac{d\left(\Delta P_i\right)}{dx_i} = \frac{1}{x_{i+1}-x_{i-1}}\left[\frac{\Delta P_i - \Delta P_{i-1}}{x_i - x_{i-1}}\left(x_{i+1}-x_i\right) + \frac{\Delta P_{i+1}-\Delta P_i}{x_{i+1}-x_i}\left(x_i - x_{i-1}\right)\right] \qquad (3)$$

В качестве функции суперпозиции для исследования на двух режимах выступает:

$$x = lg\left(\frac{t_1 + \Delta t}{\Delta t}\right) + \frac{q_2}{q_1} lg\left(\Delta t\right), \qquad (4)$$

где t_1 - длительность первого режима, час; Δt - время с начала второго режима, час; q_1 и q_2 - дебиты флюида в поверхностных условиях на первом и втором режимах работы скважины соответственно, м^3/сут.

Диагностический график, построенный по изображенной на рис. 2 кривой давления, представлен на рис. 3.

На диагностическом графике, рис. 3, отсутствует влияние ствола скважины. Первый режим притока, характеризующийся

горизонтальным участком производной 0,01-0,4 часа, радиальный приток флюида к скважине по трещинам. Уравнение притока имеет вид:

$$P_{wf} = m_1^{/}\left[lg\left(\frac{t_1+\Delta t}{\Delta t}\right) + \frac{q_2}{q_1} lg\left(\Delta t\right)\right] + P_{int,1}, \quad (5)$$

где P_{wf} - забойное давление, бар; $m_1^{/}$ и $P_{int,1}$ - коэффициенты.

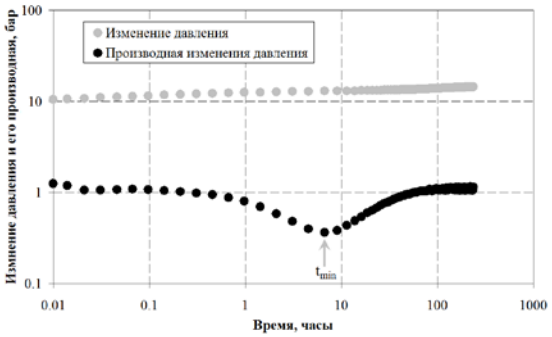


Рис. 3. Диагностический график. Модель двойной пористости

На практике этот режим часто бывает скрыт влиянием ствола скважины.

Определяется проницаемость трещин k_f, мД и скин-фактор трещин S_f, безразм.:

$$k_f = \frac{21{,}5 q_1 B\mu}{m_1^{/} h},\ P_{1hr,1} = m_1^{/} lg\left(t_1+1\right) + P_{int,1},$$

$$S_f = 1{,}1513\left[\frac{q_1}{q_1-q_2}\cdot\frac{P_{wf}\left(\Delta t=0\right)-P_{1hr,1}}{m_1^{/}} - \right.$$
$$\left. - lg\left(\frac{k_f}{\mu\left(\phi C_t\right)_f r_w^2}\right) + 3{,}0923\right] \quad (6)$$

где B - объемный коэффициент, м^3/м^3; μ - вязкость флюида, мПа·с; h - толщина пласта, м; $P_{1hr,1}$ - забойное давление через один час после перехода на второй режим; $P_{wf}\left(\Delta t=0\right)$ - первая точка забойного давления второго режима работы скважины (последняя точка первого режима); ϕ - пористость, д.ед.; C_t - сжимаемость системы пласт-флюид, 1/бар.

Второй режим притока характеризуется падением производной с ее последующим ростом, рис. 3., 0,4-80,0 часа, переходная зона, перераспределение давления между матрицей и трещинами. Определяется комплексный параметр – $t_{D\,min}$ безразмерное время при минимальном значении производной:

$$t_{D\,min} = 0{,}00036\frac{k_f t_{min}}{\mu\left(\phi C_t\right)_t r_w^2}, \quad (7)$$

где нижний индекс «*t*» - суммарное значение для матрицы и трещин; t_{min} - время с начала второго режима в момент минимального значения производной, час, рис. 3.

Третий режим притока, характеризующийся горизонтальным участком производной после 80 часов, радиальный приток флюида к скважине из системы трещины + матрица. Уравнение притока имеет вид:

$$P_{wf} = m_2^{/}\left[lg\left(\frac{t_1+\Delta t}{\Delta t}\right) + \frac{q_2}{q_1} lg\left(\Delta t\right)\right] + P_{int,2}, \quad (8)$$

где $m_1^{/} \approx m_2^{/}$.

Определяется проницаемость трещин k_f, мД, пластовое давление P_i, бар и общий скин-фактор скважины S, безразм.:

$$k_f = \frac{21{,}5 q_1 B\mu}{m_2^{/} h},\ P_{1hr,2} = m_2^{/} lg\left(t_1+1\right) + P_{int,2},$$

$$P_i = P_{int,2} - \frac{q_2}{q_1-q_2}\left[P_{wf}\left(\Delta t=0\right) - P_{1hr,2}\right],$$

$$S = 1{,}1513\left[\frac{q_1}{q_1-q_2}\cdot\frac{P_{wf}\left(\Delta t=0\right)-P_{1hr,2}}{m_2^{/}} - \right.$$
$$\left. - lg\left(\frac{k_f}{\mu\left(\phi C_t\right)_t r_w^2}\right) + 3{,}0923\right] \quad (9)$$

Специфические параметры модели двойной пористости определяются следующим образом:

$$\omega = 10^{-\Delta P / m_2^{/}},\ \lambda = \frac{\omega}{t_{D\,min}} ln\left(\frac{1}{\omega}\right), \quad (10)$$

где ΔP - разница давлений между двумя параллельными линиями в координатах (забойное давление; функция суперпозиции), аппроксимирующими два радиальных притока (по трещинам и по системе матрица + трещины), бар.

Скин-факторы S и S_f связаны соотношением [7]:

$$S_f = S + 0{,}5 ln\left(1/\omega\right). \quad (11)$$

Обработка кривой давления, приведенной на рис. 2, описанным способом приведена на рис 4. В результате обработки определены следующие параметры: общий скин-фактор +5,003 безразм.; начальное пластовое давление 349,999 бар; проницаемость трещин 98,2 мД;

омега 0,101 д.ед.; лямбда $5{,}06\cdot10^{-8}$ безразм. Получено хорошее совпадение с заданными при моделировании параметрами.

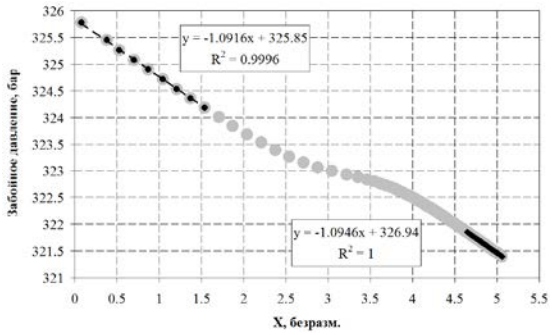


Рис. 4. Обработка исследования методом двух режимов по модели пласта с двойной пористостью в полулогарифмических координатах

2. Модель двойной проницаемости

Модель двойной проницаемости описывает фильтрацию в слоисто-неоднородной среде, состоящей из двух пропластков с резко различной проницаемостью и пористостью с перетоком между ними. Пропластки могут разделяться тонкой сверхнизкопроницаемой (глинистой) перемычкой, по которой отсутствует поток флюида по простиранию, но возможен вертикальный переток через перемычку.

Модель двойной проницаемости описывается тремя специфическими параметрами [1, 3, 4, 7, 8]:

1. «омега» – относительная емкость высокопроницаемого пропластка

$$\omega = \frac{(h\phi c_t)_1}{(h\phi c_t)_1 + (h\phi c_t)_2}, \qquad (12)$$

где нижний индекс «1» - более проницаемый пропласток, нижний индекс «2» - менее проницаемый пропласток;

2. «каппа» - относительная проводимость высокопроницаемого пропластка

$$\kappa = \frac{(kh)_1}{(kh)_1 + (kh)_2}, \qquad (13)$$

3. «лямбда» – параметр, характеризующий переток жидкости между высокопроницаемым и низкопроницаемым пропластками

$$\lambda = \frac{T_e r_w^2}{(kh)_1 + (kh)_2}, \qquad (14)$$

где

$$T_e = \frac{2}{2\dfrac{h_e}{k_{ze}} + \dfrac{h_1}{k_{z1}} + \dfrac{h_2}{k_{z2}}}, \qquad (15)$$

h_e - толщина глинистой перемычки, м; k_{ze} - вертикальная проницаемость глинистой перемычки, мД; k_{z1} и k_{z2} - вертикальная проницаемость первого и второго пропластков, мД.

Если сверхнизкопроницаемой (глинистая) перемычка отсутствует $h_e = 0$ и вертикальная проницаемость первого и второго пропластков близки $k_{z1} = k_{z2} = k_z$, то при известном параметре перетока λ можно оценить анизотропию проницаемости пласта [1, 8]:

$$\frac{k_z}{k} = \frac{1}{2}\frac{h^2}{r_w^2}\lambda. \qquad (16)$$

Если $h_e / k_{ze} >> h_1 / k_{z1} + h_2 / k_{z2}$ (что возможно, например, когда h_e с h_1 и h_2 различаются на порядок, т.е. толщина пропластков – метры, а глинистой перемычки – сантиметры, при $k_{ze} << k_{z1}$ и $k_{ze} << k_{z2}$, что вполне логично), то можно оценить вертикальную проницаемость глинистой перемычки [1, 8]:

$$k_{ze} = \frac{(kh)_1 + (kh)_2}{r_w^2} h_e \lambda. \qquad (17)$$

Если сверхнизкопроницаемой (глинистая) перемычка отсутствует $h_e = 0$, а вертикальная проницаемость пропластков выражена через проницаемость по простиранию и коэффициент анизотропии α как $k_{z1} = \alpha_1 k_1$ и $k_{z2} = \alpha_2 k_2$, то преобразуя T_e к виду:

$$T_e = \frac{2/(\phi C_t h)^2}{\dfrac{(\phi C_t h)_1^2}{(\phi C_t h)^2}\dfrac{1}{\alpha_1 (kh)_1 (\phi C_t)_1^2} + \dfrac{(\phi C_t h)_2^2}{(\phi C_t h)^2}\dfrac{1}{\alpha_2 (kh)_2 (\phi C_t)_2^2}} \qquad (18)$$

и учитывая

$$\omega = \frac{(\phi C_t h)_1}{(\phi C_t h)}, \; 1-\omega = \frac{(\phi C_t h)_2}{(\phi C_t h)},$$
$$\phi C_t h = (\phi C_t h)_1 + (\phi C_t h)_2 \qquad (19)$$

можно получить следующую формулу для связи специальных параметров ω, λ и κ:

$$\lambda = \frac{2r_w^2}{(\phi C_t h)^2\left[\frac{\omega^2}{\alpha_1(\phi C_t)_1^2}\frac{1}{\kappa} + \frac{(1-\omega)^2}{\alpha_2(\phi C_t)_2^2}\frac{1}{1-\kappa}\right]}. \quad (20)$$

Если кроме условий $h_e = 0$, $k_{z1} = \alpha_1 k_1$ и $k_{z2} = \alpha_2 k_2$ предположить $(\phi C_t)_1 = (\phi C_t)_2$, что может быть оправдано для терригенного коллектора, то:

$$T_e = \frac{2/(h_1+h_2)^2}{\frac{h_1^2}{(h_1+h_2)^2}\frac{1}{\alpha_1(kh)_1} + \frac{h_2^2}{(h_1+h_2)^2}\frac{1}{\alpha_2(kh)_2}} \quad (21)$$

и учитывая

$$\omega = \frac{h_1}{h},\ 1-\omega = \frac{h_2}{h},\ h = h_1 + h_2 \quad (22)$$

можно получить следующую формулу для связи специальных параметров ω, λ и κ:

$$\lambda = \frac{2r_w^2}{h^2\left[\frac{\omega^2}{\alpha_1}\frac{1}{\kappa} + \frac{(1-\omega)^2}{\alpha_2}\frac{1}{1-\kappa}\right]}. \quad (23)$$

Характерный вид диагностического графика для модели двойной проницаемости, как и для модели двойной пористости, включает в себя два режима радиального притока и переходный режим между ними. Более подробно диагностический график будет рассмотрен ниже. Пользуясь уравнением неустановившегося притока жидкости к скважине на позднем радиальном режиме фильтрации (когда видна работа обоих пропластков) и пренебрегая историей работы скважины можно получить приближенное соотношение, связывающее суммарные фильтрационно-емкостные свойства пласта и специальные параметры ω, κ с общим скин-фактором скважины S и скин-факторами пропластков S_1 и S_2:

$$\frac{1}{lg(\Delta t) + lg\left[\frac{(kh)_t}{(\phi C_t h)_t \mu r_w^2}\right] - 3{,}23 + 0{,}87S} =$$

$$= \frac{\kappa}{lg(\Delta t) + lg\left[\frac{\kappa}{\omega}\frac{(kh)_t}{(\phi C_t h)_t \mu r_w^2}\right] - 3{,}23 + 0{,}87S_1} +$$

$$+ \frac{1-\kappa}{lg(\Delta t) + lg\left[\frac{1-\kappa}{1-\omega}\frac{(kh)_t}{(\phi C_t h)_t \mu r_w^2}\right] - 3{,}23 + 0{,}87S_2} \quad (24)$$

где $(kh)_t = (kh)_1 + (kh)_2$ и $(\phi C_t h)_t = (\phi C_t h)_1 + (\phi C_t h)_2$; $\Delta t \geq t^{**}$, а t^{**} - время выхода скважины на второй радиальный режим фильтрации, которое в первом приближении можно оценить, как время выхода на радиальный режим фильтрации менее проницаемого пропластка [5]:

$$t^{**} = \frac{22105C}{(kh)_2/\mu} exp(0{,}14S_2), \quad (25)$$

где C - емкость ствола скважины, м^3/бар.

Для случая, когда в скважину работают оба пласта и $S_1 \neq S_2$ однозначно определить S_1 и S_2 невозможно [1, 7]. Скин-фактор второго (низкопроницаемого) пласта мало влияет на вид диагностического графика.

Будем рассматривать два варианта модели двойной проницаемости:

1. В скважину работают оба пропластка.
2. В скважину работает только низкопроницаемый (второй) пропласток. Призабойная зона высокопроницаемого (первого) пропластка забита (имеется высокий положительный скин-фактор) и добыча из него осуществляется через низкопроницаемый пропласток за счет перетока между ними.

<u>Рассмотрим вариант 1. В скважину работают оба пропластка.</u>

Интерпретацию исследования скважины методом двух режимов по модели двойной проницаемости будем рассматривать на примере теоретической кривой, полученной с помощью моделирования в ПК Saphir Kappa Engineering [4], рис. 5. Исходные данные для моделирования: радиус скважины 0,1 м; толщина пласта 10 м.; общая пористость 0,12 м.; объемный коэффициент 1,0 м^3/м^3; вязкость флюида 1,0 мПа·с; общая сжимаемость 4,3·10^{-5} 1/бар; влияние ствола скважины отсутствует; общий скин-фактор +0,019 безразм. (обусловлен перетоками между пропластками); скин-факторы первого и второго пропластков одинаковы и равны 0,0 безразм; начальное пластовое давление 250 бар; средневзвешенная проницаемость 80 мД; омега 0,05 д.ед.; лямбда 1,0·10^{-7} безразм.; каппа 0,9 д.ед.

Для определения режима притока используется диагностический график Бурде, описанной выше в пункте 1 статьи. В качестве функции суперпозиции для исследования на двух режимах выступает (4).

Диагностический график, построенный по изображенной на рис. 5 кривой давления, представлен на рис. 6.

На диагностическом графике, рис. 6, отсутствует влияние ствола скважины. Первый режим притока, характеризующийся горизонтальным участком производной 0,001-0,2 часа, радиальный приток флюида к скважине по двум пропласткам без перетоков, т.к. разность давлений между пропластками отсутствует, пластовые давления в пропластках равны. Уравнение притока имеет вид:

$$P_{wf} = m_1^{/}\left[lg\left(\frac{t_1 + \Delta t}{\Delta t} \right) + \frac{q_2}{q_1} lg(\Delta t) \right] + P_{int,1}, \quad (26)$$

где P_{wf} - забойное давление, бар; $m_1^{/}$ и $P_{int,1}$ - коэффициенты.

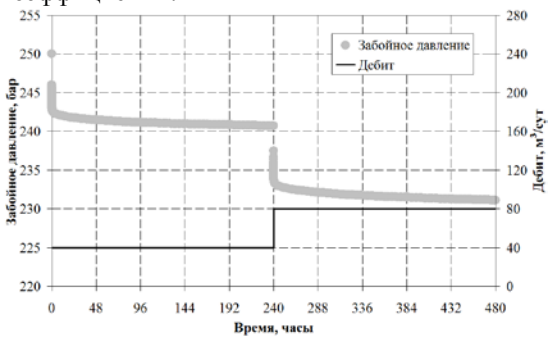


Рис. 5. Технологическая схема исследования методом двух режимов пласта с двойной проницаемостью. В скважину работают оба пропластка

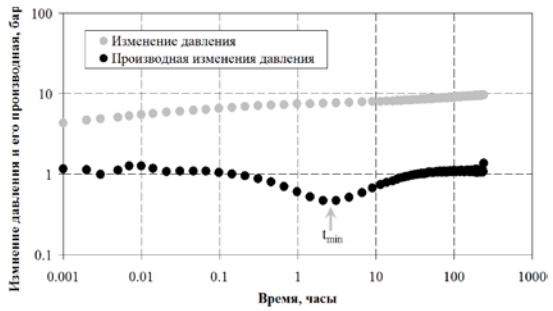


Рис. 6. Диагностический график. Модель двойной проницаемости. В скважину работают оба пропластка

На практике этот режим часто бывает скрыт влиянием ствола скважины.

Определяется средневзвешенная проницаемость системы:

$$k = \frac{21{,}5 q_1 B\mu}{m_1^{/}(h_1 + h_2)} = \frac{21{,}5 q_1 B\mu}{m_1^{/} h}. \quad (27)$$

Второй режим притока характеризуется падением производной с ее последующим ростом, рис. 6., 0,2-75,0 часа, переходная зона, возникновение перепада давления между высокопроницаемым и низкопроницаемым пропластками, перетоки жидкости между пропластками.

К концу режима давления в пропластках выравниваются и перепад давления между ними исчезает. Определяется комплексный параметр – $t_{D\,min}$ безразмерное время при минимальном значении производной:

$$t_{D\,min} = 0{,}00036 \frac{k t_{min}}{\mu(\phi C_t)_t r_w^2}, \quad (28)$$

где нижний индекс «*t*» - суммарное значение для двух пропластков; t_{min} - время с начала второго режима в момент минимального значения производной, час, рис. 6.

Третий режим притока, характеризующийся горизонтальным участком производной после 75 часов, радиальный приток флюида к скважине из двух пропластков с равным пластовым давлением. Перетоков нет. Уравнение притока имеет вид:

$$P_{wf} = m_2^{/}\left[lg\left(\frac{t_1 + \Delta t}{\Delta t} \right) + \frac{q_2}{q_1} lg(\Delta t) \right] + P_{int,2}, \quad (29)$$

где $m_1^{/} = m_2^{/}$.

Определяется средневзвешенная проницаемость системы k, мД, пластовое давление P_i, бар и общий скин-фактор скважины S, безразм.:

$$k = \frac{21{,}5 q_1 B\mu}{m_2^{/}(h_1 + h_2)} = \frac{21{,}5 q_1 B\mu}{m_2^{/} h},$$

$$P_{1hr,2} = m_2^{/} lg(t_1 + 1) + P_{int,2},$$

$$P_i = P_{int,2} - \frac{q_2}{q_1 - q_2}\left[P_{wf}(\Delta t = 0) - P_{1hr,2} \right],$$

$$S = 1{,}1513\left[\frac{q_1}{q_1 - q_2} \cdot \frac{P_{wf}(\Delta t = 0) - P_{1hr,2}}{m_2^{/}} - \right. \quad (30)$$

$$\left. - lg\left(\frac{k}{\mu(\phi C_t)_t r_w^2} \right) + 3{,}0923 \right]$$

Специфические параметры «омега» и «лямбда» модели двойной проницаемости определяются аналогично модели двойной пористости:

$$\omega = 10^{-\Delta P / m_2^{/}}, \ \lambda = \frac{\omega}{t_{D\,min}} ln\left(\frac{1}{\omega} \right), \quad (31)$$

где ΔP - разница давлений между двумя параллельными линиями в координатах (забойное давление; функция суперпозиции), аппроксимирующими два радиальных притока, бар.

Параметр «каппа» определяется из (20) или (23) в соответствии с принятыми предположениями.

Проводимости пропластков определяются из (13) и (30) следующим образом:

$$(kh)_1 = \kappa kh, \quad (32)$$

$$(kh)_2 = (kh)_1 / \kappa - (kh)_1 \text{ либо}$$

$$(kh)_2 = \frac{21{,}5 q_1 B\mu}{m_2'} - (kh)_1. \quad (33)$$

Принимая $S_1 = S_2$ по (24) оценивается скин-фактор пропластков.

Обработка кривой давления, приведенной на рис. 5, описанным способом приведена на рис 7. В результате обработки определены следующие параметры: общий скин-фактор +0,044 безразм.; начальное пластовое давление 250,005 бар; средневзвешанная проницаемость 78,9 мД; омега 0,137 д.ед.; лямбда $1{,}71 \cdot 10^{-7}$ безразм., каппа 0,993 д.ед. Получено удовлетворительное совпадение с заданными при моделировании параметрами.

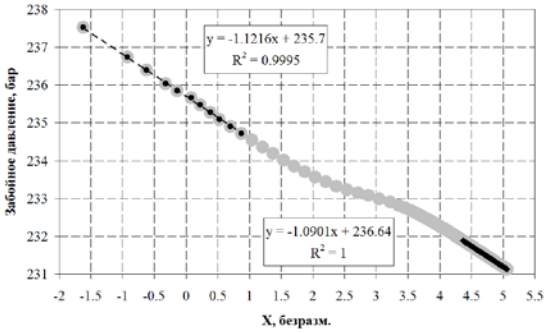


Рис. 7. Обработка исследования методом двух режимов по модели пласта с двойной проницаемостью в полулогарифмических координатах. В скважину работают оба пропластка

<u>Рассмотрим вариант 2. В скважину работает только низкопроницаемый (второй) пропласток.</u> Интерпретацию исследования скважины методом двух режимов по модели двойной проницаемости будем рассматривать на примере теоретической кривой, полученной с помощью моделирования в ПК Saphir Kappa Engineering [4], рис. 8. Исходные данные для моделирования: радиус скважины 0,1 м; толщина пласта 10 м.; общая пористость 0,12 м.; объемный коэффициент 1,0 $м^3/м^3$; вязкость флюида 1,0 мПа·с; общая сжимаемость $4{,}3 \cdot 10^{-5}$ 1/бар; влияние ствола скважины отсутствует; общий скин-фактор +18 безразм.; скин-факторы первого (высокопроницаемого) пропластка +5000 безразм. (т.е. в скважину он не работает); скин-факторы второго (низкопроницаемого) пропластка 0,0 безразм; начальное пластовое давление 300 бар; средневзвешенная проницаемость 50 мД; омега 0,05 д.ед.; лямбда $1{,}0 \cdot 10^{-6}$ безразм.; каппа 0,75 д.ед.

Для определения режима притока используется диагностический график Бурде, описанной выше в пункте 1 статьи. В качестве функции суперпозиции для исследования на двух режимах выступает (4).

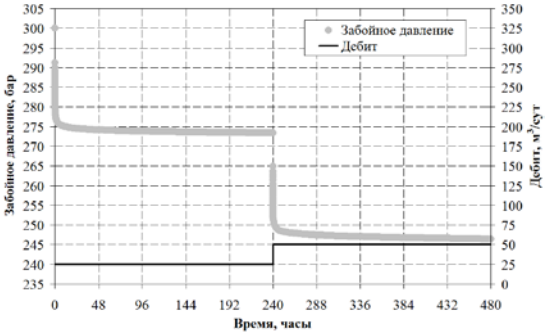


Рис. 8. Технологическая схема исследования методом двух режимов пласта с двойной проницаемостью. В скважину работает низкопроницаемый пропласток

Диагностический график, построенный по изображенной на рис. 8 кривой давления, представлен на рис. 9.

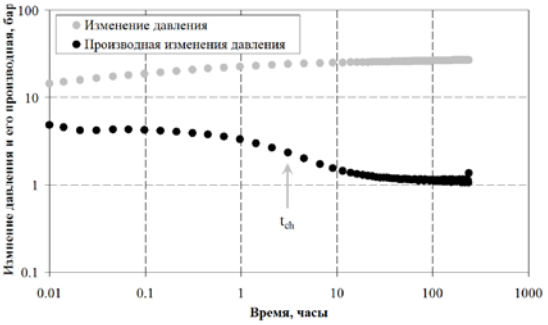


Рис. 9. Диагностический график. Модель двойной проницаемости. В скважину работает низкопроницаемый пропласток

На диагностическом графике, рис. 9, отсутствует влияние ствола скважины. Первый режим притока, характеризующийся горизонтальным участком производной 0,001-0,5 часа, радиальный приток флюида к скважине по второму (низкопроницаемому) пропластку. Уравнение притока имеет вид:

$$P_{wf} = m_1' \left[lg\left(\frac{t_1 + \Delta t}{\Delta t} \right) + \frac{q_2}{q_1} lg(\Delta t) \right] + P_{int,1}, \quad (34)$$

где P_{wf} - забойное давление, бар; m_1' и $P_{int,1}$ - коэффициенты.

На практике этот режим часто бывает скрыт влиянием ствола скважины.

Определяется проводимость второго пропластка $(kh)_2$ и его скин-фактор S_2:

$$(kh)_2 = \frac{21{,}5 q_1 B\mu}{m_1'},$$

$$P_{1hr,1} = m_1' \, lg(t_1 + 1) + P_{int,1},$$

$$(\phi h C_t)_2 = (1-\omega)(\phi h C_t)_t,$$

$$S_2 = 1{,}1513\left[\frac{q_1}{q_1 - q_2}\cdot\frac{P_{wf}(\Delta t = 0) - P_{1hr,1}}{m_1'} - lg\left(\frac{(kh)_2}{\mu(\phi h C_t)_2 r_w^2}\right) + 3{,}0923\right] \quad (35)$$

Второй режим притока характеризуется падением производной, рис. 9., 0,5-60,0 часа, переходная зона, возникновение перепада давления между высокопроницаемым и низкопроницаемым пропластками, перетоки жидкости между пропластками. К концу режима давления в пропластках выравниваются и перепад давления между ними исчезает. Определяется комплексный параметр – t_{Dch} безразмерное время при перегибе производной:

$$t_{Dch} = 0{,}00036\frac{k t_{ch}}{\mu(\phi C_t)_t r_w^2}, \quad (36)$$

где нижний индекс «*t*» - суммарное значение для обоих пропластков; t_{ch} - время с начала второго режима в момент перегиба производной, час, рис. 9.

Третий режим притока, характеризующийся горизонтальным участком производной после 60 часов, радиальный приток флюида к скважине из двух пропластков с равным пластовым давлением. Перетоков нет. Уравнение притока имеет вид:

$$P_{wf} = m_2'\left[lg\left(\frac{t_1 + \Delta t}{\Delta t}\right) + \frac{q_2}{q_1} lg(\Delta t)\right] + P_{int,2}, \quad (37)$$

где в отличие от рассметренных ранее вариантов $m_1' \neq m_2'$.

Определяется средневзвешенная проницаемость системы k, мД; пластовое давление P_i; бар и общий скин-фактор скважины S, безразм.:

$$k = \frac{21{,}5 q_1 B\mu}{m_2'(h_1 + h_2)} = \frac{21{,}5 q_1 B\mu}{m_2' h}, \; (kh)_1 + (kh)_2 = \frac{21{,}5 q_1 B\mu}{m_2'},$$

$$P_{1hr,2} = m_2' \, lg(t_1 + 1) + P_{int,2},$$

$$P_i = P_{int,2} - \frac{q_2}{q_1 - q_2}\left[P_{wf}(\Delta t = 0) - P_{1hr,2}\right],$$

$$S = 1{,}1513\left[\frac{q_1}{q_1 - q_2}\cdot\frac{P_{wf}(\Delta t = 0) - P_{1hr,2}}{m_2'} - lg\left(\frac{k}{\mu(\phi C_t)_t r_w^2}\right) + 3{,}0923\right] \quad (38)$$

Специфические параметры «каппа» и «лямбда» модели двойной проницаемости определяются как:

$$\kappa = 1 - m_2'/m_1', \; \lambda = \frac{\omega}{t_{Dch}} ln\left(\frac{1}{\omega}\right). \quad (39)$$

Параметр «омега» определяется из (20) или (23) в соответствии с принятыми предположениями.

Проводимости первого пропластка определяется из (13) и (30) следующим образом:

$$(kh)_1 = (kh)_2 \frac{\kappa}{1-\kappa} \text{ либо}$$

$$(kh)_1 = \frac{21{,}5 q_1 B\mu}{m_2'} - (kh)_2. \quad (40)$$

С учетом формулы (35) для S_2 и формулы (38) для S по формуле (24) оценивается скин-фактор первого пропластка S_1.

Обработка кривой давления, приведенной на рис. 8, описанным способом приведена на рис 10. В результате обработки определены следующие параметры: общий скин-фактор +17,9 безразм.; скин-фактор первого (высокопроницаемого) пропластка +3370 безразм.; начальное пластовое давление 300,003 бар; средневзвешанная проницаемость 48,1 мД; омега 0,04 д.ед.; лямбда $9{,}81\cdot10^{-7}$ безразм., каппа 0,754 д.ед. Получено хорошее совпадение с заданными при моделировании параметрами.

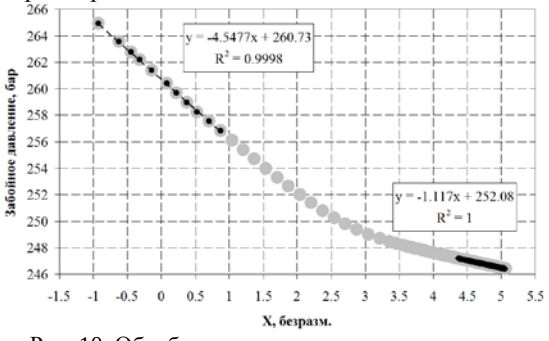


Рис. 10. Обработка исследования методом двух режимов по модели пласта с двойной проницаемостью в полулогарифмических координатах. В скважину работает низкопроницаемый пропласток

Заключение

Рассмотрены математические модели двойной пористости и двойной проницаемости для интерпретации гидродинамических исследований скважин методом двух режимов. Модели испытаны на синтетических кривых забойного давления в работающих скважинах.

Предложены формулы для связи специфических параметров модели двойной проницаемости – «каппа», «омега» и «лямбда».

Пользуясь уравнением неустановившегося притока жидкости к скважине на позднем радиальном режиме фильтрации (когда видна работа обоих пропластков) для модели двойной проницаемости и пренебрегая историей работы скважины, предложено приближенное соотношение, связывающее суммарные фильтрационно-емкостные свойства пласта и специальные параметры «омега» и «каппа» с общим скин-фактором скважины и скин-факторами пропластков.

Предлагаемый подход позволяет проводить исследования без остановки скважин, что уменьшает потери нефти, удешевляет исследования и потенциально способствует увеличению охвата пласта исследованиями.

Работа выполнена при поддержке Программы фундаментальных научных исследований государственных академий наук № I.2.П27, заказ № 18-0111_П, НИР № 0065-2018-0111.

Dual Permeability and Dual Porosity Reservoir Models for Two Rate Well Test Results Interpretation

I.V. Afanaskin, S.G. Volpin, Ju.M. Shtejnberg

Abstract: Article overviews well testing interpretation results features of dual flow regime test for dual permeability and dual porosity reservoir models. Corresponding mathematic models description included. For special parameters relation of dual permeability model two approximate formulas were derived. Full skin and separate layers skin relation approximate formula for dual permeability model also derived. Described models applicability were successfully checked on synthetic flowing bottom hole pressure curves.

Key words: well test, two-rate test, double porosity model, double permeability model.

Литература

1. Д.Бурде. Интерпретация результатов исследований скважин // Материалы лекций. Petroleum Engineering and Related Management Traning Gubkin Academy. М., 1994, 109 с.
2. Т.Д.Гольф-Рахт. Основы нефтепромысловой геологии и разработки трещиноватых коллекторов. М., Недра, 1986, 608 с.
3. Т.А.Деева, М.Р.Камартдинов, Т.Е.Кулагина и др. Гидродинамические исследования скважин: анализ и интерпретация данных. Томск, ЦППС НД ТПУ, 2009, 243 с.
4. Оливье Узе, Дидье Витура, Оле Фьяре. Анализ динамических потоков. Теория и практика интерпретации данных ГДИС и анализа добычи, а так же использование данных стационарных глубинных манометров. Kappa Engineering, 2009, 359 с.
5. Р.Эрлогер. мл. Гидродинамические методы исследования скважин. М.-Ижевск, Институт компьютерных исследований, 2007, 512 с.
6. D.Bourdet et al. A new set of type curves simplifies well test analysis // World Oil, 1983, May, P. 95–106.
7. D.Bourdet. Well test analysis: The use of advanced interpretation models. 2002, 436 p.
8. PanSystem 2. Версия 2.3. Edinburg Petroleum Services LTD, FEB, 1996.

Об оценке фильтрационной значимости нарушений, выделенных по геофизическим данным

С.Г. Вольпин[1], И.В. Афанаскин[2], В.А. Юдин[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]sergvolpin@gmail.com, [2]ivan@afanaskin.ru,[3]yudinval@yandex.ru

Аннотация: Настоящая статья посвящена детализированному описанию авторского комплексного подхода для оценки фильтрационной значимости макро- и меганарушений, выделенных по данным геофизических исследований. Проанализирована возможность применения гидродинамических методов исследования скважин и пластов для определения фильтрационной структуры и параметров дизъюнктивных нарушений на нефтяных залежах по данным: гидропрослушивания, применения трассерного метода и ГДИС.

Ключевые слова: геофизические исследования скважин (ГИС), гидродинамические исследования скважин (ГДИС), кривая восстановления давления (КВД), гидропрослушивание, трассерные исследования

Введение

При проектировании разработки нефтяных месторождений крайне существенной является информация о зонах с аномальными фильтрационными свойствами, возникающих в окрестности тектонических нарушений, которые последние годы часто выделяются в осадочных разрезах Западно-Сибирской платформы.

Наличие таких нарушений фиксируется даже в таких объектах, как баженовская свита, для которой на многих участках её распространения характерно наличие аномально-высокого пластового давления. [1 – 7, 10, 11].

Наличие аномально высокого пластового давления в залежах баженовской свиты при пластовых давлениях в выше- и нижележащих отложениях, равных гидростатическому вызывает сомнение в существовании проводящих нарушений. Тем не менее, опубликовано огромное число работ, в которых авторы сообщают о локализации различного рода нарушений.

В частности, на основе региональных и сейсмических данных, в работе [7] были выделены многочисленные разрывные нарушения северо-западного и северо-восточного направлений. По результатам комплексного анализа космогеологических, геоморфологических, магнитометрических, гравиметрических и сейсмических данных в работе [5] сделан вывод, что в баженовской свите системы флексурно-разломных дислокаций различных ранга и амплитуды контролируют распространение трещинных и трещинно-кавернозных коллекторов в отложениях баженовской свиты.

В работах [2 - 4] по данным сейсмических, промыслово-геофизических, космических исследований и результатов разработки утверждается, что в продуктивных отложениях свиты вообще проявляется блочная структура.

Пример модели Пальяновского месторождения с наличием вертикальных тектонических нарушений приведён на рис. 1 [6].

Показанная на рис. 1 тутлеймская свита является аналогом баженовской.

В последнее время в связи с развитием методик и программ обработки сейсмических данных во всё большем масштабе на различных месторождениях, в том числе и с коллекторами в баженовской свите, выявляются малоамплитудные нарушения.

Для примера можно привести исследования последних лет [1], в которых на Средне-Назымском месторождении практически на всех временных разрезах были трассированы аномалии сейсмической записи, которые интерпретировались как малоамплитудные дизъюнктивные нарушения с вертикальной амплитудой менее 5 метров.

Полученная структурная карта показана на рис. 2, на котором изолинии показаны без смещений ввиду их крайней малости.

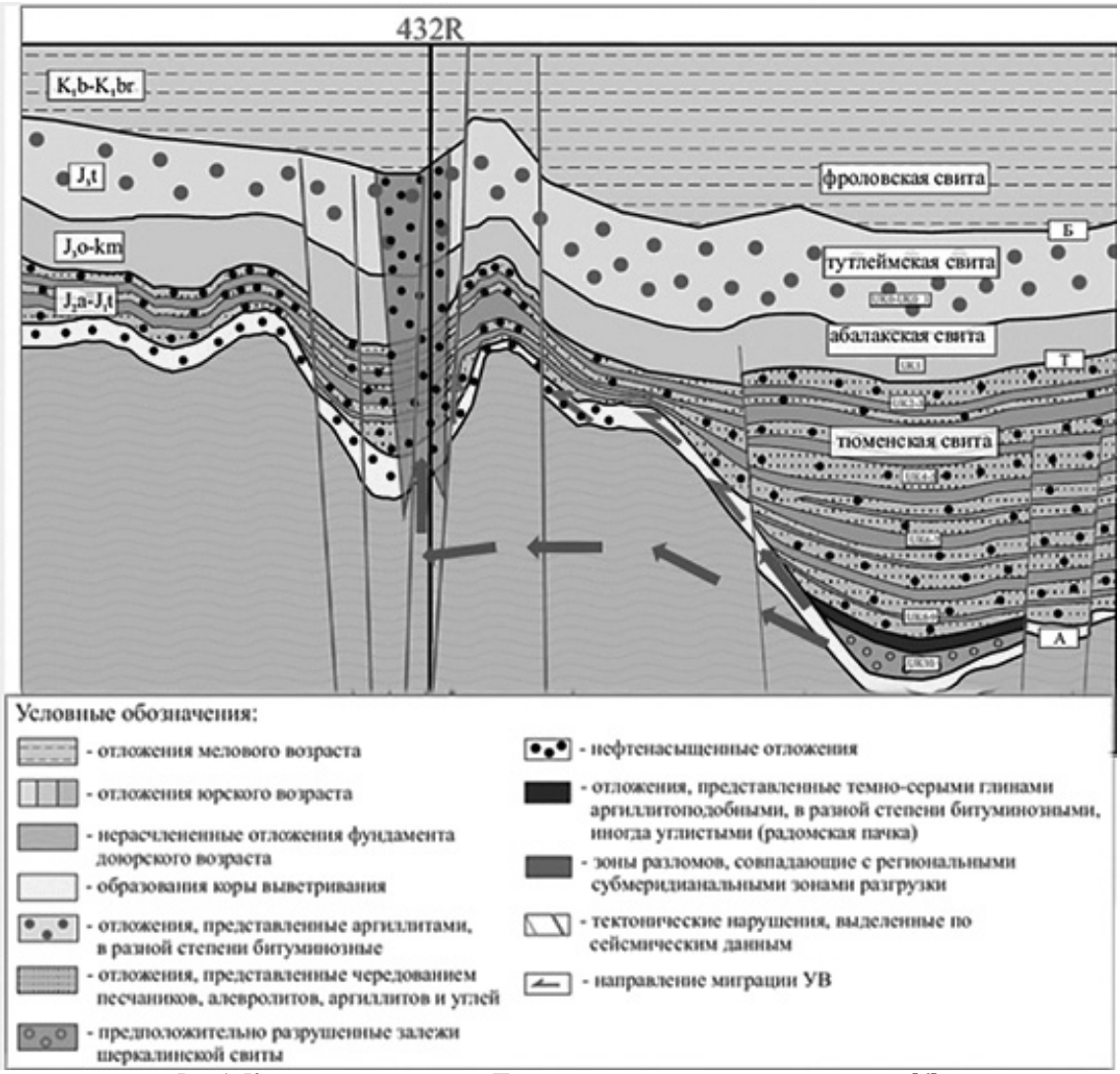


Рис.1. Концептуальная модель Пальяновского месторождения в разрезе [6]

Следует, однако, подчеркнуть, что выделение малоамплитудных нарушений по данным сейсмики связано с трудностями, обусловленными ограниченной разрешающей способностью метода. При амплитуде нарушений, соизмеримой с половиной длины волны или меньшей её, видимые скачкообразные вертикальные смещения осей синфазности во временных полях часто не наблюдаются.

В современной сейсморазведке разработано немало разных способов выделения малоамплитудных нарушений [8]. К ним относятся: анализ формы горизонта, то есть геометрических атрибутов; использование объёмных атрибутов; разложение волнового поля на кубы амплитуд на разных частотах; миграция дуплексных волн.

Опробуются и более сложные методики, например, совместный анализ диффрагированных и дуплексных волн [9].

Важно отметить, что выделение тех или иных тектонических нарушений основывается, по большей части, на характеристиках обнаруженного «сейсмического артефакта», особенно для малоамплитудных нарушений, а для трещин меньшего масштаба – на данных ГИС. Иными словами, не на прямом определении фильтрационных параметров нарушений, а на косвенных признаках – особенностях геофизических данных.

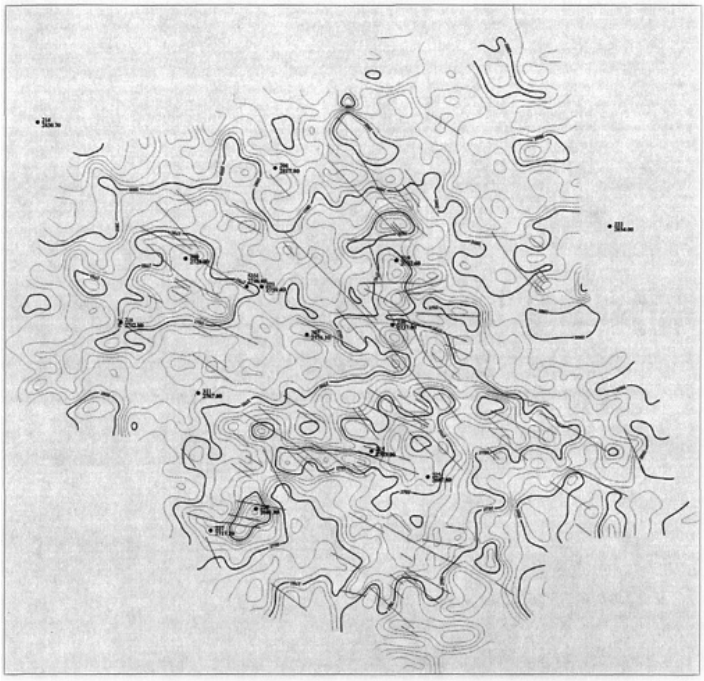


Рис. 2. Структурная карта по кровле баженовских отложений [1];
прямые линии – выделенные малоамплитудные нарушения

Естественно, выделяемые по геофизическим данным артефакты нуждаются в оценке их фильтрационной значимости – по результатам техили иных гидродинамических методов исследований. К сожалению, проверка артефактов, выделяемых по геофизическим данным, обычно по результатам гидродинамических исследований не проводится, хотя методики таких исследований известны и хорошо апробированы [12 - 16]. Важность таких комплексных работ только декларируется [8].

Примеры технологически корректных и достаточно систематических гидродинамических исследований на различных объектах – к сожалению, немногочисленны.

В частности, опыт гидродинамических исследований в отложениях БС имеется на Салымском месторождении [14, 16 - 18], рис. 3. По данным исследований методом гидропрослушивания и анализа динамики пластовых давлений установлено взаимодействие исследованных скважин, и сделан вывод, что разрабатываемый пласт можно представить, как единую фильтрационную систему в пределах зоны, вскрытой скважинами.

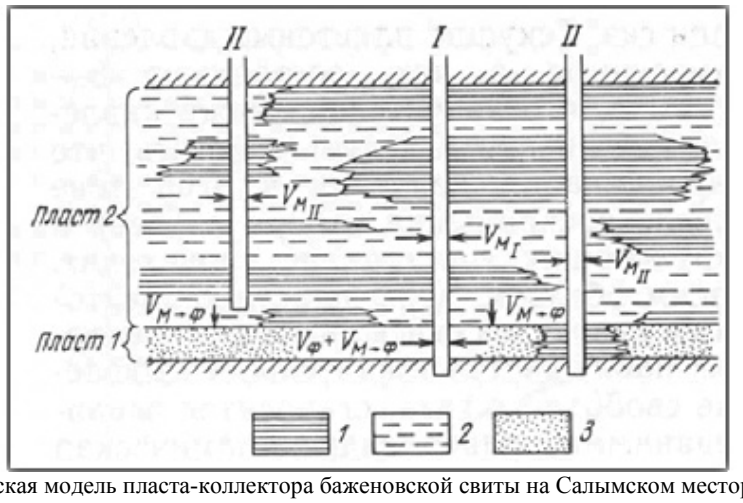


Рис. 3. Схематическая модель пласта-коллектора баженовской свиты на Салымском месторождении по данным гидродинамических исследований [14]; породы: 1 – непроницаемые, 2 – слабопроницаемые, 3 – проницаемые; скважины: I – вскрывшие флюидоподводящую среду, II – не вскрывшие флюидоподводящую среду

Как пример успешного применения ГДИС на очень сложном объекте можно привести и работу [13], в которой проведены исследования методом гидропрослушивания на Юрубчено-Тохомском нефтегазоконденсатном месторождении. Проведённые исследования подтвердили наличие гидродинамической связи между двумя участками залежи, удалёнными друг от друга на 10 км (!). Эти работы – уникальны; судя по литературным источникам, они не имеют аналогов в мировой практике. Показано также, что исследованный участок Юрубчено-Тохомского месторождения представляет собой единую гидродинамическую систему, а участие матрицы карбонатного коллектора в фильтрации флюида не подтвердилось.

1. Комплексный подход для оценки фильтрационной значимости макро- и меганарушений

Для более достоверной оценки фильтрационной значимости выделенных по данным геофизических исследований макро- и меганарушений предлагается применять комплексный подход, включающий в себя:

1. Проведение и анализ традиционных промысловых, геологических, геофизических измерений; формирование на этой основе достоверной базы гидродинамической информации – в первую очередь, измерений дебита жидкости и нефти, объёмов закачки вытесняющих агентов; построение предварительной геологической и гидродинамической моделей объекта с учётом имеющейся промысловой информации о дебитах, давлениях, накопленных отборах и другой промысловой информации. [14–18].

2. Выделение нарушений по сейсмическим данным (с заметными смещениями или малоамплитудных) на основе результатов традиционных или нетрадиционных методов обработки сейсмических сигналов, или проведения специальных измерений и последующей специальной обработки их данных, или по комплексированию сейсмических, космогеологических, геоморфологических, магнитометрических, гравиметрических данных [1, 5, 8 – 10].

3. По данным, полученным в п.1, должна быть оценена средняя проницаемость зон, в которых выделены нарушения.

4. Планирование специальных работ, в первую очередь:

– восстановления давления с регистрацией длительных во времени (длиннопериодных) кривых восстановления давления [12];

– гидродинамических исследований методом гидропрослушивания [13] и, возможно, трассерных исследований [19];

– оценки фильтрационной значимости выделенного нарушения и его параметров по комплексу имеющейся информации традиционных и специальных методов исследования;

– сопоставления полученных фильтрационных параметров выделенных нарушений и различных сейсмических атрибутов, выявление тех сейсмических атрибутов, которые в дальнейшем пригодны для последующего прогнозирования таких нарушений на исследуемом объекте только по сейсмическим данным.

Несколько более детально, предлагаемая последовательность анализа может быть представлена в следующем виде.

Выделенные в п.3. зоны расположения нарушений, скорее всего, должны быть разделены на три категории: низкопроницаемые, среднепроницаемые, высокопроницаемые. Границы категорий определяются по величинам средней проницаемости пород в скважинах согласно результатам анализа керна, измерениям ГИС и анализу промысловых данных.

Выделенные нарушения должны быть нанесены на карту разработки.

В каждой из выделенных зон нарушений проводится сравнение начальных дебитов и кривых обводнённости для скважин:

а) попадающих в полосу выделенного нарушения;

б) отстоящих от предполагаемого нарушения менее чем на характерное расстояние d, определяемое для каждого из типов зон по данным численного моделирования работы одиночных скважин и измерений КВД;

в) отстоящих от выделенных нарушений и границ объекта (уже выявленных на первых этапах), более чем на указанные выше расстояния d для каждого из типов зон.

Далее возможно несколько вариантов.

1. Если скважины групп а) и б) имеют более высокие дебиты и характеризуются гораздо более ранним и быстрым нарастанием обводнённости, чем скважины группы в), то можно сделать вывод, что нарушение действительно существует и имеет проницаемость намного выше, чем вмещающая основная часть пласта.

2. Если скважины группы а) имеют гораздо более низкие дебиты, чем скважины групп б) и в), и характеризуются гораздо более поздним нарастанием обводнённости, чем скважины группы в), или даже отсутствием воды в продукции в течение длительного времени – то

можно сделать вывод, что нарушение, скорее всего, существует и имеет проницаемость намного ниже, чем вмещающая основная часть пласта.

3. Если не наблюдается никакого различия в значениях дебитов, виде кривых обводнённости между всеми тремя группами скважин – то это свидетельствует о том, что или нарушение гидродинамически незначимо и пласт ведёт себя как однородная среда; или нарушение представляет собой узкую непроницаемую границу, незначительно влияющую на разработку (например, ввиду малой её протяжённости или расположения на значительном удалении от пробуренных скважин).

2. Возможность определения дизъюнктивных нарушений по данным ГДИС, гидропрослушивания и трассерных исследований

Для уточнения этих выводов и оценки положения границ узкополосной фильтрационной неоднородности, в скважинах групп а) и б) желательно проведение длиннопериодных измерений кривых восстановления (падения) давления с соблюдением существующих методических и технических рекомендаций [15, 16].

Расчётный пример таких длиннопериодных КВД в однородном пласте с одиночной скважиной и бесконечной непроницаемой границей показан на рис. 4.

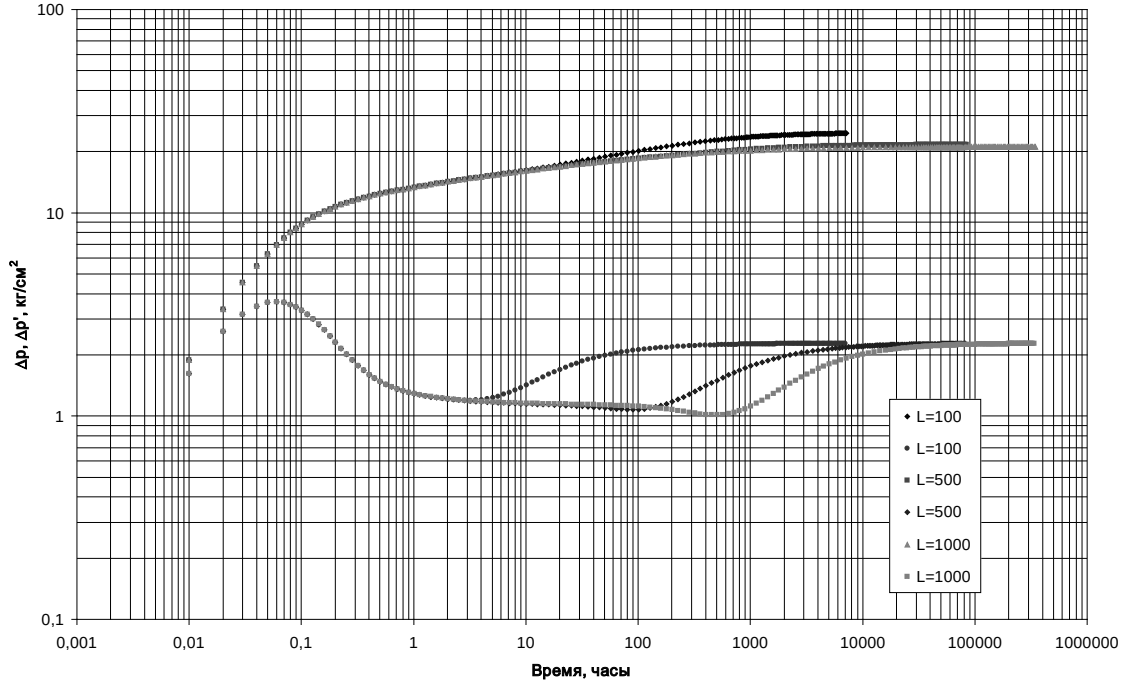


Рис. 4. Расчётная КВД в однородном пласте с одиночной скважиной и бесконечной непроницаемой границей; проницаемость пласта 50 мД, L – расстояние от скважины до границы, м.

Резкий рост производной давления на рис. 4, с её последующим значительным падением обусловлен влиянием ствола скважины. Затем следует горизонтальный участок производной, соответствующий радиальному притоку к скважине. После чего наблюдается рост производной, обусловленный переходным процессом, и выход на второй горизонтальный участок, соответствующий уже влиянию бесконечной непроницаемой границы. Значение производной на втором горизонтальном участке вдвое превышает её значение на первом участке. Наблюдённое время начала второго участка позволяет оценить расстояние до границы. При увеличении расстояния до границы — это время резко увеличивается, а при увеличении проницаемости пласта – резко уменьшается.

Соответственно, разумный выбор скважин для проведения таких исследований определяется предварительными сведениями о свойствах пород в основной части пласта и в пределах полосы расположения нарушения, полученными при анализе истории скважин. Для обоснования выбора скважин, в которых за приемлемое время могут быть получены доста-

точно достоверные результаты, должно проводиться численное гидродинамическое моделирование КВД для зон разной проницаемости, в которых расположены выделенные нарушения. По этим данным находятся характерные расстояния *d*; проведение длиннопериодных КВД осуществляется в скважинах, расстояние которых от нарушения менее *d*.

В тех же целях между скважинами групп а) и б) желательно проведение исследования методом гидропрослушивания, с соблюдением существующих методических и технических рекомендаций [13, 15, 16].

Для обоснования выбора скважин, в которых за разумное время могут быть получены достаточно достоверные результаты гидропрослушивания, проводится численное гидродинамическое моделирование гидропрослушивания для зон разной проницаемости, в которых расположены скважины, намеченные для таких исследований. По этим данным находятся характерные расстояния *l*; проведение гидропрослушивания осуществляется в скважинах, расстояние между которыми менее *l*.

При благоприятном сочетании: а) параметров пород в полосе нахождения нарушения и б) расстояний от нарушения попавших в полосу скважин – по данным КВД и гидропрослушивания возможно определить гидропроводность и пьезопроводность пород в полосе нахождения нарушения.

Если проницаемость пород в полосе расположения выделенного нарушения достаточно высока (средне- и высокопроницаемые породы), то желательно проводить трассерные исследования [19], поскольку они более чувствительны к фильтрационной неоднородности объекта, чем гидропрослушивание [20, 21].

Для закачки трассера выбираются нагнетательные скважины из группы а) [или из группы б) – но расположенные как можно ближе к полосе выделенного предполагаемого нарушения].

Для обоснования выбора скважин, в которых за разумное время могут быть получены достаточно достоверные результаты трассерных исследований, проводится численное гидродинамическое моделирование показаний трассерного метода для зон разной проницаемости, в которых расположены скважины, намеченные для таких исследований. По этим данным находятся характерные расстояния $l1$; проведение трассерных исследований осуществляется в скважинах, расстояние между которыми менее $l1$.

При такой ситуации основная часть закачанного индикатора может быть отобрана в реагирующих добывающих скважинах за разумное время. Появление индикатора в расположенных в самой полосе добывающих скважинах подтверждает факт проницаемости нарушения.

В этой ситуации интерпретация трассерных исследований может быть проведена даже с использованием достаточно простых моделей типа трубок тока.

Поскольку эффективность всех трёх перечисленных методов исследования ограничена значением проницаемостей пород в пласте и в полосе нарушения, а также расстоянием до исследуемого нарушения, то очевидно, что уточнить параметры нарушений удастся далеко не всегда.

Кроме того, в зависимости от сочетания геометрии расположения скважин и нарушения, а также свойств пород, эффективность применения перечисленных выше специальных гидродинамических исследований будет различна в разных частях объекта. В ряде случаев удастся лишь подтвердить гидродинамическую значимость нарушения, в некоторых – подтвердить установленную по иным данным геометрию границ исследуемого участка (бесконечная непроницаемая граница, полубесконечная граница, высокопроводящее нарушение и тому подобное) и лишь в некоторых случаях – определить (может быть, на качественном уровне) параметры пород в полосе нарушения.

Поэтому необходим следующий этап – уточнение гидродинамической модели исследуемого участка, чтобы результаты моделирования согласовывались со всей совокупностью гидродинамических данных: историями работы скважин, результатами КВД, гидропрослушивания и трассерных исследований, проведённых на отдельных скважинах или частях объекта. Ввиду неполноты и отрывочности гидродинамической информации такое соответствие можно обеспечить лишь «в среднем».

Решение может быть найдено при использовании подхода, предложенного в работах [22, 23], заимствованного из теории оптимального управления. Подход основан на выборе математической модели процесса (например, распространения импульса давления или оторочки индикатора), целевой функции (квадратичное отклонение экспериментальных и расчётных данных) и нахождении минимума этой целевой функции путём варьирования параметров нарушения и геометрии рассматриваемого участка месторождения.

Следующий этап – при благоприятном сочетании геолого-геофизических условий – это уточнение набора сейсмических атрибутов, который может быть использован для прогноза нарушений уже по сейсмическим данным.

Заключение

Проведение предлагаемого комплекса гидродинамических исследований позволило бы гораздо объективнее оценить гидродинамическую роль крупных дизъюнктивных нарушений (в том числе и малоамплитудных), выделяемых по сейсмическим, а также иным геологическим и географическим данным. И, таким образом, доказательно обосновать фильтрационную макроструктуру исследуемых залежей, в частности, и баженовской свиты, для последующего проектирования разработки.

Работа выполнена при поддержке гранта РФФИ 18-07-00676 А.

Permeability Influence Evaluation of Productive Reservoir Faults, Identified by Geophysical Survey Data

S.G. Volpin, I.V. Afanaskin, V.A. Yudin

Abstract: Article presents detailed author's approach description of macro- and mega reservoir faults influence evaluation generated after geophysical survey interpretation. Well and reservoir hydrodynamic survey applicability analysed for filtration structure and faults parameters identification of oil reservoir based on interference survey, tracers survey and well testing.

Keywords: well logging, well testing, pressure build-up curve, interference survey, tracers survey.

Литература

1. Ю.А.Вертиевец. Геологическое обоснование освоения трудноизвлекаемых запасов нефти кероген-глинисто-силицитовых пород баженовской свиты района Красноленинского свода // Диссер. на соиск. уч. степ. к.г.-м.н. Москва. 2011.
2. Э.М.Халимов, В.С.Мелик-Пашаев. О поисках промышленных скоплений нефти в баженовской свите // Геология нефти и газа. № 6. С. 1—10. 1980.
3. Ю.Е.Батурин Ю.Е., В.П.Сонич, М.Ю.Ахапкин и др. Основные итоги и перспективы разработки баженовской свиты Салымского месторождения // Геофизика. №4. С. 211—218. 2007.
4. А.Я.Фурсов, Б.Ю.Вендельштейн, А.В.Постников, Е.В.Постников и др. Разработка методики, подсчет геологических и оценка извлекаемых запасов нефти и газа Салымского месторождения (пласт ЮС0). Геологический отчёт ООО «Квант» / Геологический фонд по Ханты-Мансийскому округу. 2002.
5. А.С.Смолин. Литологические особенности и нефтегазоносность баженовской свиты на территории Среднего Приобья // Диссер. на соиск. уч. степ. к.г.-м.н. Москва. 2006.
6. К.В.Стрижнев, В.Т.Литвин. Возможность применения технологии интенсификации добычи нефти для коллекторов баженовской свиты // Электронный журнал «Георесурсы. Геоэнергетика. Геополитика». Вып. 10. 12.2014.
7. И.А.Никульшин. Условия формирования и прогноз нефтеносности отложений баженовской свиты на примере Сахалинской и Восточно-Сахалинской площадей // Диссер. на соиск. уч. степ. к.г.-м.н. Москва. 2008.
8. И.Ю.Хромова. Практическое сравнение методик прогноза трещиноватости по сейсмическим данным // Технология сейсморазведки. № 2. С. 62—69. 2010.
9. Д.Н.Твердохлёбов. Разработка методики выделения и использования сейсмических волн от дизъюнктивных нарушений с целью повышения надёжности и детальности их картирования // Диссер. на соиск. уч. степ. к. т. н. Москва. 2011.
10. А.М.Полищук, С.А.Власов, Р.Л.Салганик и др. Проблема интенсификации добычи нефти из коллекторов месторождений Западной Сибири // Бурение и нефть. № 10. С. 30—35. 2003.

11. Е.Д.Глухманчук, В.В.Крупицкий, А.В.Леонтьевский. Баженовская нефть – «сланцевые технологии» и отечественный опыт добычи // Недропользование XXI век. Вып. 7. С. 32—37. 2015.
12. Л.Г.Кульпин, Ю.А.Мясников. Гидродинамические методы исследования нефтегазоводоносных пластов. М.: «Недра». 200 с. 1974.
13. Р.К.Разяпов, А.С.Сорокин, С.Г.Вольпин, А.В.Свалов, Ю.М.Штейнберг, П.В.Крыганов. Уточнение геологического строения Юрубчено-Тохомского месторождения по данным исследований скважин методом гидропрослушивания // Нефтяное хозяйство. № 8. С. 10—15. 2013.
14. И.Д.Умрихин, С.Г.Вольпин, О.В.Ломакина, В.И.Погонищев, В.В.Самардаков. Уточнение гидродинамической модели залежи и коллектора на Салымском месторождении // Геология нефти и газа. -№ 1. С. 52—57. 1988.
15. Методические указания по комплексированию и этапности выполнения геофизических, гидродинамических и геохимических исследований нефтяных и нефтегазовых месторождений. РД 153-39.0-109-01. Москва: Минэнерго РФ. 2002.
16. Методическое руководство по гидродинамическим исследованиям сложнопостроенных залежей. РД-39-0147035-234-88. Москва: Миннефтегазпром, ВНИИ. 1989.
17. И.Д.Умрихин, С.Г.Вольпин и др. Определение гидродинамической модели залежи и типа коллектора Салымского месторождения // Нефтяное хозяйство. № 6. С. 33—38. 1984.
18. С.Г.Вольпин, О.В.Ломакина, И.В.Афанаскин, В.А.Юдин. Особенности геологического строения и энергетического состояния залежи в отложениях баженовской свиты // Conference «Geopetrol 2014: Exploration and production of oil and natural gas reservoirs – new technologies, new challenges». Krakow. Papers. P. 85—95. 2014.
19. Методическое руководство по технологии проведения индикаторных исследований и интерпретации их результатов для регулирования и контроля процесса заводнения нефтяных залежей. РД 39-0147428-89. Грозный: СевКавНИПИнефть. 1989.
20. D.Bogatkov. Integrated Modeling of Fracture Network System of the Midale Field. MS thesis. Edmonton, Alberta, Canada: University of Alberta. 2008.
21. Н.Р.Кривова. Разработка и исследование системы эксплуатации коллекторов многопластовых месторождений с разрывными нарушениями // Диссер. на соиск. уч. степ. к. т. н. Тюмень. 2009
22. J.L.Landa, R.N.Horne. A Procedure to Integrate Well Test Data, Reservoir Performance History and 4-D Seismic Information into a Reservoir Description // Annual Technical Conference and Exhibition. San-Antonio. Texas. 1997.
23. A.Lange, J.Bouzian, B.Bourbiaux. Tracer-Test Simulation on Discrete Fracture Network Models for the Characterization of Fractured Reservoirs // SPE 94344. 2005.

Применимость результатов трассерных исследований для моделирования процессов разработки нефтяных месторождений

Н.П. Ефимова, А.К. Пономарев, Ю.М. Штейнберг, Д.В. Солопов

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: nefimova@niisi.ras.ru

Аннотация: В статье рассматриваются вопросы применимости трассерных (индикаторных) исследований для определения гидродинамических свойств пластов в межскважинном пространстве. Перечислены теоретические подходы и практические аспекты проведения и интерпретации трассерных исследований для разных условий.

Ключевые слова*:* трассерные исследования, фильтрационно-емкостные свойства, нефтяное месторождение.

Введение

Одна из первых работ [1, 33], охватывающая как теоретические, так и практические аспекты трассерных исследований в нефтедобыче появилась в 1995 г. В данной работе значительное внимание уделяется лабораторным и промысловым исследованиям, также представлены результаты математического (компьютерного) моделирования процессов движения трассеров в нефтенасыщенных коллекторах. Существенный разрыв между теоретиками и практиками присущ и отечественному опыту трассерных исследований [2-8], характеризующихся в основном качественными подходами и результатами. Теоретические наработки, опубликованные, например, в монографии [9] и статье [10], никак не используются.

В настоящее время [33] наряду с прогрессом в области технических средств можно констатировать и развитие методов интерпретации результатов их применения [4, 12, 13], связанное с успехами математического моделирования течения в неоднородных средах.

Простейшие математические методы интерпретации результатов трассерных исследований позволяют перейти от чисто качественной интерпретации к определению числовых значений параметров используемой модели. В наибольшей степени исследованы слоистые модели с изолированными слоями (послойными флюидоупорами), в которых подлежит определению функция распределения проницаемости по слоям [33].

В работе [4] используется слоистая модель, в которой течение в каждом пропластке предполагается одномерным (галерея-галерея), течение считается двухфазным. Модель не позволяет учесть искривление линий тока, непостоянство скорости фильтрации в пласте, существование зон с очень малыми скоростями фильтрации, т.е. те эффекты, которые присущи только двумерным и трехмерным течениям, и оказывают влияние на динамику наблюдаемой концентрации трассера в скважине. Влияние неодномерности решения обратной задачи в [4] учитывается приближенно через коэффициент дисперсии. Для решения обратной задачи в [4] применяется итерационная процедура, требующая многократного численного решения уравнений фильтрации численными методами, (весьма ресурсоемкая) [33].

В статье [11, 33] рассмотрена слоистая модель с двумерными изолированными слоями. Течение в каждом слое предполагается однофазным, при переносе примеси учитывается продольная дисперсия. Метод расчета переноса индикатора потоком основан на аналитическом решении уравнения конвективной диффузии для одномерного течения в трубке тока. Обратная задача в [11] решается с помощью методов поиска экстремумов нелинейного функционала. Следует отметить, что при такой постановке обратной задачи отсутствует ответ на вопрос о единственности решения и его

чувствительности к погрешности исходных данных.

В [14, 15] рассмотрена обратная задача для слоистого пласта в случае двухфазного течения (модель Баклея-Леверетта). Задача сводится к решению интегрального уравнения Фредгольма первого или второго рода с автомодельным ядром. Такой подход достаточно универсален, единственность решения задачи может быть исследована, экономичен с точки зрения вычислительных затрат: достаточно однократного расчета течения в одном слое (пропластке). Обобщение этого подхода на случай переноса нейтральной примеси фильтрационным потоком принципиальных трудностей не вызывает [33].

Идентификация латеральных флюидоупоров по данным ГДИ и трассерных исследований – более сложная задача, приемлемого решения которой до сих пор не получено. Известно решение подобной задачи (выявление непроницаемого барьера) по данным воспроизведения истории разработки [16] гипотетического участка пласта. По сложившемуся мнению, решение этой задачи можно получить только с помощью комплексирования различных методов исследования: ГДИ, закачка трассеров, воспроизведение истории разработки и т.д. Если использовать только какой-то один из известных методов исследования для решения задачи идентификации латерального флюидоупора, то решение может оказаться неоднозначным.

Следует отметить так же еще одну важную проблему трассерных исследований [33]. По данным многих исследований после запуска трассера в нагнетательную скважину первые порции трассера с весьма незначительной его концентрацией в продукции обнаруживаются в добывающих скважинах, находящихся на значительном удалении от нагнетательной скважины, очень быстро. В некоторых промысловых экспериментах отмечается приход первых порций трассера в добывающие скважины через несколько часов. Как правило, это объясняют скрытой трещиноватостью пласта в межскважинном пространстве. Как правило, длительность трассерных исследований варьируется от нескольких месяцев до года. Поэтому возникает вопрос, возможно ли за такое короткое время, опираясь только на замеры малых концентраций первых порций трассеров в добывающих скважинах получить какие-то количественные результаты, имеющие отношение к геологическому строению пласта или к его фильтрационно-емкостным характеристикам.

Все эти соображения лишний раз подтверждают необходимость комплексирования различных методов исследования продуктивных пластов[17-19].

Так в статье [17, 33] предлагается технология построения реалистичной с геологической точки зрения модели пласта с разномасштабной неоднородностью трещиновато-пористой среды с дискретной (детерминированной) системой трещин.

Эта модель строится с учетом сейсмики, скважинной информации и по данным обнажений. Затем эта модель уточняется по данным ГДИ, интерференции скважин и профилей приемистости и продуктивности скважин, производится апскейлинг модели, что делает возможным ее применение для реальных расчетов. Данная модель позволяет в частности предсказывать ранние прорывы воды.

Предложенная в [17] технология конструирования геологической модели позволила авторам [17, 19] создать трехмерную гидродинамическую модель однофазной фильтрации для исследования движения трассеров в трещиновато-пористой среде 3D DFN (Discrete Fracture Network), рис. 1.

Данная гидродинамическая модель [33] позволяет получить дополнительную количественную информацию при проведении трассерных исследований, которая затем интегрируется с информацией, полученной с помощью гидродинамических исследований пластов и скважин. Так, например, с помощью ГДИ выявляется анизотропия фильтрационных характеристик пласта, а по результатам трассерных исследований выявляются основные фильтрационные каналы, обеспечивающие эту анизотропию. Предложенная в [18, 19] модель тестировалась на известных аналитических решениях, представленных в монографии [1].

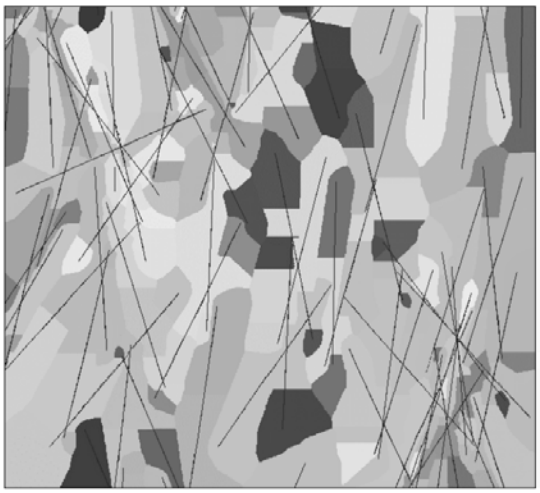


Рис. 1. Изображение плоскости дискретизации в середине пласта

Описание метода трассерных исследований

Рассматривается только применение трассеров для контроля фильтраци-онных потоков жидкости в межскважинном пространстве.

Суть метода состоит:

• в закачке в прискважинную зону пласта вокруг нагнетательной скважины (или скважин) оторочки какого-либо специального раствора (или разных растворов в разных нагнетательных скважинах);

• дальнейшем нагнетании в эту скважину (скважины) вытесняющего пластовые флюиды агента (в соответствии с принятой технологической схемой разработки);

• отборе проб на устье выбранных добывающих скважин на исследуемом участке;

• анализе проб в лабораторных условиях;

• обработке данных измерений и интерпретации полученных результатов.

Для наглядности общая схема показана на рис. 2 из работы [20].

Рис. 2. Схема трассерного метода [20]

Измеряемыми величинами при таких трассерных исследованиях являются:

• факт обнаружения трассера в продукции конкретной добывающей скважины, что свидетельствует о наличии гидродинамической связи между этой скважиной и нагнетательной, в которую закачан трассер. В дальнейшем производится геологическая и промысловая интерпретация этого факта.

• Кривая зависимости «концентрация трассера - время»

• Определяемые на данной кривой времена прихода пиков («максимумов») концентрации трассера

• Величины, вычисляемые по данной кривой: площади под выявленными пиками, суммарное количество трассера, извлечённое из данной добывающей скважины и т.д.

Задачи, решаемые с помощью трассерного метода

Применение трассерного метода позволяет решать следующие задачи [21 – 24, 25, 26, 27, 28, 29, 30, 31, 32]:

1. Установить наличие гидродинамической связи между отдельными нагнетательными и добывающими скважинами, а в ряде случаев - между отдельными пластами.
2. Качественно оценить распределение потока нагнетаемой воды между отдельными нагнетательными и добывающими скважинами, а в ряде случаев - между отдельными пластами.
3. Оценить истинную скорость пластовых флюидов.
4. Качественно оценить степень фильтрационной макронеоднородности изучаемого участка пласта, выявить высокопроницаемые и трещиноватые участки пласта.
5. Выявить на исследуемом участке наличие высокопроводящих путей фильтрации (ВПФ), оценить на качественном уровне их проницаемость по воде и объём.
6. При использовании на каждой нагнетательной скважине своего отдельного индикатора - выявлять схему связей между эксплуатационными скважинами и устанавливать скважины, определяющие динамику обводнения исследуемого участка.
7. При совместно-раздельной эксплуатации - исследовать движение флюидов по разным разрабатываем пластам и перетоки между ними.

Полученные данные весьма существенны для промыслового геолога при решении различных стоящих перед ним задач:

- оценке параметров заводнения и эффективности этого процесса,
- оценке текущей нефтенасыщенности пласта,
- выборе вида и определении эффективности проведенных ГТМ, в первую очередь, мероприятий по ограничению водопритоков по высокопроницаемым путям фильтрации,
- определении охвата разрабатываемой залежи процессом вытеснения нефти или газа и степени влияния на него отдельных скважин,
- выявлении особенностей характера фильтрации и вытеснения нефти из пласта,
- уточнении гидродинамической модели исследуемого объекта,
- подборе скважин-кандидатов для выравнивания профиля приёмистости.

Обнаружение высокопроводящих путей фильтрации по данным трассерного метода

Ещё данные ранних трассерных исследований [1 - 4] показали, что скорость перемещения части нагнетаемой воды в пластах достигает весьма больших величин, исчисляемых сотнями метров за сутки.

Значения скорости более 100 м/сут зафиксированы на многих исследованных с применением индикаторов площадях Ставропольского края, Пермской и Мангышлакской областей, Татарии, Башкирии и Белоруссии. Наибольшая величина, равная 10,2 - 10,6 км/сут, установлена на Осташковичском (БССР) и ранее на Дерюжевском (Куйбышевская область) месторождениях [1 - 4].

Аномально быстрое поступление индикатора в добывающие скважины, как правило, характеризует преждевременные прорывы отдельных частей фронта вытеснения нефти водой. Диапазон изменения максимальной скорости опережающего перемещения нагнетаемой воды в различных направлениях от одной и той же нагнетательной скважины в большинстве случаев весьма широкий [1 - 4]:

по Ачакулакскому месторождению 13,8-231,0 м/сут, Осташковскому 396,5-10200,0 м/сут, Старогрозненскому 14,3-73,3 м/сут, Ромашкинскому (Восточно-Линенагорская площадь) 158,0-480,0 м/сут, Осинскому 144,0 - 900,0 м/сут.

Время движения первых порций меченой воды в пластах, как правило, не согласуется с такими параметрами, как режимы работы добывающих и нагнетательных скважин и расстоянием между ними, обводненностью извлекаемой нефти.

Весьма большие расхождения отмечаются при сопоставлении вычисленных (по усредненным геолого-промысловым данным) и фактических значений времени первых поступлений индикатора, меньшие – при рассмотрении времени перемещения основных объемов меченой воды.

Во всех случаях для трещиноватых отложений различия более существенны, чем для пористых пластов, очевидно, из-за достоверного определения толщины пласта фильтрующей жидкостью.

По мнению Соколовского Э.В. [1 - 4] интенсивное перемещение меченой воды по исследованным залежам связано со строением пластов, в частности с наличием высокопроницаемых путей движения жидкости. Первые прорывы индикатора в добывающие скважины показывают, что по каждому фильтрационному каналу, как правило, перемещается небольшой объем жидкости, составляющий от десятитысячных долей до 1% нагнетаемой воды в соответствующую скважину, через которую введен в пласт индикатор. На эффективность заводнения фильтрационные пути опережающего движения могут влиять только в сумме.

Однако, очевидно, что важнейшей фильтрационной характеристикой процесса заводнения является скорость движения фронта воды, вытесняющего основные извлекаемые запасы нефти. Во всех исследованиях фронтальная скорость составляет 0,3 - 5,2 м/сут и значительно ниже, чем у первых поступающих в добывающие скважины порций индикатора. Значения фронтальной скорости, полученные по Арланскому и Ромашкинскому (Холмовская площадь) месторождениям [1 - 4], находятся в диапазоне 0,6 - 2,8 м/сут. Все эти величины качественно согласуются со средними значениями фильтрационных параметров пород и классической теорией фильтрации [10].

Максимальная и фронтальная скорости образуют спектр, в пределах которого заключаются скорости опережающего перемещения нагнетаемой воды из-за неоднородности пласта. Как правило, он весьма широк. В частности, по Холмовской площади он составляет 0,6-300 м/сут [1 - 4].

Аналогичные результаты получены на коллекторах различного типа (поровых, трещинных), и не наблюдается количественной зависимости скорости вытеснения нефти водой от типа коллектора.

Часто считается, что аномально высокие скорости перемещения меченой воды в естественных пластах связаны с трещиноватостью отложений. Однако существуют факты, не согласующиеся с такой точкой зрения: в чисто трещинных коллекторах верхнемеловых залежей Брагунского и Октябрьского месторождений Чечено-Ингушской АССР потребовалось 1-4 года, чтобы индикатор прошел путь 1,1-6,0 км, а, например, в поровом коллекторе месторождения Колодезное, где возможны единичные трещины, - всего 42-545 сут. при расстоянии 0,8-2,8 км. [1 - 4].

Очевидно, чем меньше трещин в пласте и чем более четкую направленность они имеют, тем благоприятнее условия для быстрых прорывов нагнетаемой воды в добывающие скважины. Факт аномального прорыва индикатора в добывающие скважины следует рассматривать как один из показателей возможного движения нагнетаемой воды по пустотам вторичного происхождения. Для однозначного решения вопроса о трещиноватости пласта требуется совместный анализ результатов применения меченой жидкости или газа, изучения керновых материалов и обнажений горных пород, данных геофизических и гидродинамических исследований [1 - 4].

Наличие аналогичных высокопроводящих путей фильтрации (ВПФ) наблюдается практически во всех трассерных исследованиях, проведенных с различными трассерами и различных районах нашей страны, начиная с первых работ СевКавНИПИ.

Например, данные трассерных исследований в Западной Сибири, проведенные более чем на 23 различных месторождениях, показывают:

• Наличие ВПФ практически на всех месторождениях

• Объём каналов ВПФ от одной нагнетательной скважины варьирует в пределах 59 - 7631 м3, при среднем значении 497 м3

• Проницаемости каналов ВПФ колеблется в интервале 0,055 - 251 Д, что на несколько порядков превышает среднюю проницаемость пород, формирующих основной объём пласта

• Скорости фильтрации закачиваемой воды, меченной индикаторами, составляют 0,8 - 2000 м/ч (!), тогда как скорость продвижения основного фронта вытеснения на несколько порядков ниже и обычно согласуется с классической теорией фильтрации [10].

• Регистрируются несколько пиков концентрации индикатора, с подъёмом концентрации от 1 до 12 раз, что свидетельствует о наличии нескольких ВПФ

• Раскрытость каналов ВПФ оценивается (4 - 1000) · 10^{-3} мм и отмечено увеличение раскрытости в процессе разработки залежи

• Давление нагнетания (репрессия на пласт) влияет на раскрытость ВПФ

• Между матрицей породы и каналами ВПФ нет обмена жидкостью

• Общий объём воды, которая фильтруется по ВПФ и расходуется непроизводительно с точки зрения вытеснения нефти, составляет 8 - 43% от общего объёма закачиваемой воды

• Преимущественная ориентация прохождения трассера, как правило, происходит в двух перпендикулярных направлениях: ЮЗ - СВ и ЮВ - СЗ

• Снижение давления нагнетания, как правило, не приводит с смыканию каналов ВПФВПФ

Аналогичные выаоды сделаны и в работе по результатам трассерных исследований многих залежей месторождений Западной Сибири (Южно-Ягунское, Дружное, Лась-Еганское, Поточное, Северо-Поточное, Ватинское, Тюменское, Ново-Молодежное, Гун-Еганское, Мало-Черногорское, Талинское и др.). Выявлено наличие аномально высокопроницаемых (до 30-5000 мкм2) каналов, а скорость прохождения по ним закачиваемой воды достигает 10-180 м/ч, что в 1000-2000 раз выше характерных значений для терригенных (гранулярных) коллекторов.

Из этих и других данных делается вывод, что тектонические движения, происходившие в пределах тех или иных месторождений или площадей, оказывают значительное влияние на фильтрационно-емкостные свойства (ФЕС) обычных (первичных) гранулярных коллекторов, а также способствуют образованию трещинно-кавернозных (вторичных) коллекторов.

На материале данных многочисленных измерений на разрабатываемых пластах месторождений Нижневартовского свода было построено распределение скорости движения меченой индикаторами воды по выявленным каналам фильтрации. Диапазон этих скоростей — от единиц до сотен метров в сутки, рис. 3.

При трассерных исследованиях на Приобском месторождении в 6-и из 10-и контрольных добывающих скважин появление первой порции индикатора зарегистрировано через 25 - 616 часов после его закачки в нагнетательную скважину, при расстоянии между нагнетательной и добывающими скважинами от 260 до 1689 м. Оценённая скорость фильтрации составляет 0,42 - 17, 2 м/час, проницаемость каналов 5,2 - 143,3 Дарси (!), что на 2 -3 порядка превышает среднюю проницаемость пород по керну и ГИС (единицы и десятки мД!). При этом объём каналов составлял 1548 $м^3$.

Превращение чисто порового коллектора в трещинно-поровый в процессе разработки отмечается и на одном из месторождений ЯНАО. При этом скорости прохождения закачиваемой воды, меченной индикатором, значительно превышали характерные скорости воды в поровом коллекторе. Это стало свидетельством наличия в пласте разветвлённой сети высокопроницаемых путей фильтрации, причём одним из факторов их появления называется развитие техногенной трещиноватости.

На Верхтарском месторождении при трассерных исследованиях, охватывающих 17 нагнетательных скважин, выявлено 114 пар взаимодействующих скважин. При этом обнаружено 17 ВПФ, по которым вода поступала в скважины через 2 -10 суток.

На Ершовом месторождении в 33 скважинах поступление флюоресцеина наблюдалось через 17 - 166 часов после закачки, что дало оценку скорости продвижения воды в 5 - 74 м/час (!), т.е. на 2 -3 порядка выше нормальных скоростей фильтрации в поровом коллекторе.

На Средне-Хулымском месторождении поступление флюоресцеина натрия было отмечено в добывающих скважинах

через 4 -157 часов после закачки и оцененные скорости движения воды составили от 180 до 3000 м/сут.

Хотя некоторые из полученных результатов, возможно, могут быть связаны с методическими ошибками и измерительными погрешностями, колоссальный объём подобных наблюдений не может быть объяснён только этими обстоятельствами. Поэтому наличие таких ВПФ в настоящее время сомнений уже не вызывает.

Природа появления таких выскопроводящих каналов практически на всех отечественных месторождениях до конца не исследована и не ясна.

Наблюдаемое существование ВПФ было одним из оснований для разработки варианта модели среды «с двойной пористостью», в которой, в отличие от традиционной трещинно-блоковой модели [10], пласт представляет собой некое подобие пористо-трещиноватой среды в виде каналов — ручейков с высокой проницаемостью и поровых блоков — островков нефти. Вода от нагнетательных скважин к добывающим движется именно по этим «ручейкам», вытесняя находящуюся в них нефть, а затем начинает бесполезно циркулировать по каналам, оставляя «островки» нефти практически не тронутыми и размывая их только по берегам, посредством капиллярных сил.

По мнению некоторых авторов «ручейковая» фильтрация встречается и в техногенных трещинах, образование которых обусловлено самопроизвольным гидроразрывом пласта при закачке воды под высоким давлением и в больших объемах.

Согласно этой теории, условно так и названной «ручейковой», за ростом добычи нефти, отбираемой из высокопроницаемых каналов, должно произойти резкое ее падение с последующей стабилизацией при низкой добыче. Это свидетельствует о работе поровых блоков, вытеснение нефти из которых происходит очень медленно.

Однако, насколько можно судить по литературным данным, эта концепция пока не получила широкого развития.

В любом случае, в настоящее время большинство исследователей склоняется к выводу, что подобные высокопроводящие пути фильтрации имеют техногенное происхождение. При этом оно обусловлено:

1) наличием динамо-напряжённых зон (линеаментов) и флексурно-разрывных нарушений осадочного чехла и

2) большими величинами депрессий и репрессий на пласт при бурении, освоении и добыче.

Следует подчеркнуть, что если обнаружение быстрого прорыва индикатора, оценка скорости его продвижения и эффективной проницаемости может быть сделана весьма уверенно, то, как следует из рассмотрения методических особенностей метода, число ВПФ, их объём и производительность по данным трассерного метода могут быть определены с не слишком большой достоверностью и точностью. Эти величины, хотя они всегда вычисляются при трассерных исследованиях, следует использовать как некое нулевое приближение при корректировке гидродинамической модели и её адаптации ко всей совокупности геолого-геофизических и промысловых данных.

Следует также отметить, что данные трассерных исследований, указывая на наличие каналов аномальной фильтрации и обеспечивая качественную оценку их параметров, не позволяют определить их пространственное расположение.

Зарубежная промысловая практика использования трассеров и анализ данных

Значительная часть работ по использованию трассеров при заводнении не нашла отражения в литературе. Информация по промысловым экспериментам представляет лишь локальный интерес, часть промысловых исследований не была успешной, некоторые исследования, опубликованные в печати, были фрагментами более обширных проектов, причем разделы, связанные с трассерами, не представляли большого интереса для авторов.

Некоторые опубликованные статьи содержат детальную информацию о выборе трассеров, дизайне трассерных исследований, анализе данных о концентрации трассеров в добываемой воде и т.п. Опубликованные результаты трассерных исследований содержат сведения о геологическом строении и параметрах пластов и их влиянии на процесс заводнения. В основном в данных исследованиях реализован качественных подход. Только незначительная часть работ посвящена актуальным проблемам использования трассеров в промысловых условиях.

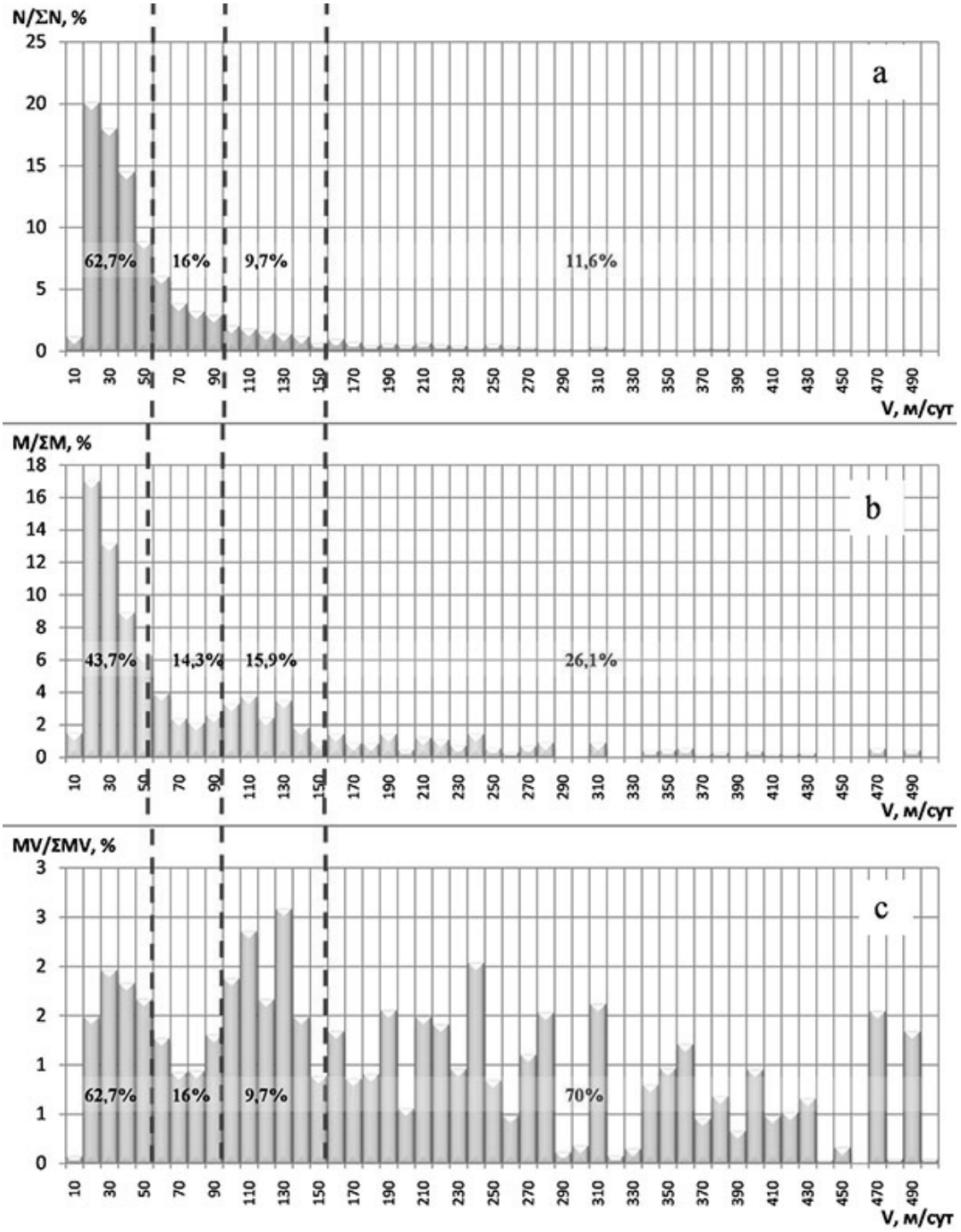


Рис. 3. Характеристики изучаемых объектов разработки свода по результатам трассерных исследований месторождений Нижневартовского свода

Много работ по применению трассеров опубликовано в SPE изданиях и в Petroleum Society of the Canadian Institute of Mining (CIM).

Как уже отмечалось выше, большая часть публикаций содержит только качественные подходы к описанию пластов с помощью трассерных исследований. И только небольшая часть работ посвящена количественной оценке результатов трассерных исследований. В основном они затрагивают:

- использование данных трассерных исследований в комбинации с другими

исследованиями, каждое из которых в отдельности не дает возможности описать объект с надлежащей полнотой;

- использование концептуальных (физико-математических) моделей для определения пространственного распределения концентрации трассеров и описания неоднородности пластов;
- использование компьютерных симуляторов для согласования трассерных данных и геологической модели пласта;
- использование данных о динамике концентрации трассера в скважинах для определения текущей водонасыщенности пласта и коэффициента охвата.

Заключение

Анализ имеющихся литературных и практических данных по трассерным исследованиям для целей уточнения с помощью гидродинамических моделей показал что:

1. Существующие теоретические наработки по технологии исследований и интерпретации имеют ряд ограничений, связанных, в том числе, с соблюдением технологии исследований.
2. Результаты могут иметь значительный разброс относительно оценки свойств пласта другими методами.
3. Для уверенного использования результатов исследований на конкретных объектах необходима предварительная «настройка» технологии и корректировка результатов интерпретации с учетом опыта трассерных исследований на аналогичных объектах.
4. Высокая достоверность результатов трассерных исследований может быть достигнута только при условии обязательного комплексирования с другими методами определения свойств пласта.

Работа выполнена при поддержке Программы ФНИ государственных академий наук на 2013-2020 гг, проект № 0065-2018-0118.

Trasser survey results applicability for oil field development simulation

N.P. Efimova, A.K. Ponomarev, Y.M. Steinberg, D.V. Solopov

Abstract: Article overviews applicability of tracers survey method for filtration capacitive reservoir properties evaluation. Theoretical and practical aspects of interpretation and survey are described for different conditions.

Keywords: tracer survey, filtration capacitive properties, oil field

Литература

1. B.Zemel. Tracers in the oil field. Elsevier Science, 1995. P. 487.
2. Э.В.Соколовский. Применение радиоактивных изотопов для контроля за разработкой нефтяных месторождений. М.: Недра, 1958. – 176 с.
3. Э.В.Соколовский, Г.Б.Соловьев, Ю.Л.Тренчиков. Трассерные методы изу-чения нефтегазовых пластов. М.: Недра, 1986. 157с.
4. М.С.Хозяинов. Диссертация на соискание степени доктора технических наук. М. 1987.
5. В.И.Зайцев, Э.В.Соколовский, С.А.Султанов, М.С.Хозяинов, Ю.С.Шималевич, В.А.Юдин. Применение тритиевого нидикатора для контроля за раз-работкой нефтяных месторождений СССР. М.: ВНИИОЭНГ, 1982. – 40 с.
6. Е.А.Данилова, Д.А.Чернокожев. Применение компьютерных технологий экспресс-анализа и интерпретации результатов трассерных исследований для определения качества выработки неоднородных пластов / Электронное научное издание «Нефтегазовое дело»: http://www.ogbus.ru/ authors/Danilova/Danilova_1.pdf, 2007.

7. М.С.Хозяинов, Д.А.Чернокожев. О возможности оценки соответствия фильтрационных моделей нефтяных пластов реальным объектам разработки по результатам индикаторных исследований / Электронное научное издание «ГЕОразрез»: http://georazrez.uni-dubna.ru/articles/2008/2-2/chernokozhev-o_vozmozhnosti_otcenky.pdf, 2008;
8. РД 39-014-7428-235-89 Методическое руководство по технологии прове-дения трассерных исследований и интерпретации их результатов для регу-лирования и контроля процесса заводнения нефтяных залежей / Э.В.Соколовский, С.Н.Чижов, Ю.И.Тренчиков и др. Грозный: СевКавНИПИнефть, 1989. – 79 с.
9. А.Бан, А.Ф.Богомолова, В.А.Максимов и др. Влияние свойств горных пород на движение в них жидкости. М.: «Красный печатник», 1962. – 276 с.
10. W.E.Brigham, M.Abbaszadeh-Dehgani. Tracer testing for reservoir description. JPT v 39, № 5, pp. 519-527.
11. M.Abbaszadeh-Dehgani, W.E.Brigham. Analysis of well-to-well tracer flow to determine reservoir layering. JPT, 1984, № 10, pp 1753-1762.
12. В.А.Юдин. Принципы количественной интерпретации данных индикаторного метода для простейшей модели заводненного участка. М.: ВИНИТИ, № 6447 Деп.
13. Ю.В.Желтов, В.М.Рыжик, А.Я.Фельдман и др. Количественная интерпре-тация данных индикаторного метода при контроле за разработкой нефтяных месторождений с заводнением. М.: 1982.
14. Г.А.Вирновский. Автомодельные обратные задачи теории нестационарной фильтрации в слоистых пластах. Изв. АН СССР, МЖГ, 1979. № 4. – с. 171-175.
15. Г.А.Вирновский. исследование обратных задач двухфазной фильтрации в слоисто-неоднородных нефтяных пластах. Сб. научных трудов ВНИИ, № 61, 1977. с. 66-75.
16. J.L.Landa. A procedure to Integrate Well Test Data, Reservoir Performance History and 4-D Seismic Information into a Reservoir Description. SPE 38653.
17. B.Bourbiaux, R.Basquet, M.Cacas, J.Damel and S.Sandra S. An Integrated Workflow to Account for Multi-Scale Fractures in Reservoir Simulation Models: Implementation and Benets. SPE 78489.
18. A.Lange, R.Basket and B.Bourbiaux. Hydraulic Characterization of Faults and Features Using Dual Medium Discrete fracture Network Simulation. SPE 88675.
19. A.Lange, J.Bouzian and B.Bourbiaux. Tracer-Test Simulation on Discrete Network Models for the Characterization of Fractured Reservoirs. SPE 94344.
20. Ч.Сунь, И.Шагирбаев и др. «Научные и прикладные аспекты про-екта по гелеполимерному заводнению на месторождении Северные Бузачи»
21. Э.В.Соколовский, В.М.Зайцев. «Применение изотопов на нефтяных промыслах», М., «Недра», 1971
22. Э.В.Соколовский, Г.Б.Соловьев, Ю.Л.Тренчиков. «Индикаторные методы изучения нефтегазовых пластов. М.: Недра, 1986. 157с.
23. М.С.Хозяинов. Диссертация на соискание степени доктора технических наук. М. 1987.
24. В.И. Зайцев, Э.В.Соколовский, С.А.Султанов, М.С.Хозяинов, Ю.С.Шимелевич, В.А.Юдин. Применение тритиевого нидикатора для контроля за разработкой нефтяных месторождений СССР. М.: ВНИИОЭНГ, 1982. – 40 с.
25. Э.В.Соколовский, С.Н.Чижов, Ю.И.Тренчиков и др. РД 39-014-7428-235-89 «Методическое руководство по технологии проведения трассерных исследований и интерпретации их результатов для регулирования и контроля процесса заводнения нефтяных залежей». Грозный, СевКавНИПИнефть, 1989. – 79 с.
26. Н.Р.Кривова. «Разработка и исследование системы эксплуатации коллекторов многопластовых месторождений с разрывными нарушениями», дисс. на соиск.уч.степ. канд. тех. наук, Тюмень, 2009
27. Д.А.Чернокожев. «Совершенствование технологии индикаторных исследований для оценки фильтрационной неоднородности межскважинного пространства нефтяных пластов», дисс.на соиск.уч.степ. канд.техн.наук, Дубна 2008
28. В.И.Зайцев. «Разработка методики контроля за движением закачиваемых вод с применением тритиевого индикатора в условиях рассредоточенных систем заводнения (на примере Ромашкинского месторождения)». Диссертация на соискание ученой степени кандидата геолого-минералогических наук. Бугульма 1983
29. И.И. Букин, Р.Р.Ганиев, Д.Н.Асанбаев, С.П.Калмацкий. «Определение скорости и направления фильтрации по пласту нагнетаемой воды с помощью индикаторов», Труды БашНИПИнефть. 1981. №62

30. Ю.А.Кузьмин. «Разработка методики оценки послойной фильтрационной неоднородности коллекторов Юрского возраста Западной Сибири»: Диссертация на соискание ученой степени кандидата геолого-минералогических наук. М., 1985;
31. М,С.Хозяинов, М.В.Веселов и др. «Методические рекомендации по количественной интерпретации данных индикаторных исследований межскважинного пространства нефтяных месторождений». М.: ВНИИгеоинформсистем, 1988;
32. А.С.Трофимов, С.В.Гусев, С.И.Грачев, В.А.Беляев, С.Г.Батурин. «Трассерные исследования Урьевского месторождения», Известия вузов. Нефть и газ. 1997.- №6;
33. Р.М.Кац. Некоторые проблемы применения трассерного метода для гидродинамического моделирования. Журнал Труды НИИСИ РАН, 2015г. т.5. №2.

Влияние аппроксимации функции распределения дисперсных включений по размерам на структуру полидисперсного пузырькового потока

А. С. Чернышев[1], А. А. Шмидт[2]

СПбО МСЦ РАН - филиал ФГУ ФНЦ НИИСИ РАН, Санкт-Петербург, Россия,

E-mail's: [1]alexander.tchernyshev@mail.ioffe.ru, [2]alexander.schmidt@mail.ioffe.ru

Аннотация: В работе представлены результаты численного исследования течения полидисперсной пузырьковой среды. Математическое моделирование основано на эйлерово-эйлеровском подходе к описанию многофазных сред. Учет полидисперсности пузырьков осуществляется при помощи дискретной модели MUltiple SIze Group (MUSIG), в рамках которой непрерывное распределение пузырей по размерам представляется конечным набором классов, каждый из которых монодисперсен. Анализируется влияние количества выделенных классов на такие параметры потока, как скорость несущей фазы, объемная доля пузырей, удельная площадь межфазной поверхности и изменение функции распределения в потоке.

Ключевые слова: Многофазные среды, пузырьковые течения, эйлеровско-эйлеровское описание, полидисперсность, численное моделирование, многоклассовая модель MUSIG.

1. Введение

Численное моделирование является актуальным и эффективным средством для исследования сложных процессов, в частности, течений пузырьковых сред (см., например, [1-2]). Пузырьковые течения характеризуются широким спектром фундаментальных явлений и процессов, детальное исследование которых важно для получения не только качественной, но и количественной информации о таких потоках.

Как правило, в многофазных и, в частности, в пузырьковых течениях включения являются полидисперсными. Монодисперсное приближение не всегда оправдано и может приводить к существенному рассогласованию данных численного моделирования и эксперимента [3]. В зависимости от способа описания многофазной среды существуют различные подходы к учету полидисперсности.

Основными актуальными подходами, использующимися для описания динамики многофазной среды, являются эйлеровско-лагранжев и эйлеровско-эйлеровский подходы, а также моделирование включений с разрешением межфазной поверхности.

Структура течения в рамках подходов с разрешением межфазной поверхности [4] моделируется не только в области, занимаемой несущей средой, но и внутри пузырей. Высокая степень детализации достигается подробной сеткой, характерный размер ячеек которой должен быть много меньше размеров пузыря. Такие подходы позволяют максимально точно оценивать размер и форму пузырей, режимы обтекания, массовые потоки за счет диффузии и испарения, однако предъявляют высокие требования к производительности компьютера и объему оперативной памяти в силу подробной сетки и необходимости описывать каждый пузырь.

В рамках эйлеровско-лагранжева подхода к описанию многофазных сред [5] полидисперсность может быть легко учтена при помощи введения пробных частиц нужного диаметра, однако в случае их большой концентрации такой подход становится ресурсоемким.

Описание распределения пузырей по размерам в рамках эйлеровско-эйлеровского подхода может быть реализовано различными способами.

Самый простой подобен монодисперсному описанию, однако пузыри имеют одинаковый размер только локально, в пространстве их размер может варьироваться. Локальный размер пузырей принимается равный среднему диаметру по Заутеру [6], что обеспечивает сохранение полной поверхности пузырей и важно в задачах с активным массообменом.

При малых скоростях относительного движения пузырей и несущей среды такой

подход оказывается достаточным.

Однако в случае скоростной неравновесности фаз усреднение диаметра пузырей приводит к некорректному вычислению межфазного силового взаимодействия, что сказывается на общей картине течения, профилях скорости и приводит к неверной оценке объемной доли пузырей.

Один из способов описания полидисперсности в рамках эйлеровско-эйлеровского подхода заключается в введении модели PBM (Population Balance Model) [7]. Эта модель позволяет описывать изменение спектра пузырей по размеру в каждой точке пространства, однако в силу интегро-дифференциальной природы заложенного в нее уравнения вычислительный алгоритм заметно усложняется.

Другой подход основывается на введении набора различных классов монодисперсных пузырей, отличающихся друг от друга по размеру (модель MUSIG) [8, 9].

Количество классов и их диапазон выбирается таким образом, чтобы включить все возможные пузыри, при этом число классов фиксировано во всей области. Для каждого класса записывается своя система уравнений сохранения, содержащая источниковые члены, отвечающие за межфазное взаимодействие.

Такой подход позволяет унифицировать описание всех сред и использовать однотипные уравнения Навье-Стокса как для несущей, так и для каждого класса дисперсной среды.

Существует несколько вариантов модели MUSIG.

Исторически первым был вариант т. н. гомогенной модели, когда для каждого класса записывалось уравнение сохранения массы, а уравнение сохранения импульса было единым для дисперсной фазы. В дальнейшем рост производительности компьютеров позволил решать независимые уравнения Навье-Стокса для каждого класса (т. н. гетерогенная модель MUSIG).

Степень детализации разрешения потока дисперсных частиц в модели MUSIG зависит от числа классов пузырей. Очевидно, что при увеличении количества классов точность описания распределения пузырей по размерам повышается с одновременным увеличением времени расчета. Исследование влияния количества классов на структуру пузырькового потока представляется важным как для апробации модели, так и для формулирования рекомендаций для последующих численных экспериментов.

2. Математическая модель

В представленной работе используется эйлеровско-эйлеровский подход к описанию многофазных сред и гетерогенная модель MUSIG для описания полидисперсности [9]. В рамках подхода каждая фаза рассматривается как континуум с объемной долей α, плотности таких континуумов вычисляются как $\rho = \alpha \cdot \rho^0$, где ρ^0 — плотность вещества соответствующей фазы. В рамках модели MUSIG вводится M классов пузырей, $i = 1 \dots M$, для каждого из классов определен свой радиус R_i, объемная доля α_i, численная концентрация N_i, скорость V_i. Для несущей фазы и всех классов дисперсной фазы записываются уравнения Навье-Стокса в следующем виде:

$$\frac{\partial \alpha_{ib}\rho^0_{ib}}{\partial t} + \operatorname{div}\left(\alpha_{ib}\rho^0_{ib}\vec{V}_{ib}\right) = D_i,$$

$$\frac{\partial \alpha_{ib}\rho^0_{ib}\vec{V}_{ib}}{\partial t} + \operatorname{div}\left(\alpha_{ib}\rho^0_{ib}\vec{V}_{ib}\vec{V}_{ib} + p\bar{\bar{E}}\right) = S_{ib},$$

$$\frac{\partial \alpha_l\rho^0_l}{\partial t} + \operatorname{div}\left(\alpha_l\rho^0_l\vec{V}_l\right) = 0,$$

$$\frac{\partial \alpha_l\rho^0_l\vec{V}_l}{\partial t} + \operatorname{div}\left(\alpha_l\rho^0_l\vec{V}_l\vec{V}_l + p\bar{\bar{E}} - \bar{\bar{\tau}}_l\right) = S_l.$$

$$\alpha_l = 1 - \sum_{i=1}^{M}\alpha_{ib},$$

Здесь p — давление, τ — тензор скоростей деформации, E — единичный тензор, S — источниковое слагаемое, D — диффузионное слагаемое, подстрочный индекс l — несущая жидкость, b — пузыри.

В рамках предложенной модели полидисперсности пренебрегается переносом массы между отдельными классами за счет коалесценции и дробления пузырей. Это предположение справедливо лишь в том случае, когда при заданных параметрах течения рассматривается изначально стабильное распределение пузырей, то есть такое, в котором скорости дробления и коалесценции равны.

Исходная система уравнений Навье-Стокса должна быть дополнена уравнением сохранения численной концентрации пузырей:

$$\frac{\partial N_{ib}}{\partial t} + \operatorname{div}\left(N_{ib}\vec{V}_{ib}\right) = D_i.$$

Исследуемые течения в диапазоне параметров, представляющими интерес, являются турбулентными. В работе используется k-ω-SST модель Ментера [10] с добавленными источниковыми слагаемыми, отвечающими за генерацию и диссипацию турбулентности пузырями:

$$S_{ibk} = \frac{3}{8}\frac{C_{iD}}{R_{ib}}\alpha_{ib}\rho_l\left|\vec{V}_{irel}\right|^3, \quad S_{ib\,\omega} = 0.8\frac{k}{\mu_{turb}}S_{ibk}.$$

Влияние турбулентности на пузыри проявляется в дисперсии траекторий пузырей, которая учитывается при помощи диффузионного слагаемого в уравнениях сохранения объемной доли и численной концентрации пузырей:

$$D_i(\alpha) = \nabla\left(\frac{\mu_{eff}}{\rho_l}\nabla\alpha_{ib}\right), D_i(N) = \nabla\left(\frac{\mu_{eff}}{\rho_l}\nabla N_{ib}\right).$$

Межфазное взаимодействие представлено силами Архимеда, Стокса, Сэффмана, присоединенных масс и силой стенки:

$$S_{ib} = \overrightarrow{F_{iA}} + \overrightarrow{F_{iD}} + \overrightarrow{F_{iL}} + \overrightarrow{F_{iWM}} + \overrightarrow{F_{iWL}},$$

$$S_l = -\sum_{i=1}^{M}\left(\overrightarrow{F_{iD}} + \overrightarrow{F_{iL}} + \overrightarrow{F_{iWM}} + \overrightarrow{F_{iWL}}\right),$$

$$\vec{F}_{iA} = \alpha_{ib}\cdot(\rho_{ib} - \rho_l)\cdot\vec{g},$$

$$\vec{F}_{iD} = \frac{3\rho_l}{8R_{ib}} C_{iD}\vec{V}_{irel}\left|\vec{V}_{irel}\right|, \vec{V}_{irel} = \vec{V}_l - \vec{V}_{ib},$$

$$\vec{F}_{iL} = C_{iL}\alpha_{ib}\rho_l\vec{V}_{irel}\times\mathrm{rot}\,\vec{V}_l,$$

$$\vec{F}_{iWM} = 0.5\,\alpha_{ib}\rho_l\left(\frac{D_b\vec{V}_{ib}}{Dt} - \frac{D_l\vec{V}_l}{Dt}\right),$$

$$\vec{F}_{iWL} = -C_{iWL}\alpha_{ib}\rho_l\left|\vec{V}_{irel} - (\vec{V}_{irel}\cdot\vec{n}_W)\cdot\vec{n}_W\right|^2\cdot\vec{n}_W,$$

здесь $\vec{g}$ - вектор ускорения свободного падения, $\vec{n}_W$ - вектор, направленный по нормали от стенки. Константы C_{iD}, C_{iL}, C_{iWL} взяты из работ [11], [12], [3] соответственно.

Граничные условия для математической модели следующие. На твердых стенках используется условие непротекания для всех фаз и условие прилипания для несущей фазы. Для пузырей используется условие проскальзывания. На входной границе задается начальное распределение пузырей, их объемная доля и скорость. На выходной границе задаются мягкие граничные условия, позволяющие пузырькам свободно покидать расчетную область, а также уровень давления.

3. Численный метод и программная реализация

Сформулированная математическая модель использована при разработке алгоритма моделирования и его программной реализации.

Алгоритм основан на методе конечных объемов и неструктурированных сетках. Аппроксимация уравнений по пространственным переменным имеет второй порядок точности и проводится при помощи противопоточных схем, удовлетворяющих критерию TVD. Аппроксимация по времени реализована при помощи схемы Эйлера с переменным шагом по времени, что позволяет рассчитывать стационарные режимы течения при помощи метода установления по псевдовремени.

Для расчета поля давления и скорости используется алгоритм SIMPLE с поправками, учитывающими многофазность.

Программная реализация включает в себя автономный построитель расчетной области и покрывающей ее сетки, а также препроцессор и решатель.

Все компоненты разработаны с использованием языка программирования C++, с включением возможности распараллеливания при помощи библиотеки OpenMP.

Параллельные вычисления реализуются в многофазном блоке при расчете разных классов пузырей [13].

Разработанный код был протестирован на задачах о движении многофазных пузырьковых сред (см., например, [14]).

4. Постановка задачи и начальные данные

В работе будет проведена серия расчетов с постоянным расходом пузырей и постоянным непрерывным распределением, но различным числом классов. Течение происходит в осесимметричной пузырьковой колонне за счет силы Архимеда, пузыри поступают со дна колонны и покидают ее сверху через свободную поверхность.

На рисунке 1 сверху представлена схема осесимметричной пузырьковой колонны диаметра *D*.

Колонна заполнена водой. Газ в виде полидисперсных пузырей поступает в колонну через вмонтированный в дно соосный осесимметричный аэратор диаметром *d*. Параметры газа соответствуют воздуху при нормальных условиях.

Всплытие пузырей обуславливает движение жидкости в колонне.

Начальное распределение пузырей по размерам, соответствующее данным из статьи [5], представлено на рисунке 1 снизу сплошной кривой в диапазоне от 0.2 до 0.9 мм в диаметре.

При моделировании вводится *M* классов монодисперсных пузырей, причем *M* варьируется от 1 (полностью монодисперсные включения) до 10.

Распределение описывается кусочно-постоянной функцией и для $M = 7$ представлено на рисунке 1 снизу (ступенчатое распределение). Давление окружающей среды считается равным атмосферному, температура — 20ºC.

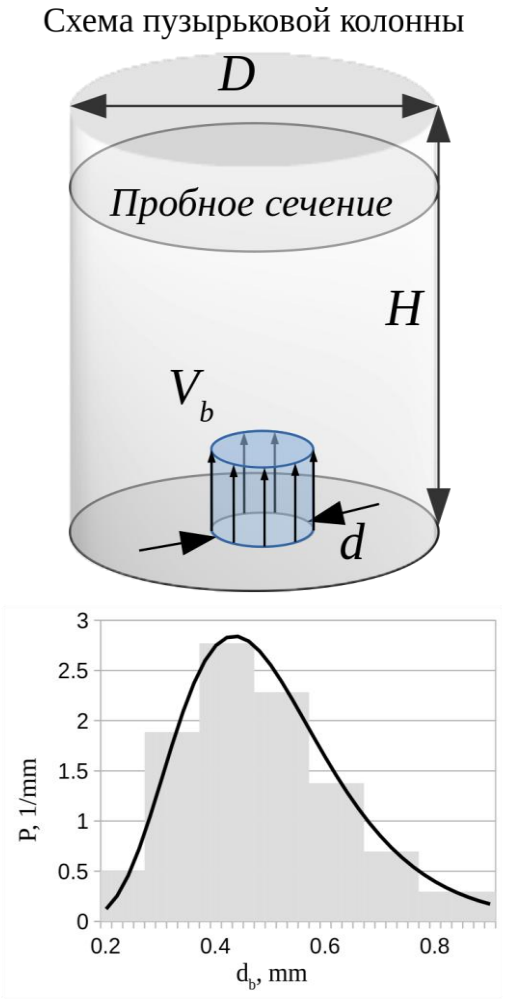


Рис. 1. Схема пузырьковой колонны, распределение пузырей по размерам и кусочно-постоянная аппроксимация. d — диаметр аэратора, D — диаметр колонны, H — высота колонны, V_b — скорость пузырей на входе, P — плотность вероятности распределения пузырей, d_b — диаметр пузырька.

5. Результаты

Для оценки сходимости алгоритма по количеству пузырей используется площадь межфазной поверхности, а также ее производная величина — полный поток через выбранное сечение. Выбор обусловлен тем, что площадь межфазной поверхности для многих практических задач является определяющим параметром. Результаты расчетов представлены на рисунках (2-5). На рисунке 2 приведено распределение плотности межфазной поверхности ($м^2$ / $м^3$) в пробном сечении, отстоящем на 0.45м от нижней поверхности колонны.

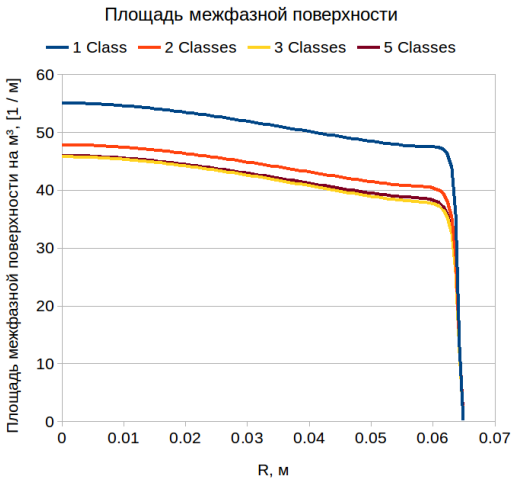


Рис.2. Распределение площади межфазной поверхности, отнесенной к объему, в пробном сечении по радиусу колонны.

Максимум находится в центре колонны из-за осевого расположения аэратора, при приближении к стенкам площадь межфазной поверхности плавно уменьшается, с резким падением практически до нуля у самой стенки. Это связано с действие межфазных сил, в частности, с влиянием силы стенки, направленной всегда внутрь области. При увеличении M с 3 до 5 абсолютные величины по радиусу колонны практически не меняются.

На рисунке 3 приведен нормированный полный поток межфазной поверхности через выходное сечение колонны в зависимости от числа классов пузырей.

Расчет потока производится интегрированием произведения площади поверхности на скорость дисперсной фазы для каждого класса по выбранному сечению, а затем все величины суммируются. Поток нормируется на значение, полученное при M = 10.

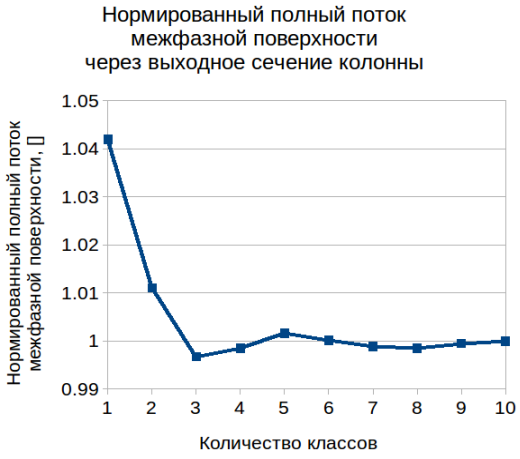


Рис. 3. Нормированный полный поток межфазной поверхности через выходное сечение колонны.

При $M > 3$ различия между вариантами составляют менее 1%, что говорит о достаточной степени детализации для оценки интегральных параметров течения. Тем не менее, наблюдается немонотонный характер сходимости по потоку полной межфазной поверхности. Это может быть объяснено тем, что при варьировании количества классов пузырей меняются и размеры пузырей для каждого класса. Это приводит к изменению сил межфазного взаимодействия, зависящих от размеров пузыря, и перестройке структуры течения.

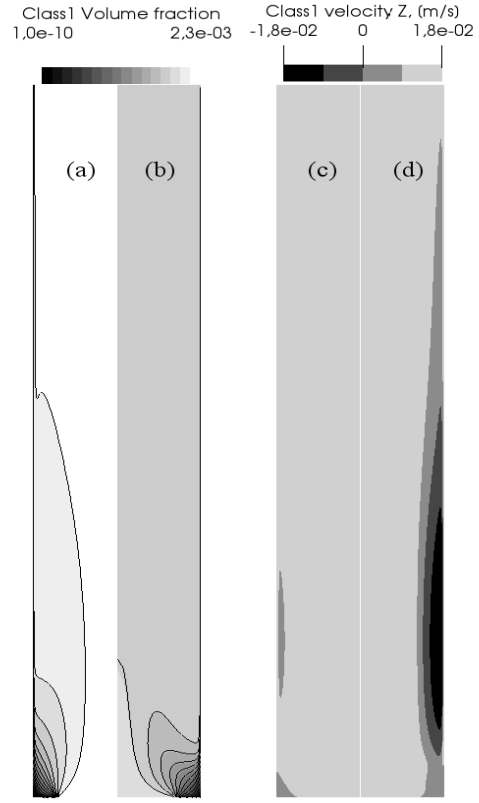


Рис. 4. Распределение объемной доли первого класса пузырей (слева) и их вертикальной компоненты скорости (справа). (a), (c) — $M = 3$; (b), (d) — $M = 7$.

Для детального исследования этого явления сравним поведение первого класса пузырей, обладающего самым малым размером, для случаев, когда $M = 3$ и $M = 7$. На рисунке 4 приведено распределение объемной доли пузырей из первого класса (слева) и их вертикальной компоненты скорости (справа). При этом распределения (a) и (c) соответствуют случаю с 3 классами пузырей, а (b) и (d) — с 7 классами.

Структура течения для 3 и 7 классов пузырей существенно различается. Необходимо отметить, что во всех вариантах существует область обратного тока несущей среды (нисходящее течения вдоль стенок колонны). Однако, если для варианта $M = 3$ наблюдается восходящее течение самых маленьких пузырей во всем объеме колонны, то в варианте M = 7 возникает пристенное обратное движение пузырей. Объясняется этот эффект падением числа Стокса при уменьшении размера пузыря и смещением режима течения пузырей первого класса в сторону равновесного.

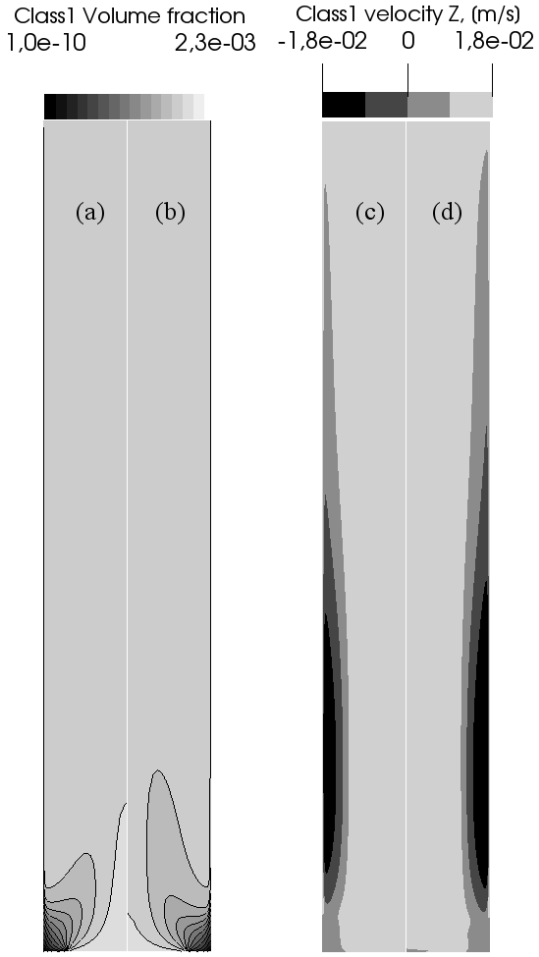


Рис. 5. Распределение объемной доли первого класса пузырей (слева) и их вертикальной компоненты скорости (справа). (a), (c) — $M = 7$; (b), (d) — $M = 10$.

При дальнейшем увеличении числа классов пузырей заметной перестройки течения не наблюдается. Оценим изменения в структуре потока на примере вариантов с 7 и 10 классами. На рисунке 5 представлены объемные доли и вертикальные компоненты скорости для первого класса, (a) и (c) соответствуют случаю с 7 классами пузырей, а (b) и (d) — с 10 классами.

Можно отметить некоторое различие в структуре течения: для варианта с 10 классами длина зоны рециркуляции несколько больше и заметнее выражен локальный максимум объемной доли у стенки внизу колонны. Однако в общем картины течения подобны, что позволяет говорить о

сходимости не только интегральных, но и локальных параметров течения.

6. Оценка времени исполнения алгоритма

Некоторые первоначальные оценки эффектности, скорости исполнения и принципиальная схема алгоритма были указаны в статье [13]. В рамках проведенной работы оценивалось время, необходимое на одну полную итерацию алгоритма в зависимости от количества пузырей. Результаты представлены в таблице 1.

Кол-во классов	Время на итерацию, мс
1	83
2	115
3	135
4	162
5	195
6	217
7	245
8	270
9	300
10	323

Таблица 1. Зависимость времени расчета от количества классов пузырей.

Видно, что в среднем время на одну итерацию на каждый дополнительный класс увеличивается на 27 мс. Таким образом, при определении интегральных параметров есть смысл ограничиться одним классом пузырей для обеспечения удовлетворительного согласия результатов. Увеличение с 1 до 4 классов приводит к двукратному увеличению времени расчета при сопоставимых интегральных характеритиках на выходе из области.

При детальном исследовании картины течения и большом числе классов относительный прирост времени не столь существенен и составляет ~33% для 10 классов по отношению к 7 классам, что может быть критично либо в случае трехмерных задач, либо при исследовании существенно нестационарных процессов.

7. Заключение

Выполнено численное моделирование течения в осесимметричной пузырьковой колонне с различным количеством классов пузырей.

Проведены оценки влияния классов пузырей как на интегральные параметры течения, так и на локальную структуру потока.

При исследовании интегральных параметров представляется рациональным использовать монодисперсное приближение, так как в этом случае достигается оптимум между временем выполнения программы и точностью получаемых результатов.

Увеличение числа классов приводит к существенной перестройке течения и разрешению локальных особенностей, как было показано при переходе от 3 к 7 классам пузырей.

Таким образом необходимо учитывать полидисперсность при детальном исследовании пузырьковых течений. В то же время дальнейшее увеличение классов значительного влияния на структуру потока не оказывает.

Работа выполнена в рамках НИР 0065-2018-0016.

The influence of the approximation of the size distribution function of disperse particles on the structure of a polydisperse bubble flow

A.S. Chernyshev, A.A. Schmidt

Abstract: Results of a numerical study of a polydisperse bubbly medium flow are presented. Mathematical modeling is based on the Eulerian-Eulerian approach to the description of multiphase media. The polydispersity of bubbles is taken into account using the discrete MUltiple SIze Group (MUSIG) model, in which the continuous distribution of bubble sizes is represented by a finite set of classes, each of which is monodisperse. The influence of the number of allocated classes on flow parameters like velocity of the carrier phase, the volume fraction of bubbles, the specific area of the interfacial surface and the change in the distribution function in the flow is analyzed.

Keywords: Multiphase media, bubble flows, Eulerian-Eulerian description, polydispersity, numerical simulation, multi-class MUSIG model.

Литература

1. A.V. Chinak, A.E. Gorelikova, O.N. Kashinsky, M.A. Pakhomov, V.V. Randin, V.I. Terekhov, Hydrodynamics and heat transfer in an inclined bubbly flow, Int. J. of Heat and Mass Transfer, V. 118(2018), 785–801
2. Д. А. Губайдуллин, Б. А. Снигерев, Численное моделирование турбулентного восходящего потока газожидкостной пузырьковой смеси в вертикальной трубе. Сравнение с экспериментом, ТВТ, т. 56(2018), № 1, 61–70
3. Th. Frank, P.J. Zwart, E. Krepper, H.-M. Prasser, D. Lucas, Validation of CFD models for mono- and polydisperse air–water two-phase flows in pipes, Nucl. Eng. and Design, V. 238(2008), 647–659.
4. B. Aboulhasanzadeh, S. Thomas, M. Taeibi-Rahni, G. Tryggvason, Multiscale computations of mass transfer from buoyant bubbles, Chemical Engineering Science, V. 75(2012), 456–467
5. S. Lain, D. Broder, M. Sommerfeld, Experimental and numerical studies of the hydrodynamics in a bubble column, Chemical Engineering Science, 54(1999), 4913-4920
6. W. Yao and C. Morel, Volumetric interfacial area prediction in upward bubbly two-phase flow, Int. J. Heat Mass Transfer, V. 47(2004), 307–328
7. T. Wang, J. Wang, Y. Jin, Population Balance Model for Gas-Liquid Flows: Influence of Bubble Coalescence and Breakup Models, Ind. Eng. Chem. Res., V. 44(2005), 7540-7549
8. S. Lo, Application of the MUSIG model to bubbly flows, AEA Technology, (1996), AEAT-1096.
9. E. Krepper, D. Lucas, T. Frank, H.-M. Prasser, P. J. Zwart, The inhomogeneous MUSIG model for the simulation of polydispersed flows, Nuclear Engineering and Design, V. 238(2008), 1690–1702
10. F. Menter, M. Kuntz, R. Langtry, Ten years of industrial experience with the SST turbulence model, Turb. Heat and Mass Tran. V. 4(2003), 625-632
11. I. Roghair, Y.M. Lau, N.G. Deen, H.M. Slagter, M.W. Baltussen, M. Van Sint Annaland, J.A.M. Kuipers, On the drag force of bubbles in bubble swarms at intermediate and high Reynolds numbers, Chemical Engineering Science,V. 66(2011), 3204–3211
12. A. Tomiyama, H. Tamaia, I. Zunb, S. Hosokawaa Transverse migration of single bubbles in simple shear flows, Chemical Engineering Science, V. 57(2002), 1849–1858
13. A. Chernyshev, A. Schmidt and L. Kurochkin, Numerical Modeling of Polydisperse Bubbly Flows by the OpenMP Parallel Algorithm, Proc. Comp. Sci., V. 108C(2017), 1990–1997
14. A. Chernyshev and A. Schmidt, Numerical modeling of gravity driven bubble flows with account of polydispersion, Journal of Physics: Conference Series, V. 754(2016), No. 3, 032005

Внутренние модели внешней среды, формируемые и используемые автономными агентами

В. Г. Редько

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: vgredko@gmail.com

Аннотация: В статье анализируются возможности формирования и использования внутренних моделей автономными агентами. Под автономным агентом понимается модельный организм. Подчеркивается важность анализа таких внутренних моделей. Рассматриваются процесс формирования и использования внутренней модели для конкретного примера, основанного на биологическом эксперименте по новокаледонским воронам. В этом случае внутренняя модель представляет собой базу знаний. База знаний – это таблица, характеризующая причинные связи между событиями во внешней среде и действиями агента. Используя базу знаний, агент формирует план целенаправленного поведения. Обсуждаются перспективы будущего анализа более совершенных внутренних моделей автономных агентов.

Ключевые слова: автономный агент, причинные связи, предсказания, внутренняя модель, база знаний.

1. Введение

Юджин Вигнер, лауреат Нобелевской премии по физике, в своей замечательной работе [1, 2] заострил внимание на проблеме «непостижимой эффективности математики в естественных науках». Он ярко изобразил проблемы применимости теорий, в особенности результатов теоретической физики, полученных на основе ограниченных фактов, для совершенно неожиданно широкого круга явлений. Согласно Ю. Вигнеру:

«Математический язык удивительно хорошо приспособлен для формулировки физических законов. Это чудесный дар, который мы не понимаем и которого не заслуживаем».

Следует подчеркнуть, что Ю. Вигнер также отметил важность исследований в области биологии и психологии для более полного понимания наших представлений о природе [3, 4]:

«Многие сегодня начинают ощущать, что мы слишком долго пренебрегали биологическими науками и науками о разуме человека и животных. Наша картина мира, несомненно, была бы более полной, если бы мы располагали более подробными сведениями о разуме людей и животных, их нравах и привычках».

Как же можно подойти к пониманию причин эффективности математики в естественных науках? Один из подходов к такому пониманию – исследование эволюции познавательных способностей биологических организмов, той эволюции, которая привела к мышлению, используемому в научном познании, в том числе к формальному логическому мышлению, используемому в математике. Исследование когнитивной эволюции целесообразно вести с помощью математических и компьютерных моделей [5, 6]. Исследование когнитивной эволюции может быть направлено на анализ эволюционных корней наших познавательных способностей, в том числе тех способностей, которые используются научном познании. Один из инструментов моделирования когнитивной эволюции – модели автономных агентов.

В настоящей работе анализируются подходы к изучению важного элемента когнитивных способностей биологических организмов – к анализу внутренних моделей внешнего мира. Такие внутренние модели являются предшественниками научных моделей, формируемых человеком.

В настоящей статье анализируются процессы формирования и использования внутренних моделей автономными агентами (модельными организмами). Рассмотрение ведется на конкретном примере, основанном на биологическом эксперименте по новокаледонским воронам. В этом примере внутренняя модель может быть представлена в виде базы знаний, характеризующей причинные связи между событиями во внешней среде и действиями агента. На основе такой базы знаний агент может формировать план целенаправленного поведения.

После изложения этого примера обсуждаются перспективы исследования более совершенных внутренних моделей.

2. Формирование и использование внутренних моделей новокаледонскими воронами

В работе [7] был проведен интересный биологический эксперимент с новокаледонскими воронами. В этом эксперименте вороны предварительно обучались выполнению отдельных элементов достаточно сложного поведения. После обучения воронам предлагалось решить следующую трехзвенную проблему:

1) подтянуть висящую на шнуре короткую палочку и освободить ее от шнура,

2) с помощью короткой палочки достать длинную палочку из первого зарешеченного контейнера,

3) длинной палочкой извлечь пищу из второго глубокого прозрачного контейнера.

При этом невозможно было извлечь пищу из глубокого контейнера с помощью короткой палочки или клюва и извлечь длинную палочку из зарешеченного контейнера клювом. Поэтому, чтобы добыть пищу, вороне надо было выполнить четко определенную цепочку указанных последовательных действий: 1→ 2 → 3.

Вороны предварительно обучались отдельным элементам решения трехзвенной проблемы. Были более полно обученные вороны, которые обучались оперированию со всеми тремя инструментами (шнур, короткая палочка и длинная палочка) по отдельности, и менее обученные вороны, которые обучались оперированию только с двумя из этих инструментов (шнур и длинная палочка). В результате более обученные вороны решали трехзвенную проблему с первой попытки; менее обученные вороны тоже решали эту проблему, но медленнее и не всегда с первой попытки.

Таким образом, вороны научились продумывать план решения новой задачи, мысленно связывая в плане ранее освоенный опыт.

В наших работах были построены две модели такого планирования новокаледонскими воронами [8, 9]. В первой модели предполагалось, что агент (модельная ворона) сначала анализирует результаты предварительного обучения, а потом, на основе этого анализа строит план решения проблемы в обратном направлении: от целевой ситуации к начальной [8]. Во второй модели предполагалось, что агент сразу мысленно стремится представить путь к цели, а потом проверяет этот путь путем обратного и прямого мысленного рассмотрения [9]. В обоих случаях агент стоит базу знаний, характеризующую ситуации, действия, прогнозы результатов действий, оценки расстояний между текущими ситуациями и целевой ситуацией. Для определенности остановимся на первой модели.

В биологическом эксперименте во время предварительного обучения воронам были представлены следующие шесть подзадач:

(1) извлечь пищу из глубокого прозрачного контейнера с помощью длинной палочки;

(2) извлечь длинную палочку из первого зарешеченного контейнера (один конец длинной палочки просунут между стержнями контейнера, так что ворона может клювом извлечь эту палочку) и извлечь пищу из второго глубокого прозрачного контейнера длинной палочкой;

(3) попытаться извлечь пищу из второго глубокого прозрачного контейнера с помощью короткой палочки (что было невозможно);

(4) подтянуть шнур и освободить пищу, висящую на шнуре и привязанную к концу шнура;

(5) извлечь длинную палочку из первого зарешеченного контейнера с помощью короткой палочки;

(6) подтянуть шнур с длинной палочкой, привязанной к концу шнура, освободить длинную палочку и извлечь пищу из второго глубокого прозрачного контейнера с помощью длинной палочки.

После предварительного обучения воронам предлагалось решить трехзвенную проблему. Исходная ситуация была такова: короткая палочка висит на шнуре и привязана к концу шнура, длинная палочка находится в первом зарешеченном контейнере, пища находится во втором глубоком прозрачном контейнере.

Основываясь на описанном биологическом эксперименте, мы построили модель формирования плана следующим образом. После обучения агент прогнозирует будущую ситуацию для данной текущей ситуации и текущего действия. Рассмотрим следующие ситуации (S_i) и действия (A_j):

S_1: короткая палочка привязана к концу шнура, длинная палочка находится в первом зарешеченном контейнере, пища находится во втором глубоком прозрачном контейнере

S_2: короткая палочка свободна, длинная палочка находится в первом зарешеченном контейнере, пища находится во втором глубоком прозрачном контейнере

S_3: длинная палочка свободна, пища находится во втором глубоком прозрачном контейнере

S_4: пища свободна

S_5: пища привязана к концу шнура

S_6: длинная палочка привязана к концу шнура; пища находится во втором глубоком прозрачном контейнере

S_7: длинная палочка частично извлечена из первого зарешеченного контейнера, пища находится во втором глубоком прозрачном контейнере

A_1: подтянуть шнур и освободить короткую палочку, привязанную к концу шнура

A_2: извлечь длинную палочку из первого зарешеченного контейнера с помощью короткой палочки

A_3: извлечь пищу из глубокого прозрачного контейнера с помощью длинной палочки

A_4: извлечь длинную палочку из первого зарешеченного контейнера (один конец длинной палочки был просунут между стержнями контейнера, так что ворона может клювом извлечь эту палочку)

A_5: попытаться извлечь пищу из второго глубокого прозрачного контейнера с помощью короткой палочки

A_6: подтянуть шнур и освободить пищу, привязанную к концу шнура

A_7: подтянуть шнур с длинной палочкой, привязанной к концу шнура, освободить длинную палочку.

После предварительного обучения агент может предсказать следующую ситуацию S_{next} для текущей ситуации и действия ($S_{current}$ и $A_{current}$:

$$\{S_{current}, A_{current}\} \rightarrow S_{next} \ .$$

Отметим, что этот прогноз аналогичен простой функциональной системе в теории функциональных систем П.К. Анохина [10]. Набор этих предсказаний представлен в таблице 1. Эти прогнозы соответствуют результатам предварительного обучения ворон. Подчеркнем, что S_1 – исходная ситуация, S_4 – целевая ситуация. Чтобы решить трехзвенную проблему, агент должен выполнить последовательность действий, которая приводит к переходу от ситуации S_1 к ситуации S_4.

Таблица 1. Результаты предварительного обучения

Номер предсказания	$S_{current}$	$A_{current}$	S_{next}
1	S_3	A_3	S_4
2	S_7	A_4	S_3
3	S_2	A_5	S_2
4	S_5	A_6	S_4
5	S_2	A_2	S_3
6	S_6	$A_7 + A_3$	S_4

Следует отметить, что согласно этой таблице агенты не могут выполнить существенное предсказание: $\{S_1, A_1\} \rightarrow S_2$. Однако это предсказание необходимо для решения трехзвенной проблемы.

Поэтому мы предполагаем, что агент угадывает мысленно (аналогично воронам) это предсказание. При этом полезна ассоциация с решением подзадач (4) и (6) при предварительном обучении. Используя прогнозы, показанные в таблице 1, агент начинает анализ способов достижения целевой ситуации S_4. Считаем, что агент изначально анализирует ситуации и действия, которые приводят к ситуации цели S_4. Ситуации S_5 и S_6 довольно далеки от исходной ситуации S_1.

Поэтому предсказания 4 и 6 (таблица 1) можно использовать только как ассоциативную помощь при анализе путей достижения цели.

Согласно предсказанию 1 достижение ситуации цели S_4 посредством одного действия возможно только из ситуации S_3 (длинная палочка свободна). Затем агент анализирует способы достижения ситуации S_3. Это возможно из ситуации S_2 (короткая палочка свободна). Далее агент догадывается до предсказания $\{S_1, A_1\} \rightarrow S_2$.

После угадывания этого прогноза агент получает схему достижения целевой ситуации S_4 из исходной ситуации S_1 (таблица 2).

Таблица 2. Результаты анализа достижения цели

Шаг	$S_{current}$	$A_{current}$	S_{next}	$\rho(S_{current}, S_4)$	$\rho(S_{next}, S_4)$
1	S_3	A_3	S_4	1	0
2	S_2	A_2	S_3	2	1
3	S_1	A_1	S_2	3	2

В этой таблице $\rho(S_{current}, S_4)$ / $\rho(S_{next}, S_4)$ – расстояние между ситуацией $S_{current}$ / S_{next} и ситуацией цели S_4; это расстояние – это количество действий, необходимых для достижения цели S_4 из рассматриваемой ситуации ($S_{current}$ или S_{next}). Таким образом, агент может мысленно представить последовательность ситуаций и действий, которые приводят к ситуации цели.

Используя результаты своего анализа, агент создает простую базу знаний, которая характеризует способ достижения целевой ситуации S_4 из исходной ситуации S_1 (таблица 3).

База данных знаний представляет собой простую реконструкцию результатов анализа, представленных в таблице 2.

Процесс реконструкции заключается в следующем. Агент начинает от исходной ситуации S_1 и выбирает действие, которое уменьшает расстояние ρ: $\rho(S_{current}, S_4) > \rho(S_{next}, S_4)$. Согласно таблице 2 это действие A_1. Следующая ситуация – S_2. Тогда, рассматривая S_2 как текущую ситуацию, агент аналогично выбирает следующее действие, которое уменьшает расстояние ρ; это действие A_2. Следующая ситуация – S_3. Наконец, в этой ситуации агент выбирает действие A_3; результатом этого действия является целевая ситуация S_4. Следует подчеркнуть, что в последнем процессе агент все время выбирает действия, которые уменьшают расстояние между рассматриваемыми ситуациями и целевой ситуацией S_4 (таблица 3).

Таблица 3. База знаний

Текущая ситуация, $S_{current}$	Текущее действие, $A_{current}$	Следующая ситуация, S_{next}	$\rho(S_{current}, S_4)$	$\rho(S_{next}, S_4)$
S_1	A_1	S_2	3	2
S_2	A_2	S_3	2	1
S_3	A_3	S_4	1	0

База данных знаний характеризует цепочку действий, в которой агент, начиная от исходной ситуации S_1, выполняет последовательные действия ($A_1 \rightarrow A_2 \rightarrow A_3$); эти действия приводят к целевой ситуации S_4; эта цепочка действий является планом поведения.

Заметим, что согласно таблице 2 для любой ситуации существует только одно действие, приводящее к уменьшению расстояния ρ. Однако в общем случае может быть несколько возможных полезных действий, любое из которых может уменьшить расстояние ρ. База знаний должна включать все эти полезные действия. Аналогичная задача с несколькими возможными полезными действиями для примера поведения рыб в лабиринтах была рассмотрена в работе [11]. В этом случае конкретное действие для выполнения в текущей ситуации может быть выбрано вероятностно, и происходит формирование нескольких возможных планов поведения.

Итак, представлена схема формирования планов поведения, основанная на прогнозах, полученных при предварительном обучении. Эта схема включает в себя следующее:

• Анализ ситуаций и действий, которые приводят к ситуации цели. Этот анализ соответствует мысленному представлению ситуаций в обратном направлении, от ситуации цели до исходной ситуации.

• Оценку близости к цели, формирование некоторой меры близости к цели, т. е. оценку расстояния между рассматриваемыми ситуациями и ситуацией цели.

• Формирование базы знаний, в которой описываются возможные пути достижения целевой ситуации из исходной ситуации. Этот процесс включает в себя сравнение расстояний до цели для разных действий и выбор действий, которые уменьшают это расстояние. База знаний соответствует прямому мысленному представлению, от исходной ситуации до цели.

• Формирование плана действий в соответствии с базой знаний.

В качестве меры близости к цели мы использовали количество действий, необходимых для достижения цели. Реальные жи-

вотные могут использовать и некоторое чувство близости к цели. Во всяком случае, новокаледонские вороны видели пищу в глубоком прозрачном контейнере [7]; поэтому у ворон было желание решить проблему и добраться до еды.

Кроме того, мы можем отметить, что вороны мысленно тратили некоторое время для формирования плана поведения [7]. Естественно предположить, что после нескольких первых решений проблемы ворона формирует стереотип поведения достижения цели. Затем ворона хранит этот стереотип в своей памяти и реализует его, когда она сталкивается с такой же проблемой в следующий раз. При стереотипном поведении не нужно тратить время на поиск решения, т.е. стереотипное поведение должно происходить быстро.

3. Обсуждение и заключение

Отметим, что в работе [9] была построена близкая к изложенной схема формирования планов новокаледонскими воронами. Основное отличие состоит в том, что в [9] предполагалось, что модельные вороны сразу делают прямое мысленное рассмотрение от исходной ситуации S_1 к целевой S_4 и догадываются до нужных предсказаний с определенными вероятностями. А после мысленного нахождения решения трехзвенной проблемы проверяют это решение путем обратного и прямого мысленного рассмотрения. Отметим, что аналогичное обратное и прямое мысленное проигрывание ситуаций наблюдалось экспериментально в работах [12, 13]. Например, когда крыса проходила коридор и в конце коридора находила пищу, то после этого она останавливалась и в клестах места гиппокампа крысы проигрывался путь к пище в обратном направлении [12]. Это указывает на то, что после нахождения полезного решения животные могут проверять путь к нахождению решения, проверять метод получения полезного результата. Результатом мысленного анализа в работе [9] также, как и в схеме, изложенной выше, становится база знаний, представленная в таблице 3.

В работах [8, 9] также было проведено компьютерное моделирование рассматриваемых процессов планирования при разумных предположениях о вероятностях угадывания воронами необходимых предсказаний. Результаты компьютерного моделирования качественно согласовывались с результатами биологического эксперимента [7].

База знаний (таблица 3) представляет внутреннюю модель автономного агента (модельной вороны) о закономерностях взаимодействия с внешней средой. Эта внутренняя модель характеризует причинные связи между ситуациями, действиями и результатами действий.

Отметим, что согласно нашей схеме формирования базы знаний агент делает простые правдоподобные выводы. Например, если цель достигается из ситуации S_3 (шаг 1 в таблице 2), то целесообразно найти способ достижения этой ситуации (шаг 2 в таблице 2). Конечно, это далеко от четких логических выводов в математике, но все же это реальные выводы, умозаключения.

Как же дальше развивать эти представления о внутренних моделях? Какие направления исследований внутренних моделей перспективны? Как от этих моделей перейти к научным моделям познания природы? Конечно, трудно сразу представить полноценную схему будущих исследований, но некоторые шаги можно наметить. Естественное обобщение рассмотренной схемы формирования внутренних моделей – анализ решения не отдельной проблемы, а совокупности проблем, связанных с потребностями организмов. Например, в схему поведения с несколькими естественными потребностями [14] достаточно несложно ввести выбор ведущей потребности в соответствии с принципом доминанты [15], а далее для каждой из потребностей вводить определенную базу знаний Еще одно важное направление, которое может быть использовано – введение самооценки своей деятельности самим агентом (см. например, рассуждения по этому поводу в работе [16]). При самооценке своего поведения агент может обучаться без учителя (для приведенного выше примера новокаледонских ворон учитель явно проводит предварительное обучение ворон). То есть возможно введение внутреннего учителя для автономного агента, учителя, оценивающего поведение агента.

Возвращаясь к вопросу о «непостижимой эффективности математики в естественных науках» [1, 2], можно отметить, что математические методы эффективно обобщают многочисленные результаты. И разумно анализировать способы обобщения внутренних моделей автономных агентов.

Настоящая работа выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП 14) по теме (проекту) «*35.16. Моделирование процессов формирования когнитивных способностей автономных агентов*» (№ 0065-2018-0002).

Internal models that are formed and used by autonomous agents

V.G. Red'ko

Abstract: The article analyzes the possibility of forming and using internal models of autonomous agents. We assume that the autonomous agent is a model organism. Initially we emphasize the importance of analyzing such internal models. Then, the process of formation and use of the internal model is considered, for a specific example that is based on a biological experiment on the New Caledonian crows. In this case, the internal model is a knowledge base. The knowledge database is a table characterizing the causal links between events in the external environment and agent actions. Using the knowledge database, the agent generates a plan for goal-directed behavior. Finally, the perspectives of the future investigations of more advanced internal models of autonomous agents are discussed.

Keywords: autonomous agents, predictions, causal relations, internal model, knowledge database, plan of goal-directed behavior.

Литература

1. E.Wigner. The unreasonable effectiveness of mathematics in natural sciences. "Communication on Pure and Applied Mathematics", vol. 13 (1960), No. 1, 1 – 14.
2. Е.Вигнер. Непостижимая эффективность математики в естественных науках. «Успехи физических наук», т. 94 (1968), № 3, 535 – 546.
3. E.Wigner. The limits of science. "Proceedings of the American Philosophical Society", vol. 94 (1950), No. 5, 422 – 427.
4. Е.Вигнер. «Этюды о симметрии». М.: Мир, 1971 (Раздел «Пределы науки»).
5. В.Г.Редько. «Моделирование когнитивной эволюции: На пути к теории эволюционного происхождения мышления». Изд. 2, испр. и доп. М.: URSS, 2018.
6. V.G.Red'ko. "Modeling of Cognitive Evolution. Toward the Theory of Evolutionary Origin of Human Thinking". Moscow: KRASAND/URRS, 2018.
7. A.H.Taylor, D.Elliffe, G.R.Hunt, R.D.Gray. Complex cognition and behavioural innovation in New Caledonian crows. "Proceedings of the Royal Society B", vol. 277 (2010), No. 1694, 2637 – 2643.
8. V.G.Red'ko, V.A.Nepomnyashchikh. Model of plan formation by New Caledonian crows. "Procedia Computer Science", vol. 71 (2015), 248 – 253.
URL: http://www.sciencedirect.com/science/article/pii/S1877050915036820
9. V.G.Red'ko, M.S.Burtsev. Modeling of mechanism of plan formation by New Caledonian crows. "Procedia Computer Science", vol. 88 (2016), 403 – 408.
URL: http://www.sciencedirect.com/science/article/pii/S1877050916317124
10. П.К.Анохин. «Очерки по физиологии функциональных систем». М.: Медицина, 1975.
11. V.G.Red'ko, V.A.Nepomnyashchikh, E.A.Osipova. Models of fish exploratory behavior in mazes. "Biologically Inspired Cognitive Architectures", vol. 15 (2015), 9 – 16.
12. D.J.Foster, A.Matthew, M.A.Wilson. Reverse replay of behavioural sequences in hippocampal place cells during the awake state. "Nature", vol. 440 (2006), No. 7084, 680 – 683.
13. K.Diba, G.Buzsaki. Forward and reverse hippocampal place-cell sequences during ripples. "Nature Neuroscience", vol. 10 (2007), No. 10, 1241 – 1242.
14. А.Г.Коваль, В.Г.Редько. Поведение модельных организмов, обладающих естественными потребностями и мотивациями. «Математическая биология и биоинформатика», т. 7 (2012), № 1, 266 – 273. URL: http://www.matbio.org/2012/Koval2012(7_266).pdf
15. А.А.Ухтомский. «Доминанта». СПб.: «Питер», 2002.
16. C.Finn, S.Levine. Deep visual foresight for planning robot motion.
URL: https://arxiv.org/abs/1610.00696v2

Система управления заданиями распределенной сети суперкомпьютерных центров коллективного пользования

Б.М.Шабанов[1], П.Н.Телегин[2], А.П.Овсянников[3], А.В.Баранов[4], А.И.Тихомиров[5], Д.С.Ляховец[6]

[1,2,3,4,5,5] *Межведомственный суперкомпьютерный центр РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия,* [6] *ФГУП «НИИ «Квант», Москва, Россия,*

E-mail's: [1]shabanov@jscc.ru, [2]ptelegin@jscc.ru, [3]apo@jscc.ru, [4]antbar@mail.ru, [5]tema4277@rambler.ru, [6]anetto@inbox.ru

Аннотация: В статье рассмотрены вопросы построения системы управления заданиями распределённой сети суперкомпьютерных научных центров. Авторами предлагается децентрализованная схема управления, основанная на асинхронном взаимодействии коллектива равноправных диспетчеров через глобальную очередь заданий. Рассмотрена практическая реализация глобальной очереди на базе распределённой СУБД Elasticsearch, предложены подходы к обеспечению защищённого доступа к распределённой базе данных и организации безопасного копирования исходных данных заданий между различными суперкомпьютерными центрами.

Ключевые слова: суперкомпьютерный центр, распределённая система управления заданиями, Elasticsearch, планирование заданий, очередь заданий.

Введение

Одним из действенных методов повышения доступности и эффективности использования ресурсов суперкомпьютерных центров коллективного пользования (СКЦ) является их объединение в единую распределённую сеть. Подобное объединение даёт возможность оперативного перераспределения нагрузки между ресурсами путём перенаправления пользовательских заданий из очереди одного СКЦ в очередь другого, менее загруженного в определённый момент времени. Активные работы над проектом единой распределённой сети научных суперкомпьютерных центров в настоящее время ведутся в Межведомственном суперкомпьютерном центре РАН (МСЦ РАН) [1].

Согласно проекту [1], принадлежащие разным организациям СКЦ объединяются друг с другом посредством закрытых каналов связи, как показано на рисунке 1. Для получения машинного времени для расчётов в СКЦ пользователь должен оформить т.н. вычислительное **задание**, включающее в себя прикладную расчётную программу, требования к ресурсам и исходные данные. Минимальные требования к ресурсам включают необходимые для расчётов число процессорных ядер и время выполнения прикладной программы. В каждый из СКЦ поступает свой поток пользовательских заданий, часть из которых может быть выполнена непосредственно на ресурсах принявшего задание СКЦ, а другая часть – перенаправлена в глобальную очередь, доступную всем СКЦ сети. Хранение глобальной очереди осуществляется в специализированной информационной системе, построенной на базе распределённой СУБД [2].

Одной из ключевых задач проекта является создание децентрализованной автоматизированной системы управления заданиями и ресурсами в распределённой сети СКЦ. Система управления за счёт поддержки глобальной очереди пользовательских заданий должна обеспечить оперативное перераспределение вычислительной нагрузки в сети СКЦ. В публикациях [1, 3] авторами обозначены подходы к общей организации подобной системы управления, согласно которым в системе выделяется два уровня управления – локальный и глобальный. Локальный уровень представлен системами управления заданиями (СУЗ) [4] отдельной суперкомпьютерной установки, такими как SLURM [5], PBS [6], Moab [7] или отечественная система управления прохождением параллельных заданий (СУППЗ) [8], на локальном уровне осуществляется распределение заданий в рамках одной отдельной суперкомпьютерной установки или её раздела с отдельной локальной очередью заданий. На глобальном уровне осуществляется распределение заданий между суперкомпьютерными

установками, входящими в состав сети СКЦ. Распределение происходит посредством глобальной очереди заданий, методы и средства организации которой рассмотрены в работе [2]. Методы и алгоритмы распределения заданий глобальной очереди для случая абсолютных приоритетов заданий исследовались авторами в работах [9-11]. В настоящей статье представлены научно-технические аспекты практической реализации системы управления.

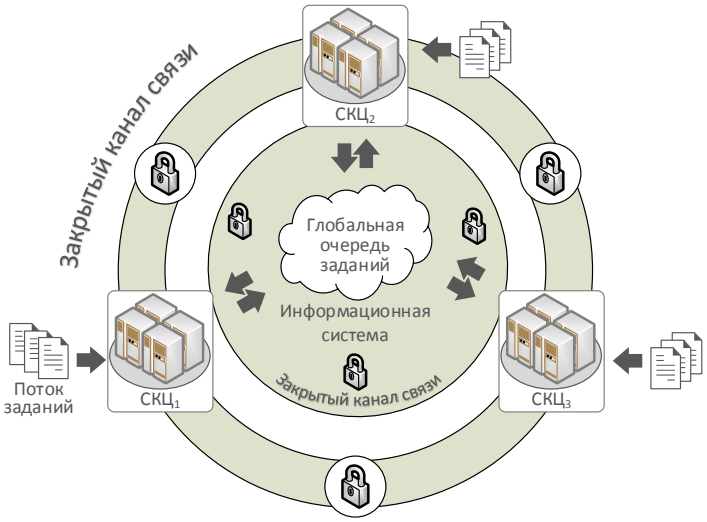


Рисунок 1. Схема распределенной сети суперкомпьютерных центров коллективного пользования

Среди таких аспектов следует выделить следующие задачи. Во-первых, необходимо определить компоненты и разработать структуру создаваемой системы управления. Во-вторых, при всех достоинствах распределённой СУБД Elasticserach, выбранной в работе [2] в качестве основы для реализации информационной системы для хранения глобальной очереди, её слабым местом является отсутствие авторизации пользователей и, как следствие, – необходимость построения системы защиты доступа к информационной системе. Во-третьих, необходимо организовать безопасное копирование данных между СКЦ с учётом различия систем авторизации пользователей и политик безопасности в разных СКЦ.

Иерархическая организация распределенной сети СКЦ

Единицей управления, включаемой в состав распределённой сети СКЦ, является вычислительная установка – суперкомпьютер или его раздел, управляемый локальной СУЗ. В состав каждой вычислительной установки (ВУ) входят:

– сервер доступа;
– управляющий сервер;
– решающее поле.

ВУ характеризуется:

– логическим именем, уникальным в рамках отдельного СКЦ;
– списков вычислительных модулей (узлов), водящих её в состав;
– доменным именем и номером порта сервера доступа;
– доменным именем и номером порта сервера управления.

Решающее поле включает в себя несколько вычислительных модулей (ВМ), объединённых высокоскоростной коммуникационной сетью. Каждый ВМ уникально именован в рамках своей ВУ.

Заметим, что вычислительная установка может являться как отдельным суперкомпьютером, так и его разделом. Управление ВУ может осуществлять отдельно выделенный экземпляр СУЗ, но допускается ситуация, когда СУЗ управляет несколькими разделами одновременно. СУЗ функционирует на сервере управления ВУ и выполняет следующие основные функции:

– приём входного локального потока пользовательских заданий;
– ведение локальной очереди заданий;
– выделение вычислительных ресурсов ВУ (подмножества ВМ решающего поля), требуемых для выполнения прошедшего через очередь задания;
– контроль выполнения задания;
– освобождение занятых заданием ВМ после его завершения.

С точки зрения организации сети СКЦ важно, что ВУ уникально именуется в рамках СКЦ и обладает собственной локальной очередью пользовательских заданий. Все ВУ из состава распределённой сети СКЦ образуют нижний уровень её иерархии.

Следующий – средний – уровень иерархии образуют СКЦ, в состав каждого из которых могут входить одна или несколько вычислительных установок с разными именами. Каждый СКЦ внутри распределённой сети характеризуется:

– уникальным именем;

– организационной принадлежностью и территориальным расположением;

– списком ВУ, входящих в состав СКЦ;

– отдельной системой хранения данных СКЦ (СХД СКЦ), содержащей проектные каталоги, исходные данные и результаты расчетов всех пользователей;

– отдельной системой авторизации пользователей СКЦ (САП СКЦ), хранящей учётные записи всех пользователей СКЦ.

Сеть СКЦ на верхнем уровне иерархии объединяет вычислительные установки нескольких СКЦ, связанных коммуникационными каналами. Сеть СКЦ характеризуется списком СКЦ, входящих в её состав, и единой для всех СКЦ глобальной очередью заданий. Назначение глобальной очереди состоит в сокращении времени обработки задания (т.е. времени с момента поступления задания в глобальную очередь до момента завершения его выполнения на вычислительных ресурсах) за счёт помещения задания в наименее загруженную ВУ из состава сети СКЦ.

Для примера на рисунке 2 изображена сеть из двух СКЦ с условными именами JSCC и SSCC (здесь и далее имена СКЦ будут записываться прописными буквами). В состав СКЦ JSCC входит вычислительная установка с именем Broadwell, в состав СКЦ SSCC – две ВУ с именами Slurmwell и Haswell.

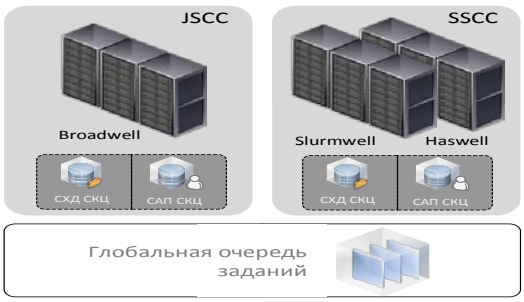


Рисунок 2. Иерархическая организации сети СКЦ

Структура и порядок функционирования глобальной системы управления заданиями и вычислительными ресурсами сети СКЦ

Управление заданиями и вычислительными ресурсами сети СКЦ строится по децентрализованной схеме вокруг глобальной очереди заданий, как показано на рисунке 3. В глобальную очередь задания помещаются пользователями с помощью специальной команды пользовательского интерфейса. Для возможности выполнения этой команды пользователь должен быть соединён с любой ВУ из состава СКЦ. ВУ, с которой пользователь отправил задание в глобальную очередь, будем называть **ВУ-заказчиком**.

В составе СКЦ выделяется специальный компонент – **диспетчер СКЦ**, функционирующий на отдельном виртуальном или физическом сервере. Диспетчер СКЦ обеспечивает, с одной стороны защиту информационной системы, с другой стороны – командные интерфейсы пользователя и администратора. Через диспетчер СКЦ выполняются любые команды администратора и пользователя, в т.ч. команда помещения задания в глобальную очередь.

Задания из глобальной очереди распределяются коллективом равноправных диспетчеров. Диспетчер вычислительной установки (**диспетчер ВУ**) – это управляющий процесс, ответственный за распределение заданий из глобальной очереди в локальную очередь своей ВУ и за последующую обработку этих заданий. Здесь и далее вычислительное задание, распределённое из глобальной очереди в одну из ВУ, будем называть **глобальным заданием**, а задания, поступающие сразу в очередь отдельной ВУ – **локальными**. ВУ, диспетчер которой выбрал глобальное задание для выполнения, будем называть **ВУ-исполнителем**.

В СКЦ, содержащем в своём составе несколько ВУ, одновременно может функционировать несколько диспетчеров ВУ (по одному на каждую ВУ). Если некоторая СУЗ управляет несколькими разделами ВУ (т.е. ведёт несколько локальных очередей заданий), то в этом случае допускается функционирование нескольких диспетчеров ВУ на одном управляющем сервере (для каждого раздела своя очередь и свой диспетчер). Иными словами, каждый диспетчер ВУ «привязывается» к своей локальной очереди заданий.

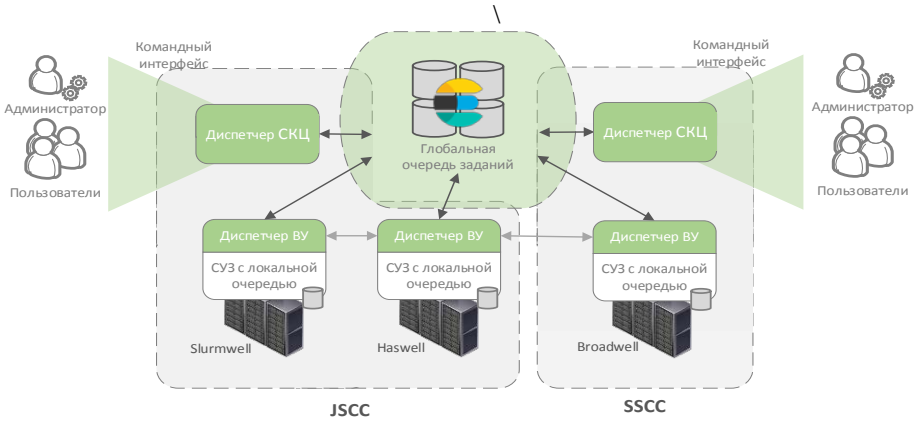


Рисунок 3. Структура системы управления вычислительными ресурсами сети СКЦ

Диспетчер ВУ выполняет следующие основные функции:

– проверка состояния своей локальной очереди с целью определения возможности размещения новых заданий глобальной очереди;

– извлечение задания из глобальной очереди и его помещение в свою локальную очередь под контроль своей СУЗ;

– копирование исходных данных задания с ВУ-заказчика на ВУ-исполнитель;

– контроль состояния задания на ВУ-исполнителе;

– копирование результатов выполнения задания с ВУ-исполнителя на ВУ-заказчик.

В процессе работы диспетчер ВУ взаимодействует со своей локальной СУЗ через её командный пользовательский интерфейс.

Частота обращений диспетчера к СУЗ задается администратором. При каждом обращении к СУЗ диспетчер ВУ выполняет проверку состояния ранее распределённых глобальных заданий и принимает решение о распределении нового глобального задания в локальную очередь ВУ. Каждое глобальное задание может иметь следующие статусы, отражающие этапы его обработки:

– «находится в глобальной очереди»: задание помещено пользователем в глобальную очередь, но пока не распределено ни в одну ВУ-исполнитель;

– «копируются исходные данные»: задание распределено в ВУ-исполнитель, диспетчер которой выполняет копирование исходных данных задания перед помещением его в свою локальную очередь;

– «находится в локальной очереди»: исходные данные перемещены на ВУ-исполнитель, задание ожидает освобождения вычислительных ресурсов в локальной очереди СУЗ ВУ-исполнителя;

– «выполняется»: задание размещено на решающем поле ВУ-исполнителя;

– «завершено»: задание выполнилось и освободило занимаемые вычислительные ресурсы;

– «ошибка выполнения»: произошла ошибка на любом из этапов обработки задания.

Все изменения состояния выполнения заданий диспетчер ВУ отражает в глобальной очереди.

При распределении нового глобального задания диспетчер ВУ соблюдает заданное администратором СКЦ ограничение по максимальному числу глобальных заданий в локальной очереди. Если это ограничение достигнуто, диспетчер ВУ перестаёт распределять задание глобальной очереди в свою ВУ до тех пор, пока часть ранее распределённых глобальных заданий не будет завершена. Решение о распределении заданий принимается диспетчером не только на основе числа количества заданий в локальной очереди ВУ, а также с учётом следующих факторов:

– наличие учётной записи пользователя в СКЦ;

– доступные квоты пользователя;

– расположение исходных данных заданий;

– наличие на ВУ необходимого инструментального и прикладного ПО.

В отличие от диспетчера ВУ, в каждом СКЦ должен функционировать только один диспетчер СКЦ, который не привязан к управляющему серверу какой-либо ВУ и может размещаться на любом из серверов СКЦ, в том числе выделенном или виртуальном. В процессе функционирования диспетчер СКЦ взаимодействует только с распределённой информационной системой [2], хранящей глобальную очередь заданий, и не может напрямую взаимодействовать с

другими диспетчерами СКЦ или диспетчерами ВУ. При необходимости взаимодействие с этими компонентами осуществляется через глобальную очередь заданий.

Основными функциями диспетчера СКЦ являются обеспечение защиты информационной системы через авторизацию пользователей и предоставление пользовательского и административного командных интерфейсов системы управления.

Пользовательский командный интерфейс предоставляет пользователю следующие возможности:

– добавлять новые глобальные задания;

– отслеживать состояния глобальных заданий;

– получать результаты выполнения глобальных заданий;

– просматривать конфигурацию сети СКЦ;

– проверять конфигурацию пользовательских настроек.

Административный командный интерфейс включает в себя следующие команды:

– добавление СКЦ (ВУ) в состав сети и удаление СКЦ (ВУ) из неё;

– добавление/удаление пользователя.

Диспетчер СКЦ реализован в виде http-сервера, поэтому каждая команда пользовательского или административного интерфейсов содержит http-запрос к диспетчеру СКЦ.

Организация глобальной очереди заданий

Для хранения глобальной очереди и информации, необходимой для работы коллектива диспетчеров, в соответствии с результатами исследования [2] применяется Elasticsearch [12] – распределённая нереляционная система управления базами данных (РСУБД) с открытым исходным кодом. РСУБД Elasticsearch разработана на языке Java и доступна для многих платформ. Информация в Elasticsearch хранится в формате JSON-документов. Структуру документа в Elasticsaerch рекомендуется определять заранее, а документы с разной структурой размещать в разных индексах. По аналогии с реляционными СУБД, индекс – это таблица, которая используется для хранения набора документов.

РСУБД Elasticsearch представляет собой систему управления БД-кластером из нескольких серверов хранения данных, которые могут быть территориально удалены друг от друга, но при этом должны находиться в одной сети.

Это условие может быть выполнено за счёт использования виртуальных сетей (VLAN) или туннелей (VPN).

Сервер хранения РСУБД Elasticsearch должен быть размещён в каждом СКЦ.

Как показано на рисунке 4, в распределённой сети СКЦ должны быть организованы две виртуальных сети – для функционирования и взаимодействия серверов РСУБД Elasticsearch (сеть $VLAN_E$) и для взаимодействия диспетчеров ВУ (сеть $VLAN_D$).

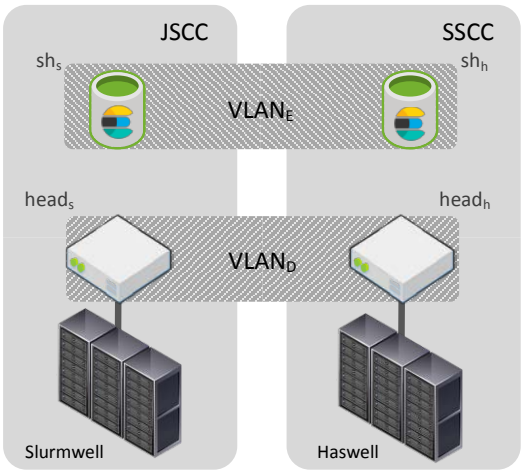


Рисунок 4. Защищённые виртуальные сети для взаимодействия серверов Elasticsearch и диспетчеров ВУ

При необходимости к уже имеющемуся БД-кластеру можно добавлять новые серверы хранения данных, и РСУБД Elasticsearch станет автоматически распределять на них данные таким образом, что при отказе любого сервера хранения данные не будут утеряны.

Такая надёжность достигается за счёт использования механизмов реплицирования и шардирования данных [2].

Elasticsearch управляется по протоколу HTTP с помощью запросов в формате JSON. Управлять БД-кластером можно с любого сервера управления из состава сети СКЦ.

Для обеспечения функций информационной подсистемы системы управления в БД Elasticsearch было выделено три индекса (аналоги таблиц в реляционных БД):

– индекс хранения глобальной очереди;

– индекс хранения информации о пользователях;

– индекс хранения информации о конфигурации системы управления (сети СКЦ).

В индексе глобальной очереди хранится информация о всех заданиях, добавленных пользователями в глобальную очередь. Диспетчер ВУ-исполнителя в этом индексе периодически обновляет статус глобального задания в процессе его обработки.

В индексе, содержащем информацию информации о пользователях сети СКЦ, содержится таблица соответствия имён входа (т.н. логинов) пользователя на разных ВУ. Информация в индекс записывается в процессе регистрации пользователя на одной из ВУ, осуществляемой администратором.

При регистрации формируется глобальный идентификатор пользователя – уникальная 16-байтная буквенно-цифровая последовательность. Глобальный идентификатор формируется для пользователя один раз при первой регистрации пользователя на одной из ВУ. В дальнейшем при регистрации пользователя на других ВУ администратор использует ранее назначенный глобальный идентификатор.

Задание из глобальной очереди могут быть распределены только на те ВУ, на которых пользователь имеет учётные записи. Каждый пользователь должен быть ассоциирован минимум с одной ВУ. В случае, если пользователь ассоциирован только с одной ВУ, её удаление приведет к удалению пользователя.

В каждый момент времени пользователь функционирует в сети СКЦ под уникальным глобальным идентификатором, привязанным к конкретным ВУ и СКЦ.

Например, если пользователь авторизовался в СКЦ JSCC на ВУ с именем broadwell под логином ivan, то глобальный идентификатор пользователя будет ivan@broadwell.jscc, если позже этот же пользователь авторизовался в том же СКЦ на ВУ slurmwell под логином ivan_1, то глобальный идентификатор пользователя будет ivan_1@slurmwell.jscc. При этом диспетчеры системы управления, используя индекс пользователей, смогут определить, что это один и тот же пользователь.

В индексе хранения информации о конфигурации системы содержится описание всех вычислительных ресурсов, доступных для заданий глобальной очереди. Эту информацию в индекс заносит администратор. В процессе работы сети СКЦ администратор СКЦ должен своевременно отмечать недоступность отдельных ВУ, соответствующим образом изменяя их статус. Информация этого индекса используется диспетчерами ВУ при распределении вычислительных заданий.

Взаимодействие диспетчеров ВУ при копировании исходных данных задания

Как уже отмечалось, основную функцию управления – распределение глобальных заданий – коллектив равноправных диспетчеров ВУ осуществляет через асинхронные обращения к глобальной очереди заданий. Необходимость непосредственного взаимодействия диспетчеров разных ВУ возникает на этапе копирования исходных данных задания с ВУ-заказчика на ВУ-исполнитель. Основной проблемой при копировании данных из одного СКЦ в другой является различие систем авторизации пользователей в разных СКЦ и, как следствие, – разные учётные записи пользователей и разные политики администрирования. При этом в децентрализованной распределённой сети администраторы разных СКЦ не могут и не должны доверять друг другу администраторские права (суперпользовательские привилегии) на доступ к своим суперкомпьютерным ресурсам.

Тем не менее, при распределении глобального задания копировать его исходные данные на ВУ-исполнитель необходимо, а при копировании, в свою очередь, необходимо проводить авторизацию пользователя в САП СКЦ-исполнителя. Поскольку время начала копирования данных заранее неизвестно, авторизация самим пользователем с помощью пароля невозможна. В настоящей работе предлагается подход, основанный на доверии между собой процессов-диспетчеров ВУ и на предположении, что их взаимодействие будет производиться в надёжно защищённой виртуальной сети ($VLAN_D$ на рисунке 4).

Отметим, что в глобальную очередь помещается только паспорт задания, включающий идентификационную информацию и требования к ресурсам. Исходные данные задания остаются в ВУ-заказчике до момента, пока задание не будет извлечено из глобальной очереди диспетчером ВУ-исполнителя. После извлечения глобального задания диспетчер ВУ-исполнителя организует процесс копирования исходных данных задания из СХД СКЦ ВУ-заказчика.

Для копирования данных между диспетчерами ВУ разных СКЦ используется системный пользователь dispatcher с правами суперпользователя и с возможностью беспарольного выполнения команды копирования и синхронизации файлов rsync. Для этого пользователя в СХД СКЦ должен

быть настроен **служебный каталог** для данных (назовём $DISP), доступный всем пользователям на запись. В этот каталог от имени пользователя копируются исходные данные задания.

Порядок копирования исходных данных задания следующий:

– процесс копирования с ВУ-исполнителя отправляет запрос процессу на ВУ-заказчике на получения исходных данных задания;

– диспетчер ВУ-заказчика с правами пользователя копирует исходные данные из проектного пользователя в служебный каталог;

– из служебного каталога ВУ-заказчика данные копируются в служебный каталог ВУ-исполнителя. По окончании копирования диспетчер ВУ-заказчика уведомляет диспетчера ВУ-исполнителя о готовности данных;

– диспетчер ВУ-исполнителя с правами пользователя перемещает задание из служебного каталога в проектный каталог пользователя, в котором заранее подготавливается директория с именем, совпадающем с глобальным идентификатором задания;

– диспетчер ВУ-исполнителя добавляет новое глобальное задание в локальную очередь СУЗ ВУ-исполнителя.

Пример, демонстрирующий порядок копирования исходных данных задания, представлен на рисунке 5. Диспетчер ВУ с именем broadwell суперкомпьютерного центра SSCC выбрал из глобальной очереди заданий паспорт задания с идентификатором jscc_17. Пусть в паспорте задания указано, что его владельцем является пользователь user@slurmwell.jscc, следовательно, исходные данные размещаются в суперкомпьютерном центре JSCC. Будем считать, что пользователь добавивший задание в глобальную очередь, на ВУ-заказчике имеет имя учётной записи ubu1, а на ВУ-исполнителе – ubu2.

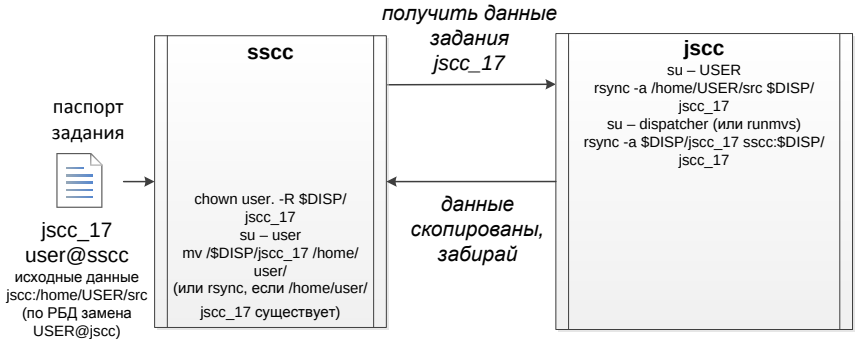


Рисунок 5. Иллюстрация копирования данных между диспетчерами ВУ СКЦ SSCC и JSCC

Диспетчер ВУ broadwell СКЦ SSCC отправляет диспетчеру ВУ slurmwell СКЦ JSCC запрос на получение исходных данных задания jscc_17. Диспетчер ВУ slurmwell с правами пользователя USER копирует данные задания в служебный каталог $DISP. Из каталога $DISP диспетчер, выполняя в беспарольном режиме команду rsync, копирует данные на ВУ-исполнитель в служебный каталог СКЦ SSCC. Напомним, что беспарольное выполнение команды копирования и синхронизации файлов rsync должно происходить внутри защищённой виртуальной сети $VLAN_D$ (см. рис. 4). После копирования диспетчер ВУ slurmwell СКЦ JSCC уведомляет диспетчер ВУ-исполнителя из СКЦ SSCC о завершении копирования данных. Диспетчер ВУ Broadwell СКЦ SSCC изменяет владельца данных скопированного задания на пользователя-владельца задания и с правами пользователя USER перемещает исходные данные в проектный каталог пользователя.

Отметим, что при размещении ВУ-исполнителя и ВУ-заказчика в одном СКЦ исходные данные заданий не копируются.

После копирования исходных данных и добавления задания в локальную очередь диспетчер ВУ-исполнителя выполняет трансляцию локального идентификатора задания в глобальный и наоборот. Глобальный идентификатор задания формируется на этапе добавления задания в глобальную очередь, а локальный идентификатор – при добавлении задания в локальную очередь СУЗ.

Пользователю доступен только глобальный идентификатор, с помощью которого он управляет заданием –проверяет состояние выполнения и получает результаты его выполнения задания.

Заключение

В статье рассмотрена разработанная авторами система управления заданиями создаваемой в МСЦ РАН распределённой сети суперкомпьютерных центров коллективного пользования. Предложена иерархическая организация системы управления, разработана децентрализованная схема управления коллективом равноправных диспетчеров, разработана и реализована в распределённой СУБД Elasticsearch информационная система для хранения глобальной очереди заданий, организована защита информационной системы через специально разработанную подсистему диспетчеров СКЦ, на которые возложены функции авторизации пользователей и поддержки командных интерфейсов пользователя и администратора. Разработана и реализована схема взаимодействия диспетчеров для организации безопасного копирования исходных данных задания и результатов его выполнения между различными СКЦ.

Предложенные авторами подходы были реализованы в виде макета системы управления заданиями распределённой сети СКЦ, развёрнутого в МСЦ РАН на действующих разделах суперкомпьютера МВС-10П ОП.

В настоящее время макет проходит опытную эксплуатацию, по завершении которой предполагается включение в состав распределённой системы ряда научных и образовательных СКЦ.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований по теме (проекту) «Исследование и разработка методов сетевой интеграции ресурсов и сервисов научных организаций» (0065-2018-0404).

The Jobs Management System for the Distributed Network of the Supercomputer Centers

B.M.Shabanov, P.N.Telegin, A.P.Ovsyannikov, A.V.Baranov,

D.S.Lyakhovets, A.I.Tikhomirov

Abstract. The article discusses issues of building jobs management system for distributed network of supercomputer research centers. The authors suggest a decentralized management scheme based on asynchronous interaction of a team of equal dispatchers via the global jobs queue. Implementation of the global queue on the base of the Elasticsearch distributed database is considered. There are proposed approaches that ensure secure access to a distributed database and secure transfer of input data between supercomputer centers.

Keywords: supercomputer center, distributed jobs management system, Elasticsearch, jobs scheduling, jobs queue.

Литература

1. Б.М.Шабанов, А.П.Овсянников, А.В.Баранов, С.А.Лещев, Б.В.Долгов, Д.Ю.Дербышев. Проект распределенной сети суперкомпьютерных центров коллективного пользования // Программные системы: теория и приложения. 2017. № 4(35). С. 245–262. DOI: 10.25209/2079-3316-2017-8-4-245-262

2. А.В.Баранов, А.И.Тихомиров. Методы и средства организации глобальной очереди заданий в территориально распределенной вычислительной системе // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2017. Т. 6. № 4. С. 28-42. DOI: 10.14529/cmse170403

3. А.В.Баранов, А.И.Тихомиров. Планирование заданий в территориально распределенной системе с абсолютными приоритетами //Вычислительные технологии. 2017. Т. 22, № S1. С. 4–12.

4. А.В.Баранов, А.В.Киселёв, В.В.Старичков, Р.П.Ионин, Д.С.Ляховец. Сравнение систем пакетной обработки с точки зрения организации промышленного счета. Научный сервис в сети Интернет: поиск новых решений // Труды Международной суперкомпьютерной конференции (Новороссийск, 17-22 сентября 2012 г.). М.: Изд-во МГУ, 2012. С. 506–508.

5. A.B.Yoo, M.A.Jette, M.Grondona. (2003) SLURM: Simple Linux Utility for Resource Management. In: Feitelson D., Rudolph L., Schwiegelshohn U. (eds) Job Scheduling Strategies for Parallel Processing. JSSPP 2003. Lecture Notes in Computer Science, vol 2862. Springer, Berlin, Heidelberg. pp. 44-60. DOI: 10.1007/10968987_3

6. R.L.Henderson. (1995) Job scheduling under the Portable Batch System. In: Feitelson D.G., Rudolph L. (eds) Job Scheduling Strategies for Parallel Processing. JSSPP 1995. Lecture Notes in Computer Science, vol 949. Springer, Berlin, Heidelberg. pp. 279-294. DOI: 10.1007/3-540-60153-8_34

7. Moab HPC Suite Enterprise Edition. URL: http://www.adaptivecomputing.com/products/hpc-products/moab-hpc-suite-enterprise-edition (дата обращения 12.07.2018)

8. Система управления прохождением параллельных заданий (СУППЗ). Руководство программиста (пользователя). URL: http://www.jscc.ru/wp-content/uploads/2017/06/SUPPZ-user-guide-2016.pdf (дата обращения 23.08.2018)

9. A.Baranov, P.Telegin, A.Tikhomirov. Comparison of Auction Methods for Job Scheduling with Absolute Priorities // Lecture Notes in Computer Science, 2017, vol. 10421. pp. 387-395. DOI: 10.1007/978-3-319-62932-2_37

10. B.M.Shabanov, P.N.Telegin, O.S.Aladyshev, A.V.Baranov, A.I.Tikhomirov. Comparison of Priority-Based and First Price Sealed-Bid Auction Algorithms of Job Scheduling in a Geographically-Distributed Computing System. Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering, ElConRus 2018. P. 1557-1562. DOI: 10.1109/EIConRus.2018.8317396

11. А.В.Баранов, В.В.Молоканов, П.Н.Телегин, А.И.Тихомиров. Применение метода английского аукциона при планировании заданий с абсолютными приоритетами в распределенной вычислительной системе // Программные продукты и системы. 2018. № 3. С. 461-468. DOI: 10.15827/0236-235X.123.461-468

12. Elastic Stack and Product Documentation [Электронный ресурс], 2018 / URL: https://www.elastic.co/guide/index.html (дата обращения 10.11.2018)

Производительность современных вычислительных платформ при обработке данных расчетов молекулярной динамики мембранных и белок-мембранных систем

Н.А. Крылов[1], Д.Е. Нольде[2], П.Н. Телегин[3], Р.Г. Ефремов[4], Б.М. Шабанов[5]

[1,3] МСЦ РАН - филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

[2,4] ФГБНУ Институт биоорганической химии имени академиков М.М. Шемякина и Ю.А. Овчинникова РАН, Москва, Россия, [5] ФГУ ФНЦ НИИСИ РАН, Москва, Россия

E-mail's: [1] krylovna@gmail.com, [2] nolde@nmr.ru, [3] telegin@jscc.ru, [4] efremov@nmr.ru, [5]shabanov@jscc.ru

Аннотация: Проведено исследование производительности 2-х алгоритмов обработки результатов молекулярной динамики (МД): расчета радиальной функции распределения (RDF) и расчета энергии на современных вычислительных платформах. Показано, что оба алгоритма эффективно распараллеливаются как на системах с общей памятью, так и на кластерах с распределенной памятью. Для обработки результатов МД систем среднего размера эффективность распараллеливания расчета RDF близка к 1 в диапазоне до 100 ядер, а при расчете энергии – до 500 ядер. Показано, что при расчете энергии при небольшом числе ядер выгоднее использовать параллелизацию по индексам атомов, а при большом числе ядер – по объему.

Ключевые слова: молекулярная динамика, параллельные вычисления, высокопроизводительные кластеры, MPI, OpenMP, энергия невалентных взаимодействий, радиальная функция распределения

1. Введение

Метод молекулярной динамики (МД) является важным инструментом атомистического моделирования биомолекулярных систем большого размера и вносит заметный вклад в развитие физико-химической биологии и биотехнологии. В предыдущей работе [1] мы подробно исследовали скорость расчета МД с помощью программного пакета Gromacs [2, 3] на широком классе вычислительных систем: от настольных компьютеров до компьютерных кластеров. При этом за рамками работы остался вопрос о производительности программного обеспечения для анализа результатов МД. Методы анализа МД, как правило, требуют меньших ресурсов, чем сам расчет МД, но при этом они хуже оптимизированы. Методы анализа можно условно разделить на 3 типа по затратам ресурсов на анализ одного состояния системы. К первому типу относятся методы, в которых затраты времени на анализ одного состояния несильно превосходят затраты на чтение состояния из файла траектории, например: расчет среднеквадратичного отклонения положения атомов (RMSD), расчет радиуса гирации молекулы и другие. Затраты времени на анализ всей траектории такими программами невелики даже при использовании настольных компьютеров в однопроцессорном режиме. Ко второму типу относятся задачи со значительными затратами времени на анализ одного состояния системы, но которые, в тоже время, позволяют провести анализ всей записанной траектории МД (как правило 10^4 – 10^5 состояний). К таким задачам относятся анализ различных энергетических термов, а также методы, основанные на анализе попарных расстояний в системах. К третьему типу относятся очень ресурсоемкие методы анализа, такие как построение поверхностей и расчет свойств на ней, качественная визуализация молекул с хорошей детализацией. Такие расчеты проводятся для небольшого набора состояний (~10^2). Самыми интересными с точки зрения оптимизации алгоритма являются задачи второго типа. При этом в пакетах программ молекулярной динамики подобные программы обычно присутствуют, но при этом плохо оптимизированы под многопроцессорные системы. В настоящей работе исследовалась производительность и масштабируемость алгоритмов расчета радиальной функции распределения (RDF) и расчета энергии невалентных взаимодействий на различных вычислительных системах (см. Таблицу 1).

2. Методы

Производительность оценивали, измеряя время выполнения расчета для одного состояния системы. Масштабируемость оценивали, вычисляя коэффициент ускорения, равный отношению времени выполнения расчета на одном вычислительном ядре ко времени выполнения на всех ядрах. Расчет выполнялся 3 раза и включал в себя обработку 5 состояний системы. Во всех случаях время выполнения одной итерации усреднялось по запускам и состояниям. Тестирование производительности расчета сил и энергий проводилось на той же системе, что и в работах [1, 4]: белок TRPV1 в гидратированном липидном бислое. Для расчета RDF мы взяли более простую систему небольшого размера, состоящую из 288 молекул липидов и 11883 молекул воды. Такие системы часто используют для оценки параметров гидратации гидрофильных групп, входящих в состав липидов и сравнения данных расчета и эксперимента. RDF строилась для всех пар кислородов фосфатной группы липида и кислородов молекул воды. Расчет RDF выполнялся 5 раз и включал в себя обработку 10 состояний системы. Разброс значений во всех расчетах составил не более 2%.

3. Расчет RDF

Расчет RDF сводится к построению гистограммы всех парных расстояний между двумя заданными группами атомов. Алгоритм относительно простой, но ресурсоемкий, так как требует $\sim N^2$ операций (где N – число атомов). При этом итерации цикла независимы, а доступ к ячейкам памяти последовательный, и можно предположить, что алгоритм масштабируется линейно. Однако объем вычислений на каждой итерации небольшой и времени их выполнения не хватает для компенсации задержек при обращении к памяти. Применение режима динамического распределения итераций по процессам частично компенсирует, но не решает указанную проблему. Поэтому, когда число процессов, обрабатывающих данные, достигает 4-8, ширины каналов доступа к памяти не хватает, и коэффициент ускорения начинает снижаться, уменьшаясь до ~2/3 от максимально возможного при использовании 32 ядер (см. Таблицу 2). Использование для расчета кластеров позволяет одновременно увеличить число каналов доступа к памяти и уменьшить объем вычислений. В результате удается получить эффективное распараллеливание вплоть до ~128 MPI процессов (рис. 1). Таким образом для расчета RDF выгоднее использовать MPI и распределять задачу на несколько узлов, задействую 8-16 ядер на каждом, следя за тем, чтобы на каждое ядро приходилось не менее 300-400 пар атомов.

4. Расчет энергии и сил

Вычисления вклада в энергию и силу от короткодействующих невалентных взаимодействий сводиться к вычислению функций вида R^{-n}, где n=1, 2, 5, 6, 11, 12, а R – расстояние между парами атомов, находящимися на расстояниях, меньших чем r_{cut} (~1,5 нм). В отличие от алгоритмов МД, расстояние по времени между обрабатываемыми состояниями может быть разным, и на каждом шаге надо пересчитывать список пар атомов, находящихся на расстояниях $<r_{cut}$. Измерения показывают, что

Таблица 1.

Время выполнения одной итерации расчета RDF и энергии на одиночных компьютерах

Число и тип процессоров	Частота, ГГц	Расчет RDF		Расчет энергии и сил	
		на 1 ядре, с	на всех ядрах, с	на 1 ядре, с	на всех ядрах, с
1*Intel i7-6700k	4,0	0,50	0,125	45,7	10,50
1*Intel i7-5930k	3,5	0,52	0,094	42,7	6,46
1*Intel Xeon E5450	3,0	0,64	0,080	62,6	7,17
2*Intel Xeon E5-2667v2	3,3	0,49	0,037	40,9	2,39
2*Intel Xeon E5-2667v3	3,2	0,54	0,040	43,7	2,47
2*Intel Xeon – E5-2690	2,9	0,52	0,045	45,1	2,66
2*Intel Xeon E5-2697v3	2,6	0,58	0,026	48,9	1,43
2*Intel Xeon E5-2697Av4	2,6	0,53	0,023	42,5	1,25
4*AMD Opteron 6378	2,4	1,09	0,047	84,5	2,37

при вычислении сил и энергий в таком режиме, основное время уходит именно на поиск пар взаимодействующих атомов. Для ускорения процесса поиска пар используется равномерная сетка с шагом равным r_{cut}. Ускорить вычисления можно и за счет распределения атомов между несколькими процессами, применяя MPI или OpenMP.

Для параллельного расчета энергии можно использовать либо равномерное распределение атомов по процессам, либо распределение атомов по набору непересекающихся объемов.

В первом случае каждый процесс обрабатывает группу атомов по возрастанию индексов, а во втором случае каждый процесс обрабатывает только часть атомов, находящихся в заданном объеме. Первый вариант проще реализовать, а второй лучше масштабируется. По этой причине в современных версиях пакета Gromacs используется только разбиение частиц по контрольным объемам [2]. Для измерения производительности применена простейшая схема разбиения на объемы. Ячейка системы разделена вдоль оси X на N слоев по числу MPI или OpenMP процессов. Процесс обрабатывает только те атомы, которые находятся в приписанном ему слое. В свою очередь поиск пар атомов для расчета взаимодействий происходит в слое, границы которого отодвинуты на величину r_{cut} от границ

Таблица 2.

Масштабируемость алгоритмов расчета RDF и энергии на различных компьютерных системах

Число и тип процессоров	Число ядер[1]	Число каналов доступа к памяти	Коэффициент ускорения расчета RDF	Коэффициент ускорения расчета энергии
1*Intel i7-6700k	4	4	4,0	4,4
1*Intel i7-5930k	6	4	5,5	6,6
2*Intel Xeon E5450	8	4	8,0	8,7
2*Intel Xeon E5-2667v2	16	8	13,2	17,1
2*Intel Xeon E5-2667v3	16	8	13,5	17,6
2*Intel Xeon E5-2690	16	8	11,6	17,0
2*Intel Xeon E5-2697v3	28	8	22,2	34,0
2*Intel Xeon E5-2697Av4	32	8	23,0	34,0
4*AMD Opteron 6378	32	16	23,2	35,7
[1]Число физических ядер для работы с числами с плавающей точкой				

основного слоя. Стоит отметить, что график зависимости коэффициента ускорения от числа процессов ведет себя нелинейно, так как края внешнего слоя “срезают” часть соседей в ячейках сетки поиска, которые проверяет и все равно отбрасывает алгоритм в случае, когда все вычисления производит один процесс. Следует отметить, что описанный алгоритм хорошо масштабируется, когда ячейка заполнена атомами более или менее равномерно. Стоит отметить, что график зависимости коэффициента ускорения от числа процессов ведет себя нелинейно, так как края внешнего слоя “срезают” часть соседей в ячейках сетки поиска, которые проверяет и все равно отбрасывает алгоритм в случае, когда все вычисления производит один процесс. Следует отметить, что описанный алгоритм хорошо масштабируется, когда ячейка заполнена атомами более или менее равномерно. Иначе

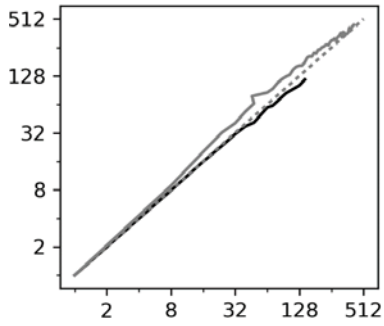


Рис. 1. Зависимость коэффициента ускорения при расчете энергий (серая линия) и RDF (черная линия) от числа MPI процессов. Графики приведены в логарифмическом масштабе по обеим осям. Серым пунктиром показана идеальная масштабируемость.

коэффициент ускорения будет заметно ниже идеального. В этом случае можно или выравнивать загрузку, сдвигая границы слоев, или использовать разбиение по группам частиц. Так как задача анализа небольших групп молекул встречается довольно часто, то алгоритм разбиения по атомам нужен, несмотря на меньшую масштабируемость.

При проведении обработки результатов МД имеет смысл проверить производительность каждого из них на доступном числе ядер и выбрать наилучший. Из рисунка 2 видим, что алгоритм разбиения по объемным слоям лучше масштабируется (вплоть до 512 ядер), но проигрывает алгоритму разбиения по группам частиц пока число процессов меньше 6-8 из-за дополнительных затрат времени на проверки попадания частиц в слой. При увеличении числа процессов больше 30 коэффициент ускорения начинает снижаться.

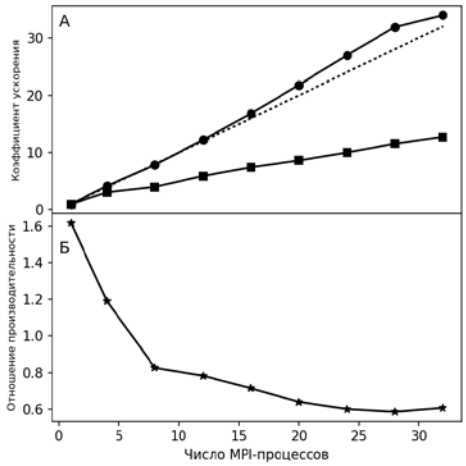


Рис. 2. А. Коэффициент ускорения расчета энергии на кластере при использовании разбиения атомов по объему (кружки) и по индексам (квадраты). Пунктиром показана идеальная масштабируемость. Б. Отношение производительности (больше – лучше) метода разбиения по индексам к разбиению по объему в зависимости от числа MPI-процессов.

5. Выводы

Как уже было отмечено ранее [1], увеличение числа ядер и рост частоты процессора приводят к увеличению скорости расчета, причем использование процессоров с большим числом ядер выгоднее, чем применение процессоров с большей частотой. Производительность серверных процессоров AMD Opteron, также, примерно в 2 раза ниже схожих по частоте и числу ядер процессоров Intel Xeon. Однако особенности алгоритмов немного меняют картину. Расчет RDF, являясь ресурсоемким, плохо масштабируется при использовании 6 или более ядер в режиме OpenMP, поэтому применения MPI и кластеров для него обосновано. Расчет сил и энергий масштабируется лучше, но достигает насыщения, когда на каждый процесс приходится порядка 1000-2000 атомов. Применение MPI возможно, и будет эффективно для систем большого размера (10^5 атомов и более). Стоит отметить, что производительность Xeon E5-2667v2 и v3 в расчетах данного типа мало отличаются (в отличии от расчета МД), поскольку протестированный код не оптимизирован под конкретный набор векторных команд, так как процесс векторных вычислений занимает небольшую долю от общего времени обработки данных. Вопрос о возможности эффективного применения графических ускорителей для вышеупомянутых типов расчетов остался за рамками данной работы, так как требует применения специфичных для ускорителей алгоритмов и специально оптимизированного программного кода. Однако предварительные оценки производительности расчета энергии показывают, что «игровая» видеокарте Nvidia GTX 1080 примерно в 100 раз быстрее, чем CPU Intel Xeon E5-2697v3.

5. Благодарности

Работа выполнена в МСЦ РАН в рамках государственного задания по теме 065-2018-0409. При проведении исследований использовался суперкомпьютер МВС-100К, находящийся в МСЦ РАН.

The performance of modern computing platforms in processing results of molecular dynamics simulation of membrane and protein-membrane systems

N.A. Krylov, D.E. Nolde, P.N. Telegin, R.G. Efremov, B.M. Shabanov

Abstract: We studied the performance of two algorithms for processing results of molecular dynamics (MD) simulation on modern computing platforms: calculations of radial distribution function (RDF) and energies. We found that both algorithms effectively parallelize both on systems with shared memory and on clusters with distributed memory. For processing the results of medium-sized MD systems, the parallelization efficiency of the RDF calculation is close to 1 in the range of up to 100 cores, and in the calculation of energy, up to 500 cores. We found that during energy calculation preferred parallelization depends on number of CPU cores. Parallelization based on atom indices is more effective on small number of cores, while parallelization based on atoms distribution in volume is preferred on large number of cores.

Keywords: molecular dynamics, parallel calculations, high-performance computing, MPI, OpenMP, nonbonded energy, radial distribution function

Литература

1. Д.Е.Нольде, Н.А.Крылов, П.Н.Телегин, Р.Г.Ефремов, Б.М.Шабанов. Производительность современных вычислительных платформ в расчетах молекулярной динамики белок-мембранных систем. «Труды НИИСИ РАН» т. 7. (2017) № 4 стр. 157-161.

2. B.Hess, C.Kutzner, D.van der Spoel, E.Lindahl. GROMACS 4: Algorithms for Highly Efficient, Load-Balanced, and Scalable Molecular Simulation. «J. Chem. Theory Comput.», vol. 4 (2008) № 3, pp. 435–447.

3. M.Abraham, T.Murtola, R.Schulz, S.Páll, J.Smith , B.Hess, E.Lindahl. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. «SoftwareX», vol. 1-2 (2015), pp. 19-25.

4. A.O.Chugunov, P.E.Volynsky, N.A.Krylov, D.E.Nolde, & R.G..Efremov. Temperature-sensitive gating of TRPV1 channel as probed by atomistic simulations of its trans- and juxtamembrane domains. «Scientific Reports» vol. 6 (2016), 33112, http://doi.org/10.1038/srep33112

Федеративная аутентификация в распределенной инфраструктуре суперкомпьютерных центров

А.В.Баранов [1], А.П.Овсянников [2], Б.М.Шабанов [3]

[3] ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

[1,2,3] Межведомственный суперкомпьютерный центр РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mails: [1]antbar@mail.ru, [2]apo@jscc.ru, [3]shabanov@niisi.ru

Аннотация: В статье рассматриваются вопросы организации аутентификации и авторизации в распределенной сети суперкомпьютерных центров. Авторами предлагают использовать механизмы федеративной аутентификации, обеспечивающей единую учетную запись пользователя, поддерживаемой его организацией-работодателем. Предложен метод реализации федеративной авторизации на основе подтверждении публичного ssh-ключа пользователя через браузер. Метод позволяет использовать широко распространенные стандартные решения на основе открытого ПО Shibboleth и SimpleSAMLphp и сочетать механизм федеративной аутентификации с традиционными методами и средствами доступа к суперкомпьютерам, основанными на протоколе ssh.

Ключевые слова: удостоверяющая федерация, федеративная аутентификация, SAML, суперкомпьютерный центр, СКЦ, распределенная сеть суперкомпьютерных центров.

Введение

На сегодняшний день уровень развития сетевых технологий в части организации высокоскоростных защищенных каналов связи, методов и технологий информационной безопасности, облачных и распределенных вычислений, технологий виртуализации и методов их применения, методов и средств имитационного моделирования и искусственного интеллекта позволяет говорить о возможности создания качественно новой цифровой научной инфраструктуры для высокопроизводительных вычислений, объединяющей в единую сеть вычислительные ресурсы территориально распределенных суперкомпьютерных центров коллективного пользования (СКЦ) [1]. Подобное объединение дает возможность оперативного перераспределения нагрузки между ресурсами, что способствует уменьшению их простоя и сокращению среднего времени выполнения пользовательских заданий. Создаваемая цифровая научная инфраструктура будет включать в себя высокопроизводительные вычислительные кластеры (суперкомпьютеры) разных архитектур, связывающие их защищенные высокоскоростные каналы связи, общую распределенную файловую систему и единую систему доступа к суперкомпьютерным ресурсам. Возникает потребность в исследовании и разработке новых методов и алгоритмов, касающихся всех аспектов функционирования распределенной сети СКЦ: организации единого доступа к пулу объединенных ресурсов, распределения пользовательских заданий по вычислительным ресурсам, мониторинга и прогнозирования состояния вычислительной среды, организации централизованного управления данными, моделирования и анализа работы всей системы.

Одной из задач исследования является разработка методов аутентификации и авторизации на распределенных суперкомпьютерных ресурсах, с использованием единой учетной записи пользователя, поддерживаемой его организацией [1].

Федеративная система аутентификации и авторизации распределенной сети СКЦ

Принципиальной чертой проектируемой распределенной сети СКЦ [1] является ее децентрализация. Все суперкомпьютерные центры коллективного пользования независимы: ими владеют и управляют разные научные организации в разных ведомствах (Минобрнауки России, МГУ им. М.В. Ломоносова, НИЦ Курчатовский институт). Соответственно, сейчас пользователь суперкомпьютерных ресурсов имеет в каждом суперкомпьютерном центре отдельную учетную запись, и каждый суперкомпьютерный центр ведет свою собственную базу (реестр) пользователей. Например, в МСЦ РАН ведется учет потреб-

ленных суперкомпьютерных ресурсов по отдельным пользователям, научным проектам (темам), организациям.

Каждый СКЦ при такой схеме сталкивается с проблемой получения и обработки персональных данных пользователя, защиту которых СКЦ обязан обеспечить в соответствии с действующим законодательством, что для научного центра является непрофильной задачей. В подавляющем большинстве случаев руководство СКЦ и представители учредителей отслеживают потребление ресурсов по организациям и научным проектам. Персональные данные пользователя требуются только для экстренной связи с ним, а также для формирования агрегированной обезличенной статистики, запрашиваемой учредителями (сколько пользователей в возрасте до 39 лет, каков процент пользователей с ученой степенью и т.п.). Подобного вида статистика без юридических и технических препятствий может быть запрошена суперкомпьютерным центром у организации – работодателя пользователя.

Заметим, что один и тот же пользователь может работать в одном или нескольких институтах (организациях) над несколькими научными проектами, а в одном проекте могут использоваться различные ресурсы из разных СКЦ или ЦКП. Организации, в которых работают пользователи, нуждаются в оперативной информации о том, сколько суперкомпьютерных ресурсов и в каких СКЦ было потреблено в рамках того или иного научного проекта (темы). При этом руководители проектов и организаций заинтересованы в возможности самостоятельного перераспределения квот на ресурсы между своими проектами и пользователями.

Выход видится в создании распределенной системы аутентификации, авторизации и учета, в которой пользователь будет иметь единственную учетную запись, а институты могли бы гибко управлять выделяемыми или приобретаемыми суперкомпьютерными ресурсами.

Подходящей основой для такой системы может стать удостоверяющая федерация [2] из суперкомпьютерных центров/институтов, доверяющих друг другу аутентификацию пользователя.

Принципы федеративных систем аутентификации разработаны глобальным консорциумом университетов и ведущих компаний IT-индустрии OASIS (англ. Organization for the Advancement of Structured Information Standards). В 2002 году OASIS был разработан язык разметки утверждений безопасности (SAML – Security Assertion Markup Language) [3, 4] – открытый стандарт, определяющий протоколы и правила взаимодействия узлов федеративной системы аутентификации. В рамках проекта eduGAIN (EDUcational Global Authentication Infrastructure) [5] европейского сетевого консорциума GÉANT создана система национальных удостоверяющих федераций для доступа к контенту, сервисам и ресурсам глобального сообщества сферы образования и науки.

Технология федеративной аутентификации подразумевает доступ к информационным ресурсам и сервисам научно-образовательных сетей по технологии «единого входа» (Single Sign-On – SSO). При этом хранение, обработка и, соответственно, защита основных персональных данных пользователей (фамилия, имя, отчество, паспортные данные, должность и т.п.) производится в его «домашней» организации, что избавляет СКЦ или ЦКП от необходимости обработки этих данных (по крайней мере, с использованием средств автоматизации).

Удостоверяющая федерация (см. рисунок 1) поддерживает центральный реестр, содержащий метаданные организаций-участников, на основе которых службы центрального реестра формируют сервис «Вы откуда?» (Where Are You From – WAYF). При обращении (1) пользователя к ресурсу, требующему авторизации, сервис-провайдер (SP – Service Provider) ресурса вызывает (2) службу WAYF, в интерфейсе которой пользователь выбирает (3) свою «домашнюю» организацию. WAYF перенаправляет (4) запрос к выбранному пользователем поставщику учетных записей (IdP – Identity Provider). IdP запрашивает (5) у пользователя данные для аутентификации и сличает их с записью в хранилище учетных записей о своих пользователях. В случае успешной аутентификации IdP обменивается (6) с поставщиком сервиса SP утверждениями на языке SAML о результатах аутентификации, авторизации и атрибутах.

При федеративной аутентификации в распределенной сети суперкомпьютерных центров для входа на суперкомпьютер пользователь для подтверждения своих полномочий по использованию вычислительных ресурсов должен обозначить «домашнюю организацию» – входящий в удостоверяющую федерацию институт/центр/университет, в котором у него есть учетная запись. Проверка логина и пароля пользователя будет выполнена на сервере аутентификации выбранного института, который отправит службе авторизации на входе суперкомпьютерного ресурса подтверждение, что пользователю разрешено использование этого ресурса в некотором допустимом объеме. На основании полученных

от института данных службы авторизации и учета суперкомпьютерного ресурса примут решение о допуске пользователя, а по завершении его заданий проинформируют институт пользователя об объемах израсходованных ресурсов.

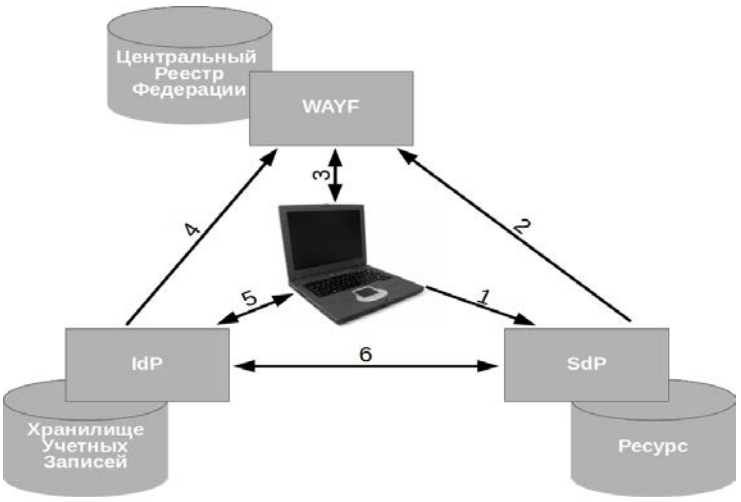


Рисунок 1. Федеративная авторизация

Сообщения, которыми обмениваются службы авторизации и учета суперкомпьютерного ресурса и сервер идентификации института, должны быть оформлены в виде утверждений (assertion) на языке SAML и зашифрованы с использованием SSL-сертификатов. Эти сообщения, помимо запроса аутентификации и ее подтверждения, также могут содержать атрибуты для идентификации проекта, квоты суперкомпьютерных ресурсов, выделяемой институтом авторизуемому пользователю в рамках научного проекта; могут связывать с пользователем, проектом и институтом имя временной учетной записи, которую назначает СКЦ как провайдер сервиса (SP).

Способ реализации федеративной аутентификации для сети СКЦ

Реализации провайдеров аутентификации (IdP) и сервис-провайдеров (SP) для технологии единого входа SSO на базе SAML ориентированы прежде всего на веб-сервисы и протокол https.

Доступ к суперкомпьютерным ресурсам в подавляющем большинстве СКЦ осуществляется по протоколу ssh [6]. Существует несколько возможных вариантов использования SSO для доступа к вычислительным ресурсам через ssh, например, GSS-API [7] или Moonshot [8]. Авторы настоящей работы предлагают метод, основанный на подтверждении публичного ssh-ключа пользователя через браузер.

Предлагаемый подход подразумевает использование стандартных реализаций провайдера идентификации IdP, сервис-провайдера SP и аутентификацию пользователя с использованием пользовательского агента (UA – User Agent) на основе браузера (см. рисунок 2).

При обращении (1) к SP пользователь указывает уникальный идентификатор eduPersonPrincipalName (ePPN) [9] и публичный ключ pkey, который будет использоваться для доступа по протоколу ssh к суперкомпьютеру (СК). На основе ePPN SP определяет IdP (2), к которому следует обратиться за аутентификацией пользователя, и перенаправляет туда браузер пользователя (3) для ввода пароля. В обращении (2) в качестве дополнительного атрибута передается имя учетной записи (login), которое будет использовано для входа пользователя на СК. Это имя генерируется на основе ePPN и сообщается IdP для использования в системе учета и квотирования института пользователя. IdP возвращает результат аутентификации пользователя (4) провайдеру сервиса SP. Помимо собственно сообщения о результатах аутентификации (AUTH_STATUS) IdP передает полученную от системы учета «домашней» организации информацию о том, к какому научному проекту (proj_id) относится работа пользователя, и сколько ресурсов (quota) ему разрешено потратить в рамках этого проекта. В случае успешной аутентификации пользователя провайдер сервиса SP авторизует (5) его на СК – создает (или разблокирует) учетную запись login и

связывает ее с используемым пользователем публичным ключом (pkey). В ответе SP о результатах авторизации (6) содержится имя учетной записи (login), которую пользователь может использовать для доступа к СК. Заметим, что предусматривается периодическое подтверждение аутентификации.

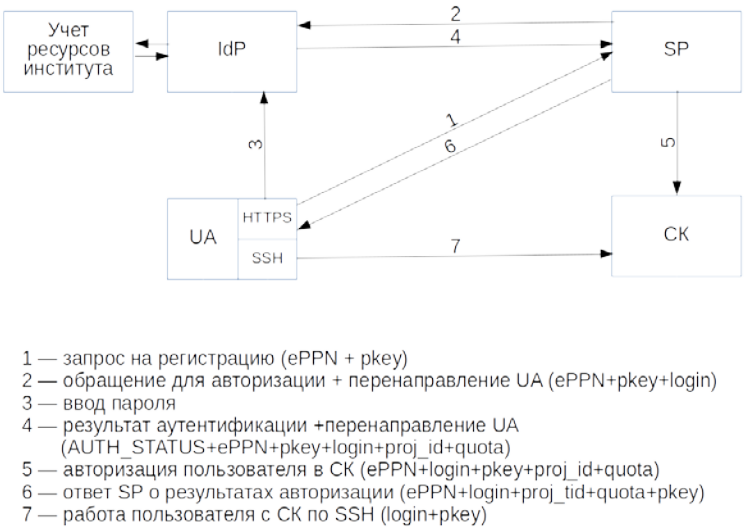


Рисунок 2. Федеративная аутентификация в сети СКЦ на основе подтверждения публичного ключа

Вместо выбора «домашней» организации пользователя в интерфейсе WAYF предлагается идентификация домашней организации по eduPersonPrincipalName.

В распределенной системе СКЦ необходимо предусмотреть отображение «глобального» имени пользователя на локальные имена в каждом из СКЦ.

При этом в качестве «глобального идентификатора» предлагается использовать имя учетной записи пользователя на одной из суперкомпьютерных систем (в одном из СКЦ) в сочетании с доменным именем системы (СКЦ) вида user@sc-domen.

Главным достоинством предложенного подхода реализации федеративной аутентификации является возможность использования существующих реализаций программного обеспечения провайдеров идентификации и сервиса, например, Shibboleth [10] или SimpleSAMLphp [11].

Следует также отметить, что утверждения SAML, передаваемые между SP и IdP, могут нести информацию о доступной пользователю квоте или потребленных ресурсах. Это позволит построить безопасный обмен информацией для системы учета.

Недостатком метода является необходимость выполнения пользователем дополнительных манипуляций с использованием веб-браузера вместо обычного входа по ssh. Однако дополнительные действия пользователя (например, выбор института в WAYF) являются неизбежной платой за использование федеративной аутентификации.

Заключение

Удостоверяющая федерация суперкомпьютерных центров обеспечит пользователю возможность использования единой учетной записи для доступа на все суперкомпьютерные ресурсы распределенной сети СКЦ, а его организации – гибкую систему управления квотами ресурсов между выполняемыми научными проектами и сотрудниками – пользователями СКЦ. При этом распределение квот и объемов ресурсов может быть произведено внутри организации (научного института), без взаимодействия с биллинговыми системами СКЦ. Для взаимного учета использования суперкомпьютерных ресурсов институтами в распределенной сети СКЦ с перераспределением заданий между центрами могут использоваться технологии распределенного реестра и умных контрактов.

Рассмотренный в статье предлагаемый авторами подход к организации авторизации пользователей в распределенной сети СКЦ позволяет сочетать механизм федеративной аутентификации с традиционными методами доступа к суперкомпьютерам, основанными на протоколе ssh.

Представляется, что удостоверяющая федерация суперкомпьютерных центров может быть интегрирована в удостоверяющую федерацию федеральной университетской компьютерной сети RUNNet (RUNNetAAI) [12], что обеспечит пользователю возможность использования одной учетной записи и для доступа к СКЦ, и для доступа к информа-

ционным ресурсам научно-образовательных сетей в рамках ассоциации eduGAIN.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований по теме (проекту) «Исследование и разработка методов сетевой интеграции ресурсов и сервисов научных организаций» (0065-2018-0404).

Federative Identity for the Distributed Infrastructure of the Supercomputer Centers

A.V.Baranov, A.P.Ovsyannikov, B.M.Shabanov

Abstract: The article considers the approach for authentication and authorization for the distributed system of the supercomputer centers. The authors propose to use federative identity for the providing Single Sign On for the users. It allows to store and process the personal data in the home institute of the user according to the legislation requirements. The method proposed by the authors is based on the confirmation of public SSH-keys using browser. The method allows to use widespread standard solutions based on open source software Shibboleth and SimpleSAMLphp.

Keywords: Federative Identity, authentication, SAML, supercomputer center, distributed system of the supercomputer center.

Литература

1. Б.М.Шабанов, А.П.Овсянников, А.В.Баранов, С.А.Лещев, Б.В.Долгов, Д.Ю.Дербышев. Проект распределенной сети суперкомпьютерных центров коллективного пользования // Программные системы: теория и приложения. 2017. № 4(35). С. 245–262. DOI: 10.25209/2079-3316-2017-8-4-245-262

2. А.П.Овсянников, Г.И.Савин, Б.М.Шабанов. Удостоверяющие федерации научно-образовательных сетей // Программные продукты и системы. 2012. № 4. С. 3–7.

3. E.Simon. Protecting Privacy Using XML, XACML, and SAML. In Privacy Protection for E-Services, ed. George Yee, 203-233 (2006). DOI:10.4018/978-1-59140-914-4.ch008

4. SAML V2.0 Executive Overview / OASIS, 2005, [электронный ресурс] URL: https://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf (дата обращения 14.08.2018)

5. eduGAIN [электронный ресурс] / GÉANT, 2018, URL: https://edugain.org/ (дата обращения 14.08.2018)

6. D.Barrett, R.Silverman, R.Byrnes. SSH, The Secure Shell: The Definitive Guide, 2nd Edition. O'Reilly Media. 2009. 672 p. ISBN 978-0596008956

7. S.Cantor. Federating SSH – CICIDM – Carmen Wiki. May 24, 2013 [электронный ресурс] / URL: https://carmenwiki.osu.edu/display/CICIDM/Federating+SSH (дата обращения 08.11.2018)

8. H.Flanagan, S.Paetow. Introduction to Moonshot. Apr 16, 2018 [электронный ресурс] / URL: https://wiki.moonshot.ja.net/display/Moonshot/Introduction+to+Moonshot (дата обращения 08.11.2018)

9. Технологический регламент Участника Удостоверяющей Федерации RUNNetAAI [электронный ресурс] / ФГАУ ИТТ Информика, 2018 URL:https://www.runnet.ru/documents-ru-1/policy (дата обращения 14.08.2018)

10. Shibboleth Consortium [электронный ресурс] / URL: https://www.shibboleth.net/ (дата обращения 14.08.2018)

11. SimpleSAMLphp [электронный ресурс] / UNINETT, 2018 URL: https://simplesamlphp.org/ (дата обращения 14.08.2018)

12. RUNNetAAI [электронный ресурс] / ФГАУ ИТТ Информика, 2018 URL: https://www.runnet.ru/services/runnetaai (дата обращения 17.05.2018)

Способы и средства представления пользовательских суперкомпьютерных заданий в виде контейнеров Docker

Г.И.Савин[1], П.Н.Телегин[2], А.В.Баранов[3], А.С.Шитик[4]

[1,2,3]Межведомственный суперкомпьютерный центр РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

[4] Московский физико-технический институт (государственный университет), Москва, Россия,

E-mails: [1]savin@jscc.ru, [2]ptelegin@jscc.ru, [3]antbar@mail.ru, [4]salexs95@yandex.ru

Аннотация: Одним из ключевых методов повышения эффективности использования вычислительных ресурсов является применение технологии контейнерной виртуализации. В отношении организации высокопроизводительных вычислений контейнерная виртуализация позволяет с минимальными накладными расходами решить проблему бинарной переносимости пользовательских заданий между различными суперкомпьютерными установками. Статья посвящена поиску и выбору способов и средств, позволяющих сформировать суперкомпьютерное задание в виде контейнера Docker с произвольным содержимым, определяемым пользователем. В статье приведены методика и рекомендации по формированию образа контейнера задания, рассмотрены вопросы информационной безопасности при представлении заданий в виде контейнеров.

Ключевые слова: контейнерная виртуализация, высокопроизводительные вычисления, суперкомпьютер, Docker, системы управления заданиями, информационная безопасность, СУППЗ

Введение

Одной из устойчивых тенденций развития суперкомпьютерных центров коллективного пользования (СКЦ) является объединение их вычислительных ресурсов в единую распределённую систему. Активные работы над проектом единой распределённой сети научных суперкомпьютерных центров в настоящее время ведутся в Межведомственном суперкомпьютерном центре РАН [1]. В качестве единицы ресурса, подключаемой к сети СКЦ, в проекте рассматривается вычислительная установка (ВУ) – высокопроизводительный вычислительный кластер (суперкомпьютер), состоящий из множества многопроцессорных (многоядерных) вычислительных модулей (ВМ), объединённых высокоскоростной сетью передачи данных. Кроме высокоскоростной сети в современных суперкомпьютерах, как правило, наличествуют транспортная и управляющая сети. Транспортная сеть служит для организации связи с системой хранения данных, управляющая сеть используется для комплексного управления решающим полем вычислительных модулей. Отметим, что создаваемая в соответствии с [1] распределённая сеть СКЦ подразумевает наличие в своём составе ВУ различных архитектур и производительности.

Современные суперкомпьютеры используются в многопользовательском режиме коллективного доступа, при котором для производства вычислений пользователь должен оформить т.н. задание, в состав которого входят:

– параллельная программа, реализующая алгоритм решения прикладной задачи, параллельная программа состоит из нескольких взаимодействующих процессов, которые могут одновременно выполняться на нескольких вычислительных модулях суперкомпьютера;

– требования к ресурсам: сколько вычислительных модулей, с какими характеристиками, на какое время требуются для выполнения параллельной программы;

– входные данные параллельной программы.

Управление пользовательскими заданиями в современных суперкомпьютерных системах осуществляется с помощью специального программного обеспечения – систем управления заданиями (СУЗ) [2], в качестве которых могут выступать такие известные продукты, как SLURM [3], PBS [4], Moab [5] или отечественная Система управления прохождением параллельных заданий (СУППЗ) [6].

Основной функционал СУЗ состоит в приёме входного потока пользовательских заданий, ведении очереди заданий, выделении вычислительных модулей суперкомпьютера для прошедшего через очередь задания, запуске и контроле выпол нения задания, освобождении выделенных ресурсов по завершении задания. Важным аспектом функционирования любой СУЗ является обеспечение надёжной изоляции пользовательских заданий. Изоляция заданий подразумевает конфигурирование выделенных заданию ВМ таким образом, чтобы доступ к этим ВМ мог получить только пользователь – владелец задания. Кроме того, должно быть исключено влияние выполняющихся процессов одного задания на выполнение процессов другого. Под влиянием здесь понимается не только злонамеренное или случайное воздействие на выполняющийся процесс (например, посылка сигнала), но и конкуренцию процессов за ресурсы ВМ (процессор, память, устройства ввода-вывода и др.), приводящую к снижению быстродействия вычислений.

Объединение нескольких высокопроизводительных ВУ в единую сеть СКЦ подразумевает формирование глобальной очереди пользовательских заданий [2], из которой задания могут быть распределены на любую ВУ из состава сети СКЦ. Поскольку включенные в сеть СКЦ ВУ могут различаться как по архитектуре, так и по составу программного обеспечения (ПО), при распределении заданий глобальной очереди возникает проблема т.н. бинарной несовместимости. Проблема заключается в том, что параллельная программа из состава пользовательского задания зависит от стека системного и инструментального программного обеспечения (ПО), установленного на суперкомпьютере. Это означает, что программа, собранная и подготовленная к выполнению на одной ВУ, как правило, не может быть выполнена на другой ВУ без перетрансляции. Кроме этого, для своего выполнения параллельная программа может требовать специфического программного окружения, установленных программных пакетов и/или библиотек, наличия приобретённых пользователем лицензий. Всё перечисленное может привести к невозможности распределения задания из глобальной очереди на произвольную ВУ сети СКЦ, что существенно снижает функциональность создаваемой распределённой системы.

Преодоление бинарной несовместимости суперкомпьютерных приложений возможно за счёт применения технологий виртуализации. Задание представляется в виде специального образа (виртуальной машины или контейнера), в который пользователь включает необходимый для выполнения задания стек системного, инструментального и прикладного ПО. Пользователь получает возможность самостоятельно выбрать программную платформу параллельных вычислений и использовать её для своих вычислительных заданий. В дальнейшем созданный образ может быть запущен на любой суперкомпьютерной системе как обычное пользовательское задание.

Целью настоящей работы является исследование и разработка способов представления пользовательского задания в виде контейнера для возможности запуска под управлением СУЗ с соблюдением принципа взаимной изоляции запускаемых заданий.

Методы и средства применения технологий виртуализации для организации высокопроизводительных вычислений

Существуют два основных подхода к виртуализации как к созданию независимых изолированных вычислительных пространств на одном физическом ресурсе: виртуальные машины и контейнерная виртуализация. В первом случае на одном физическом компьютере создаётся множество виртуальных машин, в каждой из которых используется собственная, т.н. «гостевая», операционная система (ОС) с её собственным ядром. Управление виртуальными машинами и ресурсами физического компьютера осуществляется специальным программным средством, именуемым гипервизором. По этой причине такой тип виртуализации часто называют гипервизорной виртуализацией. Гипервизорная виртуализация позволяет в образе виртуальной машины задать практически любой стек ПО, включая операционную систему, однако, развёртывание образов виртуальных машин и их выполнение в виде пользовательских заданий влечёт достаточно высокие накладные расходы [8].

Контейнерная виртуализация или виртуализация на уровне операционной системы [9] – это метод, при котором ядро ОС поддерживает несколько изолированных экземпляров пользовательского пространства, что позволяет запускать изолированные и безопасные «виртуальные машины» на одном физическом компьютере. Контейнерная

виртуализация ограничивает тип используемой гостевой ОС операционной системой физического компьютера, но накладные расходы на контейнерную виртуализацию существенно ниже по сравнению с гипервизорной [10], что очень важно для суперкомпьютерных расчётов.

Контейнерная виртуализация может быть применена для обеспечения беспрепятственной миграции пользовательских заданий в распределённой сети СКЦ. Для этого задание представляется в виде контейнера, в который включается весь необходимый для выполнения задания стек ПО. Представленное в виде контейнера задание направляется пользователем в глобальную очередь сети СКЦ. После прохода через глобальную очередь задание-контейнер распределяется на одну из ВУ (суперкомпьютер) из состава сети СКЦ, попадая при этом в локальную очередь СУЗ, установленной на этой ВУ. После прохождения локальной очереди СУЗ заданию будет выделено некоторое подмножество ВМ суперкомпьютера. На каждом из выделенных ВМ должен быть развёрнут контейнер задания, после чего внутри развёрнутых контейнеров запускаются процессы параллельной программы пользовательского задания. Содержащееся непосредственно в контейнере необходимое для этих процессов программное окружение должно обеспечить беспрепятственные старт и выполнение параллельной программы. По окончании выполнения программы (истечении заказанного пользователем времени) СУЗ освобождает занятые заданием ресурсы, тем самым завершая задание. Освобождение ресурсов производится путём завершения процессов пользователя на выделенных заданию ВМ, сворачивания контейнеров и сохранения результатов расчётов.

Существует два метода применения технологии контейнерной виртуализации для организации высокопроизводительных вычислений в режиме коллективного пользования под управлением СУЗ. Первый заключается в том, что пользователю СУЗ предоставляется подготовленный системными администраторами контейнеров репозиторий (локальный, в рамках СКЦ, или глобальный, в рамках сети СКЦ) контейнеров с разными (в идеале – со всевозможными) стеками ПО. Пользователь при запуске задания выбирает из репозитория наиболее подходящий для него контейнер. Метод репозитория обладает двумя основными преимуществами. Первое – простота и надёжность реализации, поскольку подготовкой образа для контейнера занимаются специалисты в этой области, т.е. системные администраторы. Второе преимущество, вытекающее из первого, заключается в высокой безопасности, обеспечиваемой за счёт контроля процесса подготовки контейнера со стороны системного администратора. Осуществление первого метода рассмотрено в работе [11].

Второй метод подразумевает для пользователя возможность запуска любого подготовленного им контейнера с произвольным наполнением и выбранным пользователем стеком ПО. Реализация этого варианта значительно сложнее и подразумевает решение ряда следующих научно-технических задач.

Задача 1. Необходимо разработать способ представления задания в виде контейнера, методику и средства применения этого способа пользователем. Способ должен позволять пользователю в рамках заданных правил и ограничений упаковывать в контейнер произвольный стек инструментального и прикладного ПО. Важным требованием при этом является то, что формируемый контейнер не должен быть ориентирован на выполнение под управлением какой-либо конкретной СУЗ (SLURM, PBS или т.п.), поскольку заранее неизвестно, в какую СУЗ будет распределено пользовательское задание из глобальной очереди.

Задача 2. Необходимо обеспечить безопасность вычислений в пользовательских контейнерах. С одной стороны, это означает возможность доступа пользовательских процессов параллельной программы к данным в соответствии с теми и только теми правами, которыми обладает пользователь. Иными словами, должна быть обеспечена техническая невозможность превышения пользователем своих привилегий. С другой стороны, должна быть обеспечена защита пользовательских процессов и данных контейнера от несанкционированного доступа извне.

Задача 3. Необходимо сконфигурировать подсеть развёрнутых на ВМ контейнеров таким образом, чтобы выполняющиеся внутри контейнеров процессы параллельной программы «видели» друг друга и могли совершать информационные обмены.

Задача 4. Необходимо разработать сценарии СУЗ для развёртывания на выделенных вычислительных ресурсах представленных в виде контейнеров пользовательских заданий, контроля их выполнения и сворачивания контейнеров после завершения заданий.

Способы и средства конфигурации сети развёрнутых контейнеров, а также разработанные сценарии работы с заданиями-

контейнерами в СУППЗ (решение задач 3 и 4) рассмотрены авторами в работе [12], в настоящей статье будут затронуты задачи 1-2.

Если рассматривать средства контейнерной виртуализации, то необходимо отметить, что ведущим ПО в этой области является программная система Docker, предоставляющая API для создания и использования контейнеров. Docker позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, переносимый между любыми Linux-системами, ядро которых обладает поддержкой механизма контрольных групп. За последние годы Docker стал де-факто стандартом в области контейнерной виртуализации.

ПО Docker может быть использовано для представления суперкомпьютерных пользовательских заданий в виде контейнеров [11, 13, 14]. В указанных работах развёртывание на суперкомпьютере пользовательского задания в виде набора контейнеров существенно опирается на возможности и средства распространённой СУЗ SLURM, что ограничивает применение предложенных способов и противоречит условиям задачи 1. Например, в работе [13] отмечается, что без использования средств SLURM внутри контейнеров авторам не удалось обеспечить безопасность вычислений.

Методика формирования пользовательского задания в виде контейнера Docker

Для представления вычислительного задания в виде контейнера необходимо создать образ Docker, который будет содержать весь необходимый для выполнения вычислений стек ПО. Внутри контейнера должен быть доступ к исполняемому файлу параллельной программы задания. Для сборки образов по заранее подготовленному сценарию (Dockerfile) используется команда «docker build». Возможен вариант создания образа вычислительного задания путем итеративной процедуры запуска образа, осуществления установки необходимого ПО с последующим сохранением изменений. Созданный таким способом промежуточный образ используется на следующей итерации процесса подготовки контейнера вычислительного задания.

Образ Docker-контейнера представляет собой многослойную структуру, каждый слой которой содержит определённые при его создании изменения файловой системы контейнера. Примерная структура образа представлена на рисунке 1.

Базовым слоем является образ операционной системы, на этом уровне возможно определить, какой дистрибутив ОС Linux будет использован в контейнере. В этот слой включаются необходимые системные утилиты и системные конфигурационные файлы.

Следующим уровнем идёт слой ПО, обеспечивающего использование в контейнере высокоскоростных сетевых устройств, связывающих ВМ решающего поля суперкомпьютера. Высокоскоростные коммуникационные сети предназначены для обмена информацией между процессами параллельной программы, выполняющимися на разных выделенных заданию ВМ. Стандартом последних лет в области высокопроизводительных вычислений стало использование в качестве высокоскоростных сетей Infiniband. С 2016 года началось внедрение сетей Intel Omni-Path [15] – реализации Infiniband от компании Intel. Добавление такого слоя в образ позволит ОС контейнера использовать высокоскоростные устройства вычислительного модуля, на котором происходит запуск контейнера. После своего запуска контейнеры одного задания образуют защищённую подсеть, способы и средства конфигурирования такой подсети рассмотрены авторами в работе [12].

Рисунок 1. Примерная структура образа контейнера для вычислительного задания

В следующем слое представлены средства организации удалённого выполнения команд внутри сконфигурированной подсети контейнеров, необходимые для функционирования коммуникационных библиотек и взаимодействия процессов параллельной программы. Как правило, подобные средства используют протокол Secure Shell (ssh) и пред-

ставлены в виде клиентской и серверной частей. Над этим слоем надстраивается слой коммуникационных библиотек и средств разработки приложений. Коммуникационные библиотеки являются неотъемлемой частью параллельной программы, с их помощью осуществляются информационные обмены между процессами параллельной программы. В подавляющем большинстве случаев при организации параллельных вычислений используется библиотека, реализующая стандарт Message Passing Interface (MPI).

Последним в образе контейнера идёт слой, содержащий исполнительные, конфигурационные, лицензионные и пр. файлы пользовательского задания. Можно выделить два подхода добавления файлов задания в контейнер:

а) добавление файлов вычислительного задания в образ контейнера на этапе его формирования;

б) размещение файлов вычислительного задания в домашнем каталоге пользователя, который монтируется средствами ПО Docker в контейнер в момент его запуска.

Добавление вычислительного задания в образ возможно произвести на этапе его создания в файле сценария сборки директивами «COPY» или «ADD». Директива COPY выполняет копирование указанного в первом аргументе файла/каталога в каталог создаваемого образа.

Второй вариант добавления файлов задания может являться предпочтительным по причине удобства изменения/обновления исполняемого и вспомогательных файлов в процессе подготовки задания пользователем. В случае необходимости внести определённые изменения не нужно заново выполнять сборку контейнера и производить его обновление в репозитории. В этом случае пользователь помещает файлы задания в свой домашний каталог и указывает команду, которую необходимо выполнить при запуске задания.

Модель информационной безопасности суперкомпьютерной системы

В основе построения систем защиты открытой информации в суперкомпьютерных центрах коллективного пользования обычно лежат возможности и средства базовых операционных систем, в первую очередь, Linux. Как показывает многолетняя практика, указанных возможностей и средств вполне достаточно для обеспечения информационной безопасности (ИБ) СКЦ. Применяемая при этом политика безопасности достаточно проста и заключается в следующем. Пользователям разрешён доступ ко всем информационным и вычислительным ресурсам в рамках прав (привилегий), определённых владельцем ресурса или системным администратором. Последний обладает исключительными правами на доступ к любым вычислительным и информационным ресурсам, по этой причине в Unix-подобных ОС системный администратор называется суперпользователем.

Важнейшим элементом ИБ является недопущение получения привилегий суперпользователя пользователем, не прошедшим авторизацию в установленном порядке. Заметим, что в контейнерах Docker пользователь внутри контейнера по умолчанию имеет эффективный идентификатор, обеспечивающий ему привилегии суперпользователя, что существенно осложняет задачу обеспечения ИБ при использовании контейнерной виртуализации. Для решения этой задачи построим модель безопасности суперкомпьютера, работающего в режиме коллективного пользования под управлением СУЗ.

Определим следующие термины. *Информационный объект* – данные пользователей, хранящиеся в системе хранения данных (СХД), и системные данные, доступ к которым имеет только суперпользователь. *Вычислительный объект* – ресурсы суперкомпьютера, выделяемые СУЗ субъекту для выполнения вычислительного задания. *Субъект* – пользователь суперкомпьютера, а также выполняющиеся на ВМ процессы пользовательского задания.

Разрабатываемая модель должна обеспечивать безопасность объектов, не принадлежащих (выделенных) субъекту в рамках выполнения его вычислительного задания, представленного в виде контейнера.

В рамках настоящей модели *конфиденциальность* означает невозможность получения субъектом несанкционированного доступа (НСД) к информационным объектам системы с целью получения информации других пользователей или системной информации. Другими словами, пользователи СКЦ не должны иметь возможности чтения данных других пользователей без соответствующего разрешения владельца данных.

Информационные объекты системы должны быть защищены от изменения субъектами, не имеющими таких привилегий в системе, что обеспечивает свойство *целостности*. Субъект не должен иметь возможности изменять конфигурацию вычислительных

объектов системы, а также изменять информационные объекты других субъектов системы. Действия субъекта в системе должны контролироваться с целью предотвращения удаления им информационных объектов других субъектов. Другими словами, пользователи СКЦ не должны иметь возможности изменять аппаратную или программную конфигурацию суперкомпьютера, а также изменять или удалять данные других пользователей без соответствующего разрешения владельца.

Необходимо защитить систему от захвата субъектом вычислительных объектов, которые ему не выделялись через СУЗ, чем обеспечивается такое свойство ИБ, как *доступность*.

Схема модели безопасности представлена на рисунке 2.

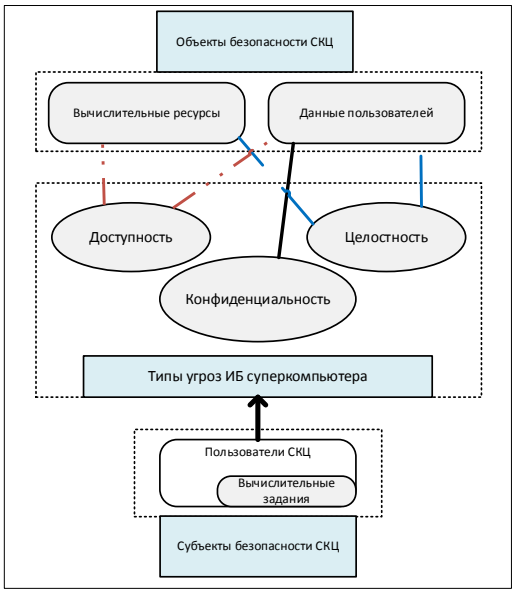


Рисунок 2. Модель безопасности для суперкомпьютерной системы в режиме коллективного пользования

Реализация модели безопасности

Реализация модели ИБ должна обеспечивать запуск вычислительных заданий, представленных в виде контейнеров, и предотвращать превышение пользователем прав, предусмотренных системным администратором, с целью получения НСД к информационным и вычислительным объектам суперкомпьютера.

Стандартно контейнеры при запуске программным средством Docker имеют единственного пользователя – суперпользователя root. Проблема состоит в том, что в случае монтирования в контейнер каталогов, расположенных в СХД СКЦ, суперпользователь контейнера становится эквивалентным суперпользователю вычислительного модуля или управляющего компьютера, на котором был запущен контейнер. Обладая фактически привилегиями суперпользователя, процессы задания из контейнера могут получить неограниченный доступ к примонтированным каталогам пользователей и могут читать, изменять и исполнять любые файлы. Это ставит под угрозу безопасность монтирования в контейнер любых каталогов, кроме домашнего каталога пользователя. Решение проблемы в виде запрета монтирования в контейнер каталогов других пользователей неприемлемо, т.к. делает невозможной совместную работу нескольких пользователей. Для совместной работы необходимо монтировать домашние каталоги всех пользователей (для возможности пользователя «А» предоставления пользователю «Б» доступа к файлам, хранящимся в домашнем каталоге пользователя «А») и/или каталоги, содержащие общие для всех пользователей файлы. В этих условиях необходимо решить задачу ограничения повышения пользователями своих привилегий внутри контейнера. Рассмотрим предлагаемое авторами решение.

Одним из аргументов команды «docker run», осуществляющей запуск контейнера из образа, является:

--user=[user | user:group | uid | uid:gid | user:gid | uid:group]

Данный параметр указывает Docker, что необходимо запустить контейнер, и пользователь внутри контейнера будет иметь указанные идентификаторы. В случае с указанием имени пользователя необходимо, чтобы в базовом образе был создан пользователь с указанными именем и группой.

При запуске контейнера с аргументом «--user» пользователь имеет в контейнере права, аналогичные правам вне контейнера. Однако, такой пользователь всё еще имеет возможность исполнять файлы, у которых установлен «suid-бит» – т.н. suid-программы. Suid-программа запускается от имени владельца исполняемого файла программы, а не от имени запускающего программу пользователя. Примером suid-программы в ОС семейства Linux является утилита passwd, предназначенная для изменения пароля пользователя. Для изменения пароля утилите passwd требуется доступ к хранилищу хэш-сумм паролей, которым обладает только суперпользователь. Установка suid-бита позволяет утилите выполняться с правами

суперпользователя, даже будучи запущенной обычным пользователем.

Учитывая сказанное, возможен следующий сценарий превышения пользователем своих привилегий. В момент создания базового образа контейнера пользователь может создать suid-программу, принадлежащую суперпользователю. Эта программа может несанкционированно повышать привилегии пользователя, осуществлять НСД к информационным и вычислительным объектам суперкомпьютера, выполнять иные вредоносные действия. После старта контейнера эта программа запускается пользователем и начинает выполняться от имени суперпользователя.

Приведём ещё один возможный сценарий присвоения пользователем привилегий суперпользователя в момент создания образа контейнера. На стадии формирования образа пользователь добавляет в файл /etc/sudoers строку «username ALL=(ALL) NOPASSWD: ALL» и устанавливает программу sudo. На стадии выполнения задания-контейнера это даёт пользователю с именем «username» возможность получения прав суперпользователя в контейнере.

Начиная с версии 1.10, Docker поддерживает новые возможности в сфере безопасности [16].

Проблема повышения привилегий с самого начала развития проекта Docker стояла достаточно остро, её решением является применение в качестве аргумента осуществляющей запуск контейнера команды «docker run» параметра «security opt no new privileges».

Использование этого параметра запуска контейнера совместно с атрибутом «user» позволяет полностью исключить возможность повышения пользователями своих привилегий внутри контейнера и примонтированных к нему каталогах.

У пользователя отсутствует возможность запуска suid-программ (его эффективный идентификатор не изменяется), отсутствует возможность запуска программ через командный файл sudo, а также любых программ, обеспечивающих изменение эффективного идентификатора пользователя.

Пример применения аргументов командной команды запуска контейнера представлен на рисунке 3.

```
[admin@localhost test]$ docker run --user=2000:100 --security-opt=no-new-privileges -it t
est bash
[ivan@71e289e62d09 /]$ sudo passwd
sudo: effective uid is not 0, is /usr/bin/sudo on a file system with the 'nosuid' option
set or an NFS file system without root privileges?
[ivan@71e289e62d09 /]$
```

Рисунок 3. Ограничение превышения привилегий пользователя при запуске контейнера

Таким образом, для обеспечения возможности запуска контейнеров на основе произвольных подготовленных заранее пользовательских образов необходимо:

– в образе, являющимся основой для запуска контейнера, создать пользователя, имеющего идентификаторы, аналогичные таковым в суперкомпьютере;

– установить в момент запуска контейнера атрибут «--user», а в качестве его параметров передать идентификаторы пользователя, который инициировал запуск контейнера;

– в момент запуска контейнера установить параметр «--security-opt no-new-privileges», предотвращающий повышение пользователем своих привилегий внутри контейнера.

Обеспечение работоспособности коммуникационной библиотеки в контейнере

Одной из задач настоящей работы является предоставление пользователю возможности свободного выбора коммуникационной библиотеки «на его усмотрение» и затем её использования в проводимых вычислениях. Добавление коммуникационной библиотеки (как правило, MPI) в контейнер осуществляется созданием слоя образа Docker, в котором производится её установка. Установка MPI-библиотеки выполняется на стадии сборки образа после установки стека ПО, обеспечивающего поддержку высокоскоростных сетевых устройств. Подобный порядок создания слоёв образа позволяет коммуникационной библиотеке использовать в своей работе такие устройства.

При установке библиотеки в образ Docker необходимо добавить переменные окружения, указывающие путь к исполняемым файлам библиотеки, а также её вспомогательным файлам. Добавление переменных окружения осуществимо одним из трёх способов.

Первый способ реализуется добавлением в файл инициализации командной оболочки пользователя (например, в файл «.bashrc» при использовании оболочки bash), от имени которого будет осуществляться запуск задания, двух переменных окружения «PATH» и «LD_LIBRARY_PATH». Монтирование домашнего каталога пользователя в контейнер обеспечит загрузку для пользователя установленных переменных окружения.

Существует возможность установить переменные окружения в момент сборки образа Docker директивой «ENV», указанной в файле сценария создания образа. Подобный подход позволяет использовать переменные окружения любому пользователю, запустившему контейнер.

Переменные окружения в контейнере можно задать в момент его старта, указав параметр команды запуска «docker run»:

--env=[$ENV].

Однако, такой способ не является универсальным в случае автоматического запуска вычислительных заданий, представленных в виде контейнера, поскольку для каждой из библиотек MPI переменные окружения должны иметь различное значение.

Наиболее предпочтительным способом является установка переменных окружения в момент создания образа Docker, что позволяет добиться отсутствия лишних значений переменных окружения в файле инициализации командной оболочки.

Для обеспечения работоспособности коммуникационной библиотеки в контейнере, запущенном с применением механизма ограничения повышения пользовательских привилегий, необходимо гарантировать возможность взаимодействия контейнеров одного задания по транспортной и управляющей сетям суперкомпьютера.

Коммуникационные библиотеки используют в своей работе беспарольное подключение по протоколу ssh (либо по менее безопасному протоколу rsh), необходимое для обмена служебной информацией. Обмен информацией о вычислениях, производимых с использованием коммуникационной библиотеки, обеспечивается через устройства высокоскоростной сети.

Для каждого контейнера конкретного задания необходимо в момент его старта запустить сервер sshd, установка которого производится, как правило, в момент сборки базового образа. Запуск сервера sshd, если его производить по умолчанию, требует прав суперпользователя. Для инициализации сервера sshd от имени пользователя, не имеющего таких прав, в момент запуска контейнера необходимо обеспечить выполнение команды вида:

/usr/sbin/sshd -D -f /path_to/sshd_config -E /path_to_log/log

Аргумент параметра «-f» указывает на конфигурационный файл сервера. У пользователя, от имени которого запускается контейнер, должен быть доступ к этому файлу. Аргумент параметра «-E», указывающий куда выводить информацию о работе сервера, является необязательным параметром для запуска sshd. Параметр «-D» необходим для включения режима отладки, который позволяет при старте контейнера в фоновом режиме не завершать работу контейнера после запуска sshd-сервера. Стоит отметить, что следует запускать сервер sshd на нестандартном порту (порты 0–1024 являются привилегированными в системе, и их открытие требует прав суперпользователя), порт запуска указывается в конфигурационном файле сервера.

Для обеспечения возможности бесперебойного подключения клиентов по протоколу ssh следует установить в контейнере порт подключения клиента по умолчанию, аналогичный таковому в конфигурационном файле сервера, и отключить проверку соответствия ключей узлов IP-адресам. Операцию настройки необходимо выполнять на этапе сборки базового образа контейнера.

Макет суперкомпьютерной системы с представлением пользовательских заданий в виде контейнеров Docker

Для реализации возможности представления вычислительных заданий в виде контейнеров авторами было разработано программное средство, предоставляющее пользователю интерфейс создания образа контейнера и сохранения его в файл. Программное средство реализовано на языке python3.5 и содержится в python-библиотеке, подготовленной авторами. Для его использования пользователю необходимо выполнить команду:

task_build -d path/to/Dockerfile -t TAG [-h] [-o OUTFILE] [--rm].

Параметр ключа запуска «-d» определяет путь к сценарию сборки образа, исполь-

зуемому командой «docker build». По окончании сборки образа он по умолчанию сохраняется в локальном хранилище образов. Для сохранения образа в файл можно использовать ключ «-o», параметром которого является имя файла сохраняемого образа. После сохранения образа в файл существует возможность удалить собранный образ из локального хранилища, для этого необходимо в момент запуска командного файла task_build указать параметр «--rm».

Для экспериментальной проверки рассмотренных в статье подходов авторами был подготовлен макет суперкомпьютерной системы, построенный на базе сегмента установленного в МСЦ РАН суперкомпьютера МВС-100К. Сегмент включает три вычислительных модуля МВС-100К. Один из ВМ сегмента выступил в роли управляющей ЭВМ, остальные представляли вычислительные модули решающего поля суперкомпьютера. Подробно структура и состав макета рассмотрены авторами в [12]. В качестве системы управления заданиями в макете использована СУППЗ, для которой авторами были разработаны командные сценарии разворачивания представленного в виде контейнера пользовательского задания перед его стартом, а также сценарии сворачивания такого задания по его завершении. На макете были успешно отработаны как методика подготовки пользователем образа контейнера, так и задачи, связанные с обеспечением информационной безопасности. В частности, были отработаны возможные сценарии захвата прав суперпользователя изнутри контейнера, авторы убедились, что подобный захват на созданном макете невозможен.

Заключение

Для решения задачи представления пользовательского задания суперкомпьютера в виде Docker-контейнеров были исследованы ряд методов, способов и средств. Авторы остановили свой выбор на методе, предполагающим использование в супер-компьютерах коллективного пользования контейнеров с произвольным содержимым, формируемым самим пользователем. Была выработана методика формирования образа такого контейнера, включающая рекомендации по обеспечению работоспособности коммуникационной библиотеки MPI внутри контейнера. Особое внимание авторы уделили вопросам информационной безопасности, в частности предложили модель безопасности суперкомпьютерной системы, а также способы и средства, позволяющие предотвратить повышение пользователем своих привилегий при запуске процессов задания внутри контейнеров.

Авторы произвели практическую реализацию предложенных способов и средств для отечественной Системы управления прохождением параллельных заданий, эксплуатируемой в качестве СУЗ на суперкомпьютерных системах МСЦ РАН и ИПМ им. М.В. Келдыша РАН. Результатом стал программно-аппаратный макет суперкомпьютерной системы, обеспечивающий обработку представленных в виде Docker-контейнеров заданий, сформированных пользователями. Ближайшей перспективой работы является внедрение рассмотренных в статье подходов на суперкомпьютерах МСЦ РАН, входящих в состав оборудования центра коллективного пользования вычислительными ресурсами МСЦ РАН.

Публикация выполнена в рамках программы №27 Президиума РАН «Фундаментальные проблемы решения сложных практических задач с помощью суперкомпьютеров» по теме (проекту) «Исследование технологий виртуализации для создания цифровой платформы суперкомпьютерного проектирования и моделирования, ориентированной на субъекты цифровой экономики» (0065-2018-0407).

Tools for representation of user supercomputer jobs as Docker containers

G.L.Savin, P.N.Telegin, A.V.Baranov, A.S.Shitik

Abstract: Containerization (operating-system-level virtualization) is one of the key methods to increase the efficiency of using computational resources. In high-performance computing, containerization makes it possible to solve the problem of user jobs binary portability with minimal overhead. The article considers methods and tools, which allow preparing a supercomputer job in the form a Docker container with user-defined arbitrary content. The article also discusses the issues of information security when supercomputer jobs are represented in the form of containers.

Keywords: containerization, high performance computing, supercomputer jobs, Docker, workload manager, information security

Литература

1. Б.М.Шабанов, А.П.Овсянников, А.В.Баранов, С.А.Лещев, Б.В.Долгов, Д.Ю.Дербышев. Проект распределенной сети суперкомпьютерных центров коллективного пользования // Программные системы: теория и приложения. 2017. № 4(35). С. 245–262. DOI: 10.25209/2079-3316-2017-8-4-245-262

2. А.В.Баранов, А.В.Киселёв, В.В.Старичков, Р.П.Ионин, Д.С.Ляховец. Сравнение систем пакетной обработки с точки зрения организации промышленного счета. Научный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции (17-22 сентября 2012 г., г. Новороссийск). М.: Изд-во МГУ, 2012. С. 506-508. URL: http://agora.guru.ru/abrau2012/pdf/506.pdf (дата обращения: 12.04.2018)

3. A.Yoo, M.Jette, M.Grondona M. (2003) SLURM: Simple Linux Utility for Resource Management. In: D.Feitelson, L.Rudolph, U.Schwiegelshohn. (eds) Job Scheduling Strategies for Parallel Processing. JSSPP 2003. Lecture Notes in Computer Science, vol 2862. Springer, Berlin, Heidelberg. pp. 44-60. DOI: 10.1007/10968987_3

4. R.L.Henderson. (1995) Job scheduling under the Portable Batch System. In: Feitelson D.G., Rudolph L. (eds) Job Scheduling Strategies for Parallel Processing. JSSPP 1995. Lecture Notes in Computer Science, vol 949. Springer, Berlin, Heidelberg. pp. 279-294. DOI: 10.1007/3-540-60153-8_34

5. Moab HPC Suite Enterprise Edition. URL: http://www.adaptivecomputing.com/products/hpc-products/moab-hpc-suite-enterprise-edition (дата обращения 12.07.2018)

6. Система управления прохождением параллельных заданий (СУППЗ). Руководство программиста (пользователя). URL: http://www.jscc.ru/wp-content/uploads/2017/06/SUPPZ-user-guide-2016.pdf (дата обращения 23.08.2018)

7. А.В.Баранов, А.И.Тихомиров. Методы и средства организации глобальной очереди заданий в территориально распределенной вычислительной системе // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2017. Т. 6. № 4. С. 28-42. DOI: 10.14529/cmse170403

8. Oleg S. Aladyshev, Anton V. Baranov, Rem P. Ionin, Evgeny A. Kiselev, Boris M. Shabanov. Variants of deployment the high performance computing in clouds. 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). pp. 1453-1457. DOI: 10.1109/EIConRus.2018.8317371

9. И.В.Подорожный, А.Н.Светличный, А.В.Подлеснов. Введение в контейнеры, виртуальные машины и docker // Молодой ученый. 2016. №19. С. 49-53. URL: https://moluch.ru/archive/123/33873/ (дата обращения: 02.07.2018).

10. А.В.Баранов, Д.С.Николаев. Использование контейнерной виртуализации в организации высокопроизводительных вычислений // Программные системы: теория и приложения. 2016. Т. 7. № 1 (28). С. 117–134. DOI: 10.25209/2079-3316-2016-7-1-117-134

11. Д.С.Николаев, В.В.Корнеев. Использование механизмов контейнерной виртуализации в высокопроизводительных вычислительных комплексах с системой планирования заданий Slurm // Программная инженерия. 2017. Том 8. № 4. С. 157-160.

12. А.В.Баранов, А.С.Шитик. Способы и средства динамической реконфигурации сетей суперкомпьютера при представлении пользовательских заданий в виде контейнеров // Программные продукты, системы и алгоритмы, 2018, № 3, с. 62-70. DOI: 10.15827/2311-6749.18.3.8

13. В.А.Щапов, С.Р.Латыпов. Способы запуска задач на суперкомпьютере в изолированных окружениях с применением технологии контейнерной виртуализации Docker // Научно-технический вестник Поволжья. 2017. № 5. С. 172-177.

14. В.А.Щапов, А.В.Денисов, С.Р.Латыпов. Применение контейнерной виртуализации Docker для запуска задач на суперкомпьютере // Суперкомпьютерные дни в России: труды международной конференции (26-27 сентября 2016 г., г. Москва). М.: Изд-во МГУ, 2016. С. 505-511.

15. Intel® Omni-Path Architecture (Intel® OPA) Driving Exascale Computing and HPC with Intel. URL: https://www.intel.ru/content/www/ru/ru/high-performance-computing-fabrics/omni-path-driving-exascale-computing.html (дата обращения 02.07.2018)

16. Docker CE release notes. URL: https://docs.docker.com/release-notes/docker-ce/ (дата обращения 02.07.2018).

Методы и алгоритмы снижения фрагментации суперкомпьютерных ресурсов при планировании заданий

А.В.Баранов[1], С.А.Лещев[2] , Д.С.Ляховец[3]

[1,2] Межведомственный суперкомпьютерный центр РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

[3]ФГУП «НИИ «Квант», Москва, Россия,

E-mails: [1]antbar@mail.ru, [2]sergey.leshchev@jscc.ru, [3]anetto@inbox.ru

Аннотация: В статье рассматривается задача повышения эффективности использования вычислительных ресурсов суперкомпьютерных систем за счёт уменьшения их фрагментации. Причиной фрагментации является неделимость вычислительного модуля суперкомпьютера при планировании пользовательских заданий и, как следствие, – неполная загрузка выделенных заданию процессорных ядер. Одним из способов уменьшения фрагментации является формирования пакетов заданий. В статье приводится обзор существующих способов формирования пакетов заданий, предложен алгоритм формирования пакетов заданий и представлены предварительные оценки эффективности предложенного алгоритма.

Ключевые слова: планирование заданий, фрагментация вычислительных ресурсов, пакетирование заданий, суперкомпьютер

Введение

Современные суперкомпьютерные системы, как правило, представляют собой высокопроизводительные вычислительные установки (ВУ), основным компонентом которых является решающее поле вычислительных модулей (ВМ), объединённых высокоскоростной коммуникационной сетью. Вычислительный модуль включает в себя один или несколько многоядерных процессоров. Главной тенденцией развития вычислительной техники последнего десятилетия является постоянное увеличение числа ядер в новых микропроцессорах. По этому пути развития идёт и высокопроизводительная вычислительная техника, о чём свидетельствуют характеристики самых мощных суперкомпьютеров мира из рейтинга TOP500. В передовых суперкомпьютерных системах общее число ядер исчисляется десятками тысяч, а число ядер на вычислительном модуле – десятками. В ближайшие годы ожидается появление систем экзафлопсного уровня производительности, в которых тенденция увеличения числа процессоров и числа ядер на процессор будет проявляться ещё ярче.

В суперкомпьютерных системах коллективного доступа для производства высокопроизводительных расчётов пользователь должен оформить т.н. вычислительное задание, включающее в себя расчётную прикладную программу, требования к ресурсам и исходные данные для расчётов. Основным подходом при планировании заданий и выделении им суперкомпьютерных ресурсов в системах коллективного пользования является неделимость вычислительных модулей. Иными словами, отдельный вычислительный модуль независимо от числа наличествующих процессорных ядер, как правило, не разделяется между разными заданиями.

В случае если число используемых заданием процессорных ядер меньше общего числа ядер на выделенных ему системой планирования вычислительных модулях, возникает недозагрузка вычислительных ресурсов, называемая фрагментацией.

В настоящей работе авторы предприняли попытку исследования факторов, влияющих на фрагментацию ресурсов, а также методов и алгоритмов снижения фрагментации в суперкомпьютерных системах коллективного пользования.

Причины фрагментации вычислительных ресурсов

Распределение суперкомпьютерных ресурсов между пользовательскими заданиями осуществляет специальное программное обеспечение – система управления заданиями (СУЗ). Каждое задание из входного потока поступает в СУЗ, ожидает в очереди, после чего ему выделяются вычислительные ресурсы (подмножество вычислительных модулей решающего поля суперкомпьютера), на которых задание обрабатывается. По завершении задания СУЗ освобождает занятые им ресурсы. Подсистема статистики ведет учёт поступивших в систему заданий и их параметров,

время их старта и завершения. Процесс функционирования СУЗ представлен на рисунке 1.

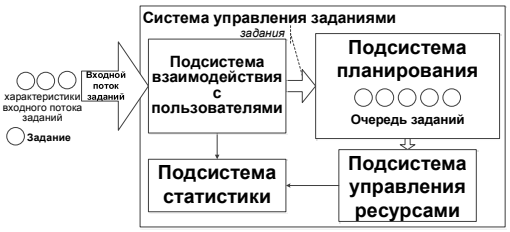


Рисунок 1. Обобщённая структура системы управления заданиями

Лидером мирового рейтинга самых высокопроизводительных вычислительных систем Top-500 [1] на июнь 2018 года является суперкомпьютер Summit [2], состоящий из 4 608 вычислительных модулей. Каждый модуль комплектуется двумя процессорами IBM Power9 и 6 графическими ускорителями NVIDIA Volta V100. Используемый в Summit процессор IBM Power9 имеет 22 ядра и поддерживает технологию Multi-Threading, позволяющую запускать по 4 потока на ядро.

Проанализируем рейтинг Top-500 июня 2018 года с точки зрения числа количества ядер в отдельном процессоре (таблица 1). Жирным шрифтом выделены те группы, для которых число ядер в процессоре является степенью двойки. Это будет важно для дальнейшего изложения. Отметим, что только у 156 из 500 (31%) суперкомпьютеров число ядер в процессоре является степенью двойки (8, 16, 32 или 64 ядра на процессор).

При формировании задания пользователь указывает требуемое для задания число процессорных ядер. Для некоторых заданий критичным являются не вычислительные ядра, а иной ресурс – например, оперативная память. Пользователь может для задания заказывать число вычислительных модулей, и на каждом модуле запускать произвольное число экземпляров программы. Например, для вычислительного модуля Summit (2 процессора по 22 ядра, всего 44 ядра) пользователь может запустить 2 потока (по одному на процессор), или 176 потоков (2 процессора по 22 ядра по 4 потока благодаря Multi-Threading), или 2000 потоков (например, потоки часто должны ожидать некоторых данных и не работают все одновременно).

Таблица 1. Количество суперкомпьютеров с указанным число ядер в процессоре для рейтинга Top-500 июня 2018 года

Число ядер в процессоре (cpu per socket)	Число суперкомпьютеров из рейтинга Top 500	Процент
16	**109**	**21,8%**
12	105	21%
20	69	13,8%
18	43	8,6%
10	43	8,6%
14	41	8,2%
8	**34**	**6,8%**
24	15	3%
68	11	2,2%
6	11	2,2%
64	**8**	**1,6%**
22	5	1%
32	**5**	**1%**
260	1	0,2%

Под модуль-часом будем понимать вычислительный ресурс, эквивалентный 1 вычислительному модулю, доступному для обработки задания в течение 1 часа. 1 модуль-час может означать, что 1 ВМ работал 1 час или 2 ВМ работали по 0,5 часов. Аналогично вводятся модуль-года. Если рассматривать работу ядер в ВМ, то возможно использовать ядро-часы и ядро-года.

Рассмотрим план запуска заданий для системы из 3 процессоров за некоторых период времени (рисунок 2).

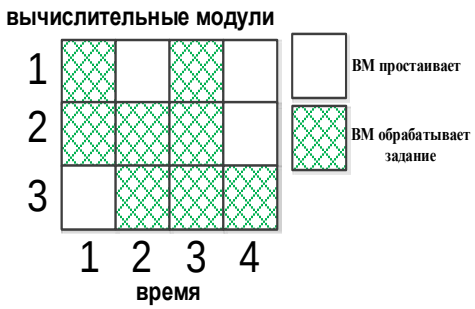


Рисунок 2. План запуска заданий

Белым обозначены простаивающие вычислительные модули, зеленой штриховкой – вычислительные модули, обрабатывающие пользовательские задания. Одним из показателей качества СУЗ является загрузка вычислителя или утилизация [3]. Формальное определение загрузки будет дано ниже. Будем понимать под загрузкой отношение обрабатывающих задание вычислительные модулей ко всем доступным ВМ за период. Для рисунка 2 загрузка составляет 8/12 ≈ 67%.

Рассмотрим подробнее тот же пример с учётом того, что каждый ВМ имеет 4 ядра. Пользователь может задействовать 1, 2, 3 или 4 ядра. Если пользователь для задания заказывает 6 ядер, то задание будет запущено на 2 ВМ, и из 8 ядер участвовать в обработки будут только 6. В большинстве современных СУЗ на 2 свободных ядрах не может быть запущено другое задание.

Пусть для простоты расчётов пользовательские задания используют только 1 ядро из 4 (рисунок 3). В этом случае обрабатывают задания 8 ядер из 48 доступных, а загрузка составляет 8/48 ≈ 17%.

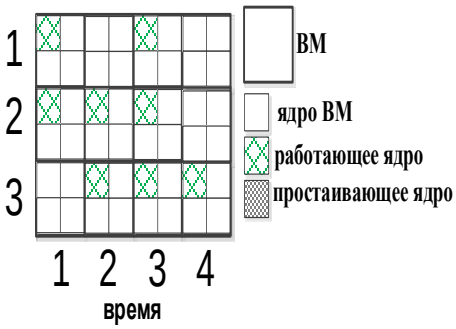


Рисунок 3. План запуска заданий с учётом загрузки ядер

Существует тенденция, что пользователи заказывают число ядер, соответствующее степени числа 2 [4], то есть 1, 2, 4, 8, 16, 32 и так далее. Если пользователь суперкомпьютера Summit (22 ядрами на ВМ) закажет для задания 32 ядра, то ему будет выделено 2 ВМ с общим числом ядер в 44. Таким образом, такое задание оставит простаивающими 12 ядер.

Пусть на ВУ, состоящей из M вычислительных модулей с k ядрами в каждом, за определённый интервал времени T было выполнено N заданий. Обозначим выделяемое для выполнения i-го задания число ядер как c_i, а время выполнения i-го задания как Δt_i. В этом случае фактическая загрузка ВУ S_{used} за интервал времени T может быть рассчитана по формуле

$$S_{used} = \sum_{i=1}^{N} c_i \cdot \Delta t_i \quad (1)$$

полная теоретическая загрузка ВУ S_{full} за интервал времени T рассчитывается по формуле

$$S_{full} = M \cdot k \cdot T \quad (2)$$

утилизация ВУ Q определяется по формуле

$$Q = \frac{S_{used}}{S_{full}} \quad (3)$$

как отношение фактической загрузки к её теоретическому максимуму. Следует отметить, что утилизация любой действующей ВУ никогда не будет равна единице из-за возникающих в процессе эксплуатации сбоев, выхода из строя части вычислительных модулей и простоев – как плановых для обслуживания ВУ, так и незапланированных.

Обозначим число выделенных, но неиспользованных i-м заданием ядер для как f_i, тогда объём S_{lost} неэффективно использованных вычислительных ресурсов (фрагментацию) можно определить по формуле

$$S_{lost} = \sum_{i=1}^{N} f_i \cdot \Delta t_i \quad (4)$$

эффективно использованных вычислительных ресурсов S_{work} – по формулам

$$S_{work} = \sum_{i=1}^{N} (c_i - f_i) \cdot \Delta t_i \quad (5)$$

Или

$$S_{work} = S_{used} - S_{lost} \quad (6)$$

а возникающий процент потерь из-за фрагментации F можно рассчитать по формуле

$$F = \frac{S_{lost}}{S_{used}} \cdot 100\% \quad (7)$$

Фрагментация вычислительных ресурсов МСЦ РАН

Основным научным оборудованием Межведомственного суперкомпьютерного центра РАН (МСЦ РАН) является уникальная научная установка – суперкомпьютер

МВС-10П, созданный в 2013 году и модернизированный в 2014-2017 годах. МВС-10П представляет собой гетерогенную вычислительную систему с водяным охлаждением суммарной производительностью 960 ТФЛОПС, состоящую из пяти разделов, основанных на разных архитектурных решениях. Каждый раздел может функционировать как отдельная ВУ с отдельной очередью задач пользователей, ориентированных на соответствующую архитектуру.

Раздел МВС-10П ОП Broadwell построенный на универсальных процессорах, наиболее востребованных пользователями, в том числе при решении задач с авторским кодом содержит 136 вычислительных узлов (4352 ядра) на процессорах Intel Xeon микроархитектуры Broadwell, объединённых коммуникационной сетью OmniPath 100 Гбит/с.

Раздел МВС-10П ОП Haswell содержит 42 вычислительных узла (1176 ядер) на процессорах Intel Xeon микроархитектуры Haswell, объединенных коммуникационной сетью OmniPath 100 Гбит/с.

Раздел МВС-10П «Торнадо» содержит 207 вычислительных узлов (28566 ядер), включающих процессоры Intel Xeon E5-2960 (микроархтектура Sandy Bridge) и Intel Xeon Phi 7110X (микроархитектура KNC). Для объединения узлов кластера в единое решающее поле используется коммуникационная сеть Infiniband FDR 56 Гбит/с.

Авторами была проанализирована статистика использования вычислительных ресурсов на разделах МВС-10П за трехмесячный период с апреля по июнь 2018 года, для этого для каждой исследуемой ВУ:

- по формуле (1) были просуммированы ядро-часы, занятые вычислительными заданиями (столбец ядро-часы работы в таблице 2);
- для заданий, которые заняли ядер меньше, чем им было выделено, по формуле (4) было рассчитано количество неэффективно израсходованных ядро-часов (столбец ядро-часы заказанные, но не использованные в таблице 2);
- по формуле (7) был определен процент потерь вычислительных ресурсов из-за фрагментации.

Результаты проведенных расчетов представлены в таблице 2.

Таблица 2. Потери вычислительных ресурсов суперкомпьютера МВС-10П из-за фрагментации

Раздел суперкомпьютера	Ядер в ВМ, k	Ядро-часы работы ВУ, S_{used}	Ядро-часы заказанные, но не использованные, S_{lost}	Потери из-за фрагментации, F
МВС-10П ОП Broadwell	32	8219539,73	318567,1	3,88%
МВС-10П ОП Haswell	28	1724428,5	374452,6	21,71%
МВС-10П Торнадо	16	3770120,8	44802,7	1,19%

Из таблицы 2 видно, что у двух вычислительных установок потери на фрагментацию составляют ~4%, а у одной вычислительной установки потери свыше 20%.

Это значит, что каждый пятый вычислительных модуль работает вхолостую, затрачивая электроэнергию, но не производя при этом полезной работы.

Прослеживается следующая тенденция – фрагментация существенно возрастает, если число ядер в ВМ не является степенью двойки, а для разделов, где число ядер в ВМ является степенью двойки, фрагментация растет с увеличением степени.

Пакетирование заданий как способ уменьшения фрагментации

Один из возможных способов уменьшения потерь от фрагментации – формирование пакетов заданий. Рассмотрим существующие способы формирования пакетов заданий.

В работе [5] описаны 2 различные стратегии пакетирования. Первая стратегия называется regular time interval, и пакет заданий формируется в фиксированные моменты времени, например, каждые 10 секунд.

Все поступившие за указанный период задания включаются в пакет. Согласно второй стратегии (fixed count) задаётся требуемый размер пакета k, и пакет заданий формируется, если при появлении очередного задания формируемый пакет достигает заданного размера в k заданий. Чтобы избежать длительного ожидания формирования пакета при отсутствии поступающих заданий во входном потоке, для второй стратегии предлагается ввести «максимальное время ожидания задания в очереди на формирования пакета», по

истечении которого пакет формируется из n заданий, где n < k.

В работе [4:6] предложен алгоритм группировки мелкозернистых заданий (fine-grained jobs) в крупнозернистые (coarse-grained jobs) для уменьшения издержек назначения заданий на узлы, которые включают выделение заданию вычислительных модулей, выполнение служебных действий перед запуском исполняемого файла пользователя, освобождение ВМ и завершение всех пользовательских процессов по завершении пользовательского исполняемого файла.

Для работы предложенного алгоритма необходимо задать во временных единицах (например, 10 минут) характеристику «размер зерна» (granularity size).

Задания будут формироваться в такие пакеты, чтобы обработка пакета на выделенных ресурсах заданной производительности не превысила заданный «размер зерна». Например, в один пакет могут быть включены три задания с длительностью обработки по 3 минуты, и их суммарное время обработки не превысит заданного ограничения в 10 минут на пакет.

Размер задания считается в MI – миллионах инструкций, производительность вычислительного ресурса – в MIPS. Существенным недостатком предлагаемого алгоритма является необходимость оценить каждое задание, определив требуемое количество инструкций для его обработки.

Работы [5] и [6] позволяют включить в один пакет заданий любые задания. В работах [3, 7] исследуется метод формирования заданий по типам. В этом случае во входном потоке заданий должна явно присутствовать информация о типе задания, что позволяет объединять однотипные задания в один пакет. При формировании пакета система пакетирования должна принять два решения: какое число заданий войдёт в очередной пакет, и какое число вычислительных модулей следует выделить для обработки пакета. Главным параметром при принятии этих решений является т.н. порог целесообразности, определяющий отношение времени обработки пакета заданий ко времени инициализации пакета.

В настоящей работе производится исследование, направленное на выявления различных типов заданий в потоке. Использование вычислительных установок предполагает три вида формируемых пользователем заданий:

1) выходные данные задания используются для формирования входных данных следующего – в этом случае новое задание поступает в очередь после выполнения предыдущего, такие задания не могут быть сгруппированы в пакет;

2) выходные данные задания не требуются для формирования входных данных следующего, т.к. в заданиях используются разные алгоритмы обработки данных – в этом случае в очереди может одновременно находится несколько существенно отличающихся друг от друга по требованиям к ресурсам и времени выполнения заданий пользователя, такие задания также не могут быть сгруппированы в пакет;

3) выходные данные задания не требуются для формирования входных данных следующего, в заданиях используются одинаковые алгоритмы обработки данных, различаются лишь наборы входных данных – в этом случае в очереди может одновременно находится несколько одинаковых по требованиям к ресурсам и времени выполнения заданий пользователя, такие задания могут быть сгруппированы в пакеты.

Задания, сформированные одним и тем же пользователем, в которых используются одинаковые алгоритмы обработки данных, требующие для выполнения одинаковых ресурсов ВУ и, как следствие, имеющие небольшие различия во времени выполнения далее будем называть однотипными.

Разработаем алгоритм выделения групп однотипных заданий и объединения их в пакеты, а затем произведем оценку сверху возможного выигрыша от применения системы пакетирования однотипных заданий.

Алгоритм формирования групп однотипных заданий

Собираемая в МСЦ РАН статистика использования вычислительных ресурсов включает в том числе следующие параметры заданий пользователей:

а) время постановки в очередь t^{queue};

б) время старта t^{start};

в) время завершения t^{stop};

г) имя пользователя (login);

д) имя исполняемого файла (taskname);

е) заказанное время выполнения задания (ntime);

ж) заказанное количество процессорных ядер $cores^{used}$;

з) фактически использованное количество процессорных ядер $cores^{work}$.

Необходимыми признаками однотипных заданий является совпадение у них имени пользователя, заказанного времени выполнения и заказанного количества процессорных ядер.

Предположим, что признаком одного и того же алгоритма обработки данных, используемого пользователем в задании, с высокой вероятностью будет являться одинаковое имя исполняемого файла из проектной директории пользователя, т.к. общеупотребимой практикой работы пользователей на ВУ при разработке нового алгоритма является присвоение исполняемому файлу с реализующим его кодом уникального имени, а при изменении алгоритма и, как следствие, кода программы, его реализующей – добавление к имени исполняемого файла суффиксов, при этом имя нового исполняемого файла уникальность сохраняет.

В результате получим, что признаками заданий, которые с высокой вероятностью окажутся однотипными и могут быть сгруппированы в пакет, являются совпадающие параметры login, ntime, ntime, $cores^{used}$ и taskname.

Также учтём, что время постановки в очередь последнего задания в группе не должно превышать времени завершения любого предыдущего задания в группе для гарантии того что при формировании групп в них не попадут задания, входными данными которых могут быть выходные данные заданий группы, завершившихся ранее.

Из накопленных подсистемой статистики за исследуемый интервал времени данных сформируем выборку информации о завершенных заданиях, у которых количество фактически использованных процессорных ядер оказалось меньше заказанного количества процессорных ядер. Полученную выборку упорядочим по возрастанию вложенной сортировкой по значениям имени пользователя, заказанного времени выполнения, заказанного количества процессорных ядер, имени исполняемого файла и времени постановки в очередь. Далее выделим группы однотипных заданий и отделим их от задач, не подлежащих группировке по сформулированным выше признакам, в соответствии с алгоритмом, блок-схема которого приведена на рисунке 4.

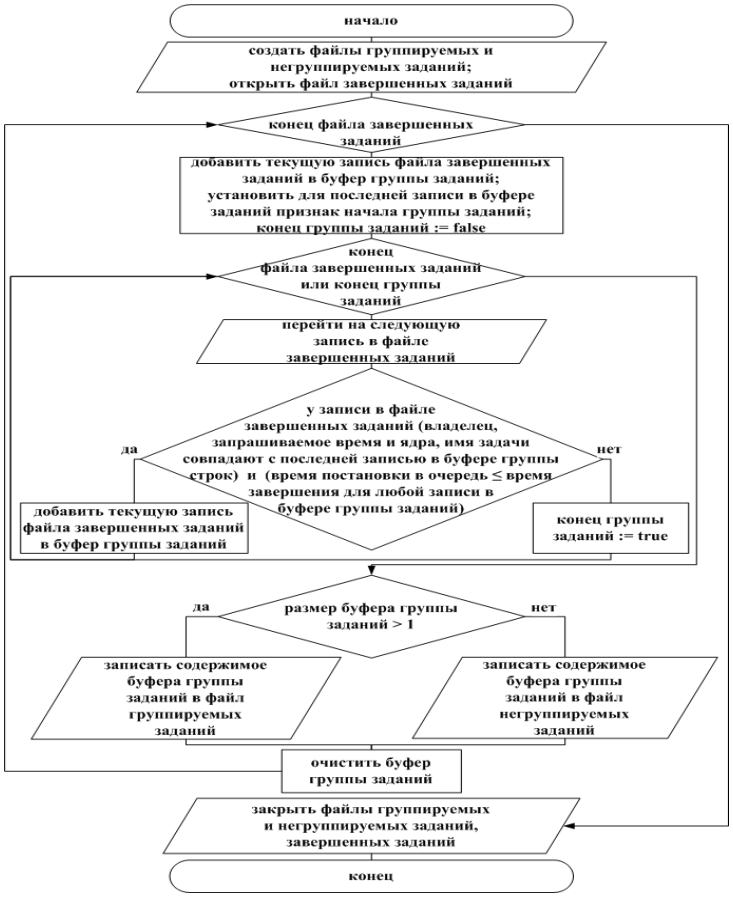


Рисунок 4. Блок-схема алгоритма формирования групп заданий

Выделение групп производится следующим образом: отсортированный файл с записями о завершенных заданиях проходится последовательно в цикле с предусловием, контролирующим завершение файла записей. Т.к. записи в файле упорядочены по

признакам группы, формирование группы сводится к заполнению буфера группы заданий во вложенном цикле, выход из которого контролируется при помощи логической служебной переменной «конец группы заданий». Перед началом работы вложенного цикла в буфер группы заданий добавляется первая запись группы, в теле записи выставляется признак того, что это первое задание в группе, производится инициализация служебной переменной выхода из цикла. В теле вложенного цикла в файле с записями о завершённых заданиях указатель переводится на следующую запись, после чего проверяются 2 условия:

- совпадают ли для этой записи внешние признаки формируемой группы – поля владельца задания, заказанного времени выполнения, запрошенного времени выполнения, заказанного количества процессорных ядер и имени исполняемого файла;

- не превышает ли время постановки в очередь текущего задания время завершения какого-либо из уже накопленных в буфере заданий.

Для проверки второго условия используется вызов специализированной функции, не детализированный на блок-схеме алгоритма. Если результат проверки оказывается положительным – текущая запись о задании из файла добавляется в буфер и цикл продолжается, если отрицательным – производится выход из вложенного цикла. После этого проверяется количество записей о заданиях в буфере группы заданий. Если оно превышает единицу, то в буфере находится сформированная группа, и она выводится в файл группируемых заданий, если нет – то группу сформировать не удалось, и находящаяся в буфере запись о задании помещается в файл негруппируемых заданий, после чего буфер очищается и продолжается цикл обхода файла с записями о завершённых заданиях.

Следует отметить, что разработанный алгоритм не учитывает множество рабочих ситуаций, возникающих при эксплуатации ВУ, в частности в нём не предусмотрено исключение заданий, формально имеющих признаки однотипных, но выполнение которых было завершено с ошибкой либо за очень короткий срок.

Одним из основных препятствий при совершенствовании алгоритмов планирования пользовательских заданий является необходимость принятия решения о пакетировании находящихся в очереди заданий в условиях недостатка у них формальных признаков, позволяющих выделить однотипные задания, в то же время пользователь, как правило, обладает необходимой полнотой информации о своих заданиях и способен принять решение об объединении однотипных заданий в пакет.

Эффективность применения пакетирования однотипных заданий

После работы алгоритма в файле группируемых заданий находится M групп G однотипных заданий, для каждой группы нам известен её размер N^g.

При объединении заданий в группу время выполнения i-й группы Δt_i^g будет равняться максимальному времени выполнения задания в группе

$$\Delta t_i^g = max\{\Delta t_j\}, \quad \text{где } j \in (1, N_i^g) \tag{8}$$

а количество выделяемых для группы заданий ядер c_i^g– округлённому вверх ближайшему целому от суммы всех выделенных для заданий группы ядер, кратному числу k ядер в модуле

$$c_i^g = \lceil \sum_{j=1}^{N_i^g} c_j^g : k \rceil \tag{9}$$

Тогда затраты ВУ на выполнение групп однотипных заданий в пакетах определяются по формуле

$$S_{used}^2 = \sum_{i=1}^{M} c_i^g \cdot \Delta t_i^g \tag{10}$$

По формуле (1) рассчитаем для каждой ВУ ее затраты на выполнение найденных однотипных заданий S_{used}^1 до пакетирования, по формуле (10) – затраты на выполнение найденных однотипных заданий S_{used}^2 после пакетирования, и определим количество затрат ВУ ΔS_{used}, сэкономленных при пакетировании однотипных заданий, по формуле

$$\Delta S_{used} = S_{used}^1 - S_{used}^2 \tag{11}$$

Выигрыш от пакетирования W для каждой ВУ рассчитаем как

$$W = \frac{\Delta S_{used}}{S_{used}} \cdot 100\% \tag{12}$$

где S_{used} рассчитывается по формуле (1) для каждой ВУ. Результаты проведенных расчетов представлены в таблице 3.

Таблица 3. Выигрыш от применения пакетирования однотипных заданий

ВУ	Ядро-часы, сэкономленные при пакетировании, ΔS_{used}	Выигрыш от пакетирования, W
МВС-10П ОП Broadwell	218869	2,66 %
МВС-10П ОП Haswell	81672	4,74 %
МВС-10П Торнадо	781	0,02 %

Заключение

Переведя данные об экономии вычислительных ресурсов, которую может дать нам объединение однотипных заданий пользователя в пакеты, из ядро-часов в модуль-дни мы видим, что для ВУ МВС-10П ОП Haswell может быть сэкономлено 121,5 модуль-дня, а для ВУ МВС-10П ОП Broadwell – почти 285 модуль-дней, что при 42-х и 136 модулях на установку составляет в первом случае почти три дня, а во втором – немногим более двух дней работы целой вычислительной установки. Это существенный резерв для повышения утилизации вычислительных установок, эксплуатируемых в МСЦ РАН.

Следовательно, имеет смысл продолжить исследования методов снижения фрагментации ресурсов и планирования пользовательских заданий, разработав для пользователя командный механизм взаимодействия с системой управления заданиями для объединения заданий в пакеты и оценив эффективность действия такого механизма.

Публикация выполнена в рамках программы №26 Президиума РАН «Фундаментальные основы создания алгоритмов и программного обеспечения для перспективных сверхвысокопроизводительных вычислений» по теме (проекту) «Исследование методов и алгоритмов функционирования систем планирования и управления заданиями сверхвысокопроизводительных вычислений для повышения эффективности работы современных суперкомпьютерных центров» (0065-2018-0408).

Methods and Algorithms for Reducing Supercomputer Resources Fragmentation when Scheduling Jobs

F.V.Baranov, S.A.Leshchev, D.S.Lyakhovets

Abstract: The article discusses the problem of increasing efficiency by reducing fragmentation when using supercomputer resources. The cause of fragmentation is the indivisibility of the supercomputer nodes when scheduling user jobs. This results incomplete load of the processor cores allocated to user jobs. One way to reduce fragmentation is jobs packaging. The article provides an overview of the existing methods of forming job packages proposes the job packages forming algorithm and presents preliminary estimates of the proposed algorithm effectiveness.

Keywords: jobs packaging system, computing resources fragmentation, jobs scheduling efficiency, HPC, supercomputer

Литература

1. TOP500. The List. URL: https://top500.org (дата обращения: 02.11.2018).

2. Summit. Oak Ridge National Laboratory. URL: https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/ (дата обращения: 02.11.2018).

3. А.В.Баранов, Д.С.Ляховец. Влияние пакетирования на эффективность планирования параллельных заданий // Программные системы: теория и приложения. 2017. Т. 8. № 1. С. 193-208.

4. W.Cirne, F.Berman. A Model for Moldable Supercomputer Jobs // Proc. 15th Intern. Parallel and Distributed Processing Symposium (IPDPS 2001). 23 – 27 April 2001. San Francisco, California, USA. 2001. P. 59 – 79.

5. M.Maheswaran. Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems (1999) // M. Maheswaran, S. Ali et al. // Journal of Parallel and Distributed Computing. — 1999. — Vol. 59, — No. 2. — P. 107–131.

6. N.Muthuvelu. A Dynamic Job Grouping-Based Scheduling for Deploying Applications with Fine-Grained Tasks on Global Grids (2005) // N. Muthuvelu, J. Liu et al. // Grid Computing and e-Research (AusGrid 2005): Proceedings of the 3rd Australasian Workshop (Newcastle, NSW, Australia, January 30 – February 4, 2005). — Australian Computer Society, 2005. — P. 41–48.

7. А.В.Баранов, Д.С.Ляховец. Экспериментальные оценки влияния пакетирования на некоторые показатели эффективности планирования параллельных заданий// Программные продукты, системы и алгоритмы. 2017. № 3. С. 1-8.

8. Д.М.Емельянов, П.А.Потехин, В.В.Топорков. Формирование пакетов заданий в грид с учетом предпочтений пользователей // Открытое образование. 2016. №3 С.4-8.

9. В.В.Топорков, Д.М.Емельянов, П.А.Потехин. Формирование и планирование пакетов заданий в распределенных вычислительных средах (2015)// Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2015. №2. С.44-57.

Организация виртуальной учебной лаборатории параллельного программирования на базе платформы виртуализации Proxmox VE

Ф.А.Аникеев[1], А.В.Баранов[2], Ф.С.Зайцев[3], Е.А.Киселёв[4], С.А.Лещев[5]

[1,3] ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

[2,4,5] Межведомственный суперкомпьютерный центр РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mails: [1]snowfed@gmail.com, [2]antbar@mail.ru, [3]fza@mail.ru, [4]kiselev@jscc.ru, [5]sergey.leshchev@jscc.ru

Аннотация: В статье рассматривается оптимальный подход к построению надёжной вычислительной инфраструктуры для виртуальной учебной лаборатории. Созданная виртуальная лаборатория успешно апробирована на Межфакультетской кафедре высокопроизводительных вычислений МГУ имени М.В. Ломоносова для эффективного обучения студентов параллельному программированию. Виртуальная инфраструктура лаборатории построена на основе программной платформы виртуализации Proxmox VE, свободно распространяемой в открытых исходных текстах. В статье рассмотрена методика создания виртуальной инфраструктуры, выделены ключевые моменты в подготовке и настройке виртуальной среды и виртуальных машин. Предложенный подход реализован на базе суперкомпьютера МВС-10П МП Petastream Межведомственного суперкомпьютерного центра – филиала НИИСИ РАН.

Ключевые слова: виртуальная инфраструктура, параллельное программирование, Proxmox VE, суперкомпьютер

Введение

Важным аспектом развития суперкомпьютерного образования является постоянное совершенствование и актуализация учебно-методической и элементной базы для проведения всех форм и видов обучения. Возможности использования высокопроизводительных систем для обучения часто ограничены из-за их высокой загруженности и дороговизны. Одним из путей преодоления таких ограничений является создание виртуальных вычислительных систем, обладающих сходным составом программного обеспечения и достаточным для проведения учебных занятий количеством виртуальных вычислителей.

Технология виртуализации позволяет полноценно обучать программированию на суперкомпьютерах и экономить при этом значительные средства. В типичных условиях аппаратное и программное оснащение терминальных классов заметно отстаёт от передового уровня развития вычислительной техники и поэтому не даёт возможности изучения в полной мере новых технологий параллельного программирования. Однако вместо дорогостоящего апгрейда десятков морально устаревших компьютеров их можно использовать в качестве виртуальных рабочих столов, обеспечивающих доступ к самым новым суперкомпьютерам, программистским средам и параллельным вычислительным методикам. Виртуальный рабочий стол не требует больших вычислительных ресурсов и его легко создать даже на планшете или смартфоне.

Кроме того, виртуализация позволяет существенно снизить накладные расходы на подготовку и поддержку суперкомпьютерного практикума. Так достаточно создать одну виртуальную машину с нужным программным обеспечением и затем размножить её копированием в требуемом числе экземпляров.

Преимущества виртуализации подтверждены авторами статьи совместно с Е.П. Сучковым, А.Г. Шишкиным и С.В. Степановым в реальном учебном процессе по обучению суперкомпьютерным технологиям на Межфакультетской кафедре высокопроизводительных вычислений МГУ имени М.В. Ломоносова. Заведующий кафедрой – научный руководитель НИИСИ РАН, академик В.Б. Бетелин.

Положительный опыт применения виртуализации в учебном процессе имеется и в других учебных заведениях. Например, в Южно-Уральском государственном университете создана облачная образовательная плат-

форма «Персональный виртуальный компьютер», использовавшая для своего выполнения суперкомпьютеры ЮУрГУ «СКИФ-Аврора» и «СКИФ Урал» [1]. Основу этой платформы составили гипервизоры Microsoft Hyper-V и инфраструктура рабочих столов Citrix XenDesktop VDI Edition.

Однако авторами настоящей статьи для создания виртуальной инфраструктуры, в отличие от работы [1], использована другая методика виртуализации. Её выбор основан на проведённых в Межведомственном суперкомпьютерном центре Российской академии наук (МСЦ РАН) научно-исследовательских работах [2-4] по вопросам построения виртуальных вычислительных систем для организации высокопроизводительных вычислений. Наиболее подходящей для виртуальной лаборатории по обучению в области параллельного программирования представляется свободно распространяемая платформа виртуализации Proxmox VE [5].

Платформа Proxmox уже нашла применение для учебных целей. Так, в работе [6] Proxmox используется как часть реконфигурируемой и отказоустойчивой платформы для создания открытых онлайн-курсов и проведения лабораторных работ. В [6] рассмотрено взаимодействие Интернет-ресурса и виртуального окружения Proxmox VE, однако не раскрыты процесс подготовки виртуального окружения Proxmox VE, его состав и характеристики используемого оборудования. Именно этим вопросам уделено основное внимание авторов настоящей статьи. Здесь представлены основные этапы и методика построения инфраструктуры для организации виртуальной учебной лаборатории на базе суперкомпьютерных ресурсов МСЦ НИИСИ РАН, описаны особенности функционирования построенной инфраструктуры.

Выбор платформы виртуализации

Основными инструментами для создания виртуальных инфраструктур являются гипервизоры и контейнеры. Преимуществом гипервизоров по сравнению с контейнерами является возможность одновременного выполнения на одном вычислительном узле виртуальных машин с несколькими отличными друг от друга гостевыми операционными системами (ОС). Поскольку создаваемая инфраструктура для виртуальной учебной лаборатории предполагает универсальное использование, в т.ч. виртуальных машин (ВМ) с ОС как семейства Linux, так и семейства Microsoft Windows, в качестве технологии для создания инфраструктуры была выбрана гипервизорная виртуализация.

Поскольку бюджеты научных и образовательных учреждений обычно сильно ограничены, использование платного программного обеспечения для построения виртуальных сред нецелесообразно. Создаваемая авторами виртуальная инфраструктура предназначена для использования исключительно в образовательных целях, её функционирование не требует регулярных обновлений, а уровень квалификации сотрудников МСЦ РАН позволяет администрировать виртуальную инфраструктуру без обращения в службы технической поддержки производителей программного обеспечения. С учётом перечисленных особенностей, для построения виртуальной инфраструктуры было решено выбрать гипервизор KVM.

Следует отметить, что выбор гипервизора всегда сопровождается выбором программных средств управления виртуальной вычислительной средой: виртуальными машинами, серверами виртуализации, гипервизорами, учётными записями пользователей, файловым хранилищем.

Для работы с гипервизором KVM используют следующие системы управления виртуальной средой, распространяемые в открытых исходных текстах: Virtual Machine Manager [7], Proxmox Virtual Environment [8], AQEMU [9], GKVM [10], Cloudmin [11], oVirt [12].

Virtual Machine Manager входит в состав всех репозиториев ОС семейства Linux и представляет собой приложение, позволяющее управлять гипервизорами KVM и Xen, а также виртуальными машинами на одном или нескольких серверах виртуализации.

Proxmox Virtual Environment (VE) – комплекс программного обеспечения для построения отказоустойчивых виртуальных сред с централизованной моделью управления через web-интерфейс. Комплекс Proxmox VE распространяется в виде отдельного дистрибутива на основе ОС Debian Linux, но может быть установлен и на другие ОС семейства Linux.

AQEMU – графическая оболочка управления виртуальными машинами QEMU и KVM. AQEMU разрабатывалась как более удобная и простая в использовании альтернатива Virtual Machine Manager. Её отличительной особенностью является возможность генерации командных сценариев (скриптов) для автоматизации запуска

виртуальных машин с требуемыми настройками.

GKVM – графическая оболочка гипервизора KVM для ОС семейства Linux. Её отличительной особенностью является минимальный объём занимаемого дискового пространства после установки.

Cloudmin – web-интерфейс для централизованного управления виртуальными машинами и гипервизорами Xen, OpenVZ, KVM, VServers, Solaris Zones, Amazon EC2. Бесплатная версия данного программного обеспечения позволяет управлять гипервизором Xen или KVM на одном сервере. В платной версии реализована возможность одновременного управления несколькими гипервизорами Xen, OpenVZ, KVM, Vservers, EC2 и Solaris Zone.

OVirt – web-интерфейс для централизованного управления виртуальными машинами и гипервизорами KVM, развивающийся открытым сообществом разработчиков как альтернатива VMware vSphere. Отличительной особенностью этого программного обеспечения является возможность установки и запуска с флеш-накопителя объёмом 64 Мбайта.

Для работы всех перечисленных программных средств, за исключением Proxmox VE, Cloudmin и oVirt, необходимо наличие графической оболочки. Программные средства AQEMU, GKVM, QtEmu обладают функционалом в большей степени ориентированным на обычные рабочие станции пользователей, что сильно затрудняет их использование для реализации масштабируемых виртуальных систем. При выборе системы управления виртуальной средой из Proxmox VE, Cloudmin и oVirt особое внимание обращалось на совместимость с различными видами дистрибутивов ОС семейства Linux и наличием в бесплатной версии программного продукта необходимого функционала для развёртывания и управления виртуальной вычислительной средой на нескольких физических серверах. Среди перечисленных программных средств указанными свойствами обладал только комплекс программного обеспечения Proxmox VE, который и был выбран в качестве основного средства построения инфраструктуры для виртуальной учебной лаборатории.

В Proxmox VE каждая виртуальная машина представлена двумя видами файлов: файлом с настройками ВМ и файлами виртуального диска. Файл с настройками ВМ может создаваться отдельно и использоваться для подключения уже имеющихся образов диска. Образ виртуального диска является файлом или группой файлов, в которых виртуальная машина хранит свои данные. В Proxmox VE могут использоваться несколько различных форматов образов виртуальных дисков: raw, qcow2 и vmdk.

Для построения инфраструктуры виртуальной учебной лаборатории авторами был выбран формат qcow2 – формат гипервизора KVM, основным преимуществом которого является возможность автоматического увеличения размера файла-образа ВМ по мере его заполнения, а также поддержка механизма снапшотов. По сравнению с остальными альтернативами формат qcow2 позволяет экономнее расходовать дисковое пространство системы хранения, но при этом обладает достаточной производительностью для решения задач обучения. Кроме этого, формат qcow2 является одним из рекомендованных для ВМ в среде Proxmox.

Порядок установки и настройки виртуальной инфраструктуры

При создании виртуальной инфраструктуры для проведения учебных занятий в качестве базового использовался подход [3] с рядом изменений, отражающих специфику настоящего проекта. Для построения виртуальной инфраструктуры было выделено:

– 4 вычислительных узла (hps05-hps08) из состава суперкомпьютера МВС-10П МП Petastream [13];

– дисковое пространство подсистемы хранения данных на базе NetApp FAS под управлением ОС 7-mode Data ONTAP (далее – 7-mode);

– дисковое пространство подсистемы хранения данных на базе NetApp FAS под управлением ОС Clustered Data ONTAP (далее – CDOT);

– часть ресурсов VPN-концентратора существующей сетевой инфраструктуры МСЦ РАН.

Для организации сетевого взаимодействия и управления виртуальной средой было создано две виртуальных локальных (VLAN-сегмента) сети:

– сеть управления и доступа для удалённого управления через сеть Интернет виртуальными машинами и гипервизорами, организации сетевого взаимодействия ВМ между собой и доступа студентов к удалённым рабочим столам ВМ;

– сеть хранения и передачи данных для организации доступа гипервизоров к

хранилищам виртуальных машин, размещённых на подсистемах хранения данных.

Общая структура инфраструктуры для виртуальной учебной лаборатории представлена на рисунке 1.

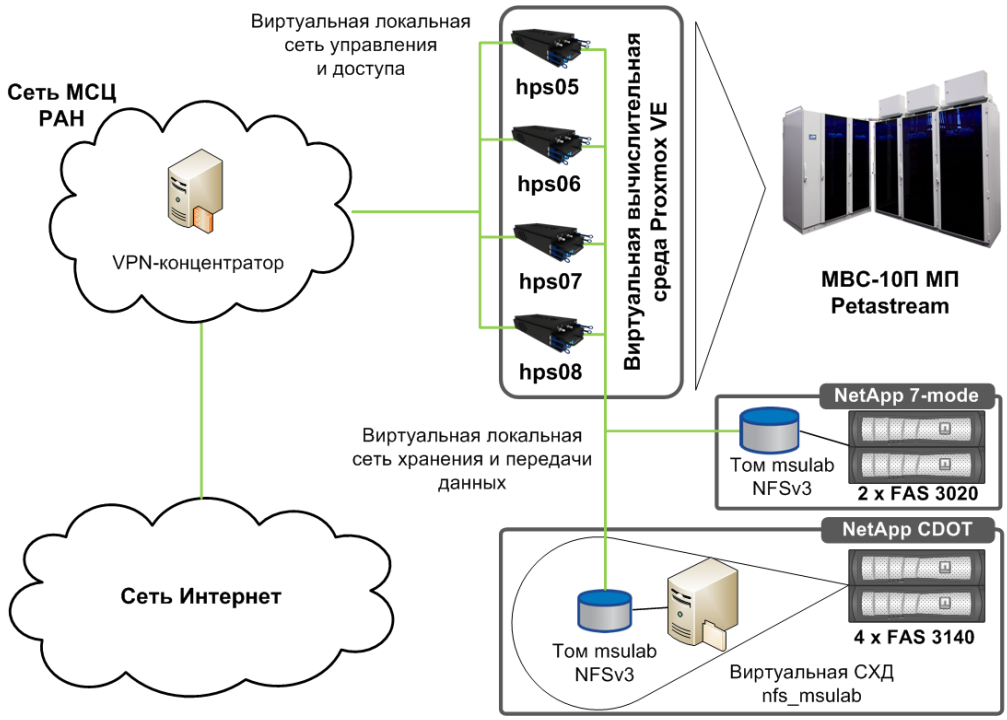


Рисунок 1. Общая структура виртуальной среды

Обязательным требованием создания отказоустойчивой виртуальной среды является наличие дискового пространства для хранения виртуальных машин, разделяемого между всеми гипервизорами. Дисковое пространство для проекта в размере 2 ТВ первоначально было выделено на подсистеме хранения данных 7-mode NetApp FAS 3020, доступ к сетевому каталогу осуществлялся по протоколу NFSv3. Рост требований к производительности и объёму дискового пространства во время эксплуатации виртуальной среды потребовал изменения подхода к выделению дискового пространства, поэтому тела виртуальных машин были перемещены в сетевой каталог, организованный на более производительной подсистеме CDOT NetApp FAS 3140 с тем же файловым протоколом доступа NFSv3. Для более эффективной утилизации дискового пространства и работы механизма кэширования чтения на выделенном томе была задействована блочная дедупликация, выполняемая по расписанию еженедельно в выходной день. Дедупликация обеспечила экономию дискового пространства в 79% (1,29 ТБ).

На каждом вычислительном узле суперкомпьютера МВС-10П МП Petastream была установлена ОС Proxmox VE 5.1 и выполнена процедура объединения узлов в кластер. Объединение вычислительных узлов в кластер ОС Proxmox необходимо для повышения отказоустойчивости ВМ и организации балансировки нагрузки.

Необходимое число вычислительных узлов кластера было определено исходя из требований сохранения работоспособности кластера при отказе либо временном выводе из эксплуатации для обслуживания одного из узлов.

Для каждой ВМ через графический интерфейс ОС Proxmox можно задать порядок загрузки ВМ, правила миграции ВМ при сбоях и активировать или деактивировать балансировку нагрузки.

Поскольку балансировка нагрузки может приводить к деградации производительности, увеличению нагрузки на процессорные ядра, дисковую подсистему и сеть при миграции ВМ с одного на другой вычислительный узел, данная настройка была отключена для всех виртуальных машин, кроме сервисных.

Для построения виртуальной инфраструктуры были созданы два типа виртуальных машин:

– сервисные – для организации сетевого доступа и функционирования ВМ;

– рабочие станции – виртуальные рабочие станции для проведения лабораторного практикума.

К сервисным ВМ были отнесены DHCP- и DNS-сервер под управлением ОС Linux, файловый сервер для обмена данными между пользователями ВМ, совмещённый с ActiveDirectory, сервер лицензий для ПО Microsoft Visual Studio 2015 и Intel Parallel Studio XE 2018 под управлением ОС Windows Server. Для сервисных ВМ были разрешена миграция и выставлен наивысший приоритет для загрузки. Для каждой сервисной ВМ под управлением ОС Linux были выделены 1 виртуальное ядро и 1 ГБ оперативной памяти, для ВМ под управлением ОС Windows Server – 2 виртуальных ядра и 4 ГБ оперативной памяти. Сетевые интерфейсы всех сервисных ВМ были размещены внутри сети управления и доступа. Компонентная схема созданной виртуальной инфраструктуры представлена на рисунке 2.

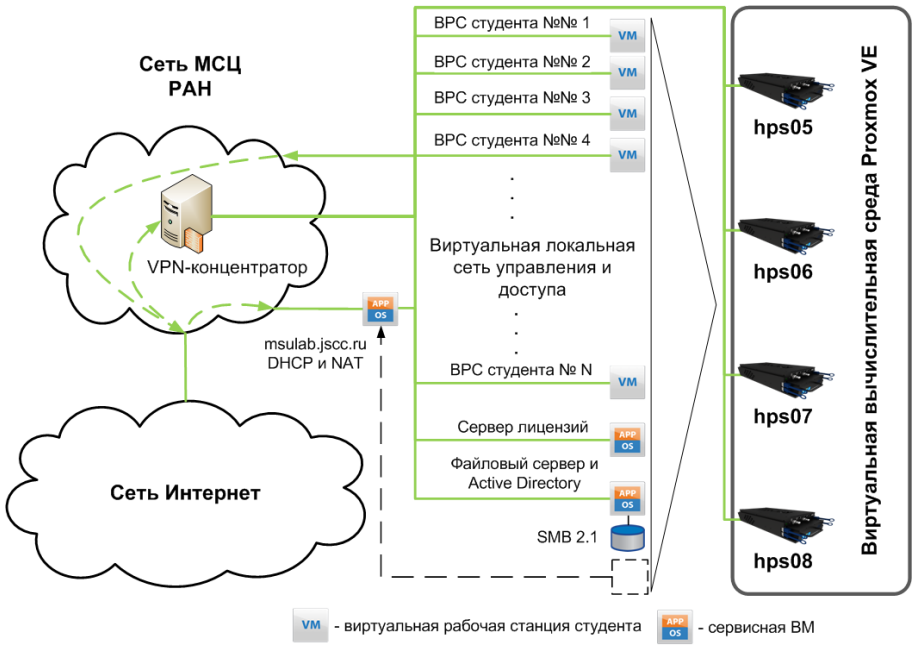


Рисунок 2. Компоненты виртуальной инфраструктуры

При создании виртуальной среды для организации и проведения лабораторного практикума у преподавателя должна быть возможность раздавать студентам учебные материалы на занятиях. Для этих целей внутри сети управления и доступа средствами сервисной ВМ файлового сервера под управлением ОС Microsoft Windows Server был создан сетевой каталог с правами доступа на чтение для студентов и полным доступом для преподавателей.

В роли рабочих станций студентов выступили ВМ под управлением ОС Windows 10. Эталонный образ виртуальной рабочей станции был создан и настроен в ПО VMware Player. Подготовленный образ ВМ был конвертирован в формат qcow2 с использованием штатного программного средства qemu-img, входящего в состав ОС Proxmox, и размножен в 25-ти экземплярах – по числу студентов и преподавателей, работающих через web-интерфейс ОС Proxmox.

Для каждой виртуальной рабочей станции выделено по 2 виртуальных ядра и 4 ГБ оперативной памяти. Сетевые интерфейсы всех виртуальных рабочих станций размещены внутри сети управления и доступа. Для каждого студента выделена собственная ВМ и создана персональная локальная учётная запись. В целях обеспечения отказоустойчивости для каждой ВМ средствами Proxmox VE настроено расписание автоматического создания резервной копии после окончания занятий.

Сетевой доступ к виртуальным рабочим станциям был организован посредством технологии NAT через сервисную ВМ, выполняющую функции DHCP- и DNS-сервера, один из сетевых интерфейсов которой размещён в сети Интернет, а второй – внутри сети управления и доступа. Доступ к виртуальным рабочим станциям организован по протоколу RDP (Remote Desktop Protocol) и может быть осуществлён с любой рабочей станции, планшета или смартфона.

Для организации обратного сетевого маршрута при доступе к виртуальным рабочим станциям и возможности выхода в

Интернет при удаленной работе на них для сети управления и доступа была организована маршрутизация трафика через существующую сетевую инфраструктуру МСЦ РАН.

Управление элементами виртуальной инфраструктуры осуществляется через web-интерфейс Proxmox с использованием отдельных учётных записей сервера LDAP МСЦ РАН. Каждой учётной записи средствами Proxmox VE назначена одна из ролей: администратор виртуальной инфраструктуры или администратор виртуальных машин. Роль администратора виртуальной инфраструктуры подразумевает полный доступ ко всем элементам управления виртуальной средой Proxmox VE и назначается специально выделенным сотрудникам МСЦ РАН. Роль администратора виртуальных машин назначается учётным записям преподавателей и позволяет управлять набором виртуальных машин, заданным администратором виртуальной инфраструктуры. Для всех администраторов средствами VPN-концентратора сетевой инфраструктуры МСЦ РАН был организован VPN-доступ в сеть управления и доступа.

Порядок настройки виртуальных рабочих станций

Для проведения лабораторного практикума на виртуальные рабочие станции студентов установлено следующее программное обеспечение:

– Notepad++;

– Microsoft Visual Studio 2015 Community Edition;

– Intel Parallel Studio XE 2018 Cluster Edition;

– Microsoft Visual Studio 2008 Professional Edition;

– NVIDIA CUDA Toolkit 2.3;

– NVIDIA SDK 2.3.

Последние три пункта нужны для использования CUDA-эмулятора на компьютерах без видеокарт NVIDIA.

Intel Parallel Studio предоставляется с лицензией на ограниченное число пользователей, лицензирование проводится через сервер лицензий Intel Software License Manager и лицензионный ключ, установленные на специально созданной для этой цели сервисной ВМ.

Отдельным важным вопросом подготовки виртуальных рабочих станций является настройка автоматических обновлений ОС. Автоматические обновления повышают безопасность ОС, но могут приводить к частым перезагрузкам ОС, в том числе при проведении занятий. В ОС Windows 10 не предусмотрено полное отключение обновлений, но существует возможность настройки интервалов между ними.

Другим не менее важным пунктом настройки виртуальной рабочей станции для проведения лабораторного практикума является ограничение доступа обучающихся к сети Интернет. Доступ к сети Интернет может быть ограничен конкретным приложениям с использованием стандартных компонентов ОС Windows 10. Закрыть доступ в Интернет рекомендуется, по крайней мере, следующим приложениям: Internet Explorer, Microsoft Edge, Microsoft Store, Microsoft OneNote, Skype, Windows Messenger, Windows Store.

Следует отметить, что у студентов остаётся возможность выхода в сеть Интернет с помощью протокола RDP, который позволяет копировать файлы, в том числе исполняемые, с локального диска на удалённый компьютер. Студент может скопировать на удалённый компьютер портативный браузер, для которого доступ в Интернет не будет ограничен настроенными правилами. Для того чтобы предотвратить подобное копирование, необходимо запретить перенаправление дисков для службы удалённых рабочих столов рабочей станции. При этом запрет будет действовать на всех пользователей, включая администраторов.

Организация доступа к виртуальным машинам на учебных занятиях

Доступ к виртуальным рабочим столам организуется по протоколу RDP с использованием любой совместимой программы-клиента. При первом подключении к ВМ студент должен зайти в магазин приложений на своем устройстве и установить приложение Remote Desktop от Microsoft («Удалённый рабочий стол»). В ОС Windows для рабочих станций по умолчанию уже установлен стандартный RDP-клиент для подключения к удалённому рабочему столу. В ОС семейства Linux может быть использован RDP-клинет Remmina.

Доступ к удалённым рабочим столам организован через специальную шлюзовую машину, доступную через Интернет. При подключении в адресе должен быть указан специальный номер порта, по которому шлюзовая машина перенаправляет соединение на соответствующую номеру порта ВМ.

Стандартными средствами ОС Windows на ВМ учётным записям студентов разрешён вход только на время занятий. Через минуту после окончания занятия для пользо-

вателя-студента принудительно выполняется выход из системы без возможности повторного подключения до следующего занятия. У учётных записей преподавателей доступ не ограничивается по времени. В часы проведения занятий студент может также подключаться к машинам фактически из любого места, где есть доступ к сети Интернет. Другими словами, инфраструктура пригодна и для удалённого обучения. Кроме того, построенная инфраструктура позволяет, в случае необходимости (болезнь, семейные обстоятельства и др.), по заявкам студентов в течение семестра предоставлять дополнительные интервалы доступа.

Методика построения инфраструктуры для организации виртуальной учебной лаборатории

Опыт реализации рассматриваемого проекта позволил определить следующую последовательность этапов построения виртуальной инфраструктуры для организации и проведения процесса обучения.

1. Выделение требуемого количества однородных вычислительных узлов, которое может быть вычислено по формуле (1):

$$max\left\{\left\lceil\frac{M_{vm}\cdot C_{vm}}{M_n-k}\right\rceil;\left\lceil\frac{C_{vm}\cdot N_{vm}}{N_n}\right\rceil\right\}+n \qquad (1)$$

$\mathbf{C_{vm}}$ – общее количество ВМ, необходимое для проведения занятий;

$\mathbf{M_{vm}}$ – требуемый объём оперативной памяти для 1 ВМ (определяется на основании требований ОС и устанавливаемого программного обеспечения);

$\mathbf{M_n}$ – объём оперативной памяти вычислительного узла;

– объём оперативной памяти на каждом узле, резервируемой под нужды гипервизора и загрузочной ОС;

– количество резервируемых вычислительных узлов;

$\mathbf{N_n}$ – количество процессорных ядер в вычислительном узле;

$\mathbf{N_{vm}}$ – требуемое количество ядер в ВМ.

2. Выделение требуемого объёма дискового пространства в системе хранения данных для размещения файлов ВМ и служебной информации. Объем выделяемого дискового пространства может быть вычислен по формуле (2):

$$S = V_{HDD}\cdot C_{vm} + \sum S_{HDD} \qquad (2)$$

V_{HDD} – объём дискового пространства для ВМ рабочей станции (как правило, определяется на сайте разработчика гостевой ОС);

$\mathbf{C_{vm}}$ – общее количество ВМ рабочих станций, необходимое для проведения занятий;

S_{HDD} – объём дискового пространства для сервисной ВМ.

3. Объединение вычислительных узлов коммуникационной средой, состоящей из двух сегментов сети:

– сеть управления и доступа (маршрутизируемая сеть для организации доступа к ВМ и вычислительным узлам, в том числе через VPN-соединения);

– сеть хранения и передачи данных (изолированная сеть для доступа вычислительных узлов к системам хранения данных).

4. Установка и настройка гипервизоров на выделенных вычислительных узлах.

– установка Proxmox VE на все узлы;

– объединение вычислительных узлов в кластер виртуализации средствами ОС Proxmox VE с использованием сети управления и доступа;

– подключение к кластеру виртуализации через сеть хранения и передачи данных сетевых хранилищ для размещения ВМ и дистрибутивов ПО.

5. Подготовка и настройка сервисных ВМ в формате qcow2 с использованием средств ОС Proxmox VE.

5.1. Создание и настройка сервисных ВМ:

– для организации сетевого доступа (DNS, DHCP, VPN) с двумя сетевыми интерфейсами к сети управления и доступа и к сети Интернет;

– для управления пользователями (LDAP, ActiveDirectory) с сетевым интерфейсом к сети управления и доступа;

– для файлового сервера обмена данными между пользователями ВМ с сетевым интерфейсом к сети управления и доступа;

– для функционирования специального ПО (серверы лицензий) с сетевым интерфейсом к сети управления и доступа.

5.2. Настройка правил миграции сервисных ВМ при выходе из строя вычислительного узла через web-интерфейс Proxmox VE.

6. Подготовка образа ВМ (в формате qcow2) с рабочими столами для проведения

лабораторного практикума с использованием средств ОС Proxmox VE.

6.1. Создание и настройка эталонного образа ВМ с сетевым интерфейсом к сети управления и доступа.

6.2. Создание через интерфейс Proxmox VE из подготовленного эталонного образа требуемого для проведения лабораторного практикума числа копий ВМ.

6.3. Назначение на DHCP-сервере развёрнутым ВМ IP-адресов в соответствии с MAC-адресами их сетевых интерфейсов.

7. Тестирование созданной виртуальной инфраструктуры методом белого ящика.

Заключение

В представленной работе авторами решена фундаментальная задача организации виртуальной учебной лаборатории параллельного программирования. Виртуальная инфраструктура построена на базе свободно распространяемой платформы виртуализации Proxmox VE и вычислительных ресурсах суперкомпьютера МВС-10П МП Petasream МСЦ НИИСИ РАН. Определены оптимальный состав и компоненты подобной инфраструктуры, разработана эффективная методика её построения и отказоустойчивого функционирования.

Важно подчеркнуть, что настройка виртуальной инфраструктуры лаборатории проводится один раз. Далее она функционирует автономно практически без вмешательства специалиста по виртуализации.

Реализованная с помощью предложенной методики виртуальная учебная лаборатория успешно апробирована в весеннем семестре 2018 года для проведения учебных занятий (лабораторного практикума) по параллельному программированию на супер-ЭВМ и графических картах NVIDA для студентов «Межфакультетской кафедры высокопроизводительных вычислений» МГУ имени М.В. Ломоносова, заведующий кафедрой – научный руководитель НИИСИ РАН, академик В.Б. Бетелин.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований по теме (проекту) «Разработка архитектур, системных решений и методов для создания вычислительных комплексов и распределенных сред мультипетафлопсного диапазона производительности, в том числе нетрадиционных архитектур микропроцессоров» (0065-2018-0409).

The virtual educational laboratory based on the Proxmox VE virtualization platform

F.A.Anikeev, A.V.Baranov, F.S.Zaitsev, E.A.Kiselev, S.A.Leschev

Abstract: The article discusses methods and approaches to build a virtual educational laboratory based on the Proxmox VE virtualization platform. The structure and components of the virtual infrastructure that is built on the base of the Proxmox system are considered. Virtual infrastructure designed by the authors was deployed on the MVS-10P supercomputer at Joint supercomputer center of RAS. The infrastructure was used in Moscow state university for teaching parallel programming.

Keywords: proxmox, education, parallel programming, HPC, supercomputer

Литература

1. А.Л.Шестаков, О.Н.Иванова, Л.Б.Соколинский, П.С.Костенецкий. Использование «облачной» образовательной платформы «Персональный виртуальный компьютер» в обучении программированию студентов ИТ-направлений. Преподавание информационных технологий в Российской Федерации: материалы Одиннадцатой открытой Всероссийской конференции. 2013. С. 56-58.

2. Б.М.Шабанов, А.П.Овсянников, А.В.Баранов, О.С.Аладышев, Е.А.Киселёв, Я.О.Жуков. Методы управления параллельными заданиями суперкомпьютера, требующими развёртывания отдельных программных платформ и виртуализации сетей. Суперкомпьютерные дни в России: Труды международной конференции. 2017. С. 616-627.

3. Б.М.Шабанов, А.П.Овсянников, А.В.Баранов, Е.А.Киселёв, С.А.Лещев, Б.В.Долгов, Д.Г.Гуменный, Д.Л.Шуров. Решение задач фотореалистичной компьютерной графики на базе защищённой инфраструктуры суперкомпьютерного центра коллективного пользования. Суперкомпьютерные дни в России: Труды международной конференции. 2017. С. 733-741.

4. О.С.Аладышев, А.В.Баранов, Р.П.Ионин, Е.А.Киселёв, В.А.Орлов. Сравнительный анализ вариантов развертывания программных платформ для высокопроизводительных вычислений // Вестник Уфимского государственного авиационного технического университета. 2014. Т. 18. № 3 (64). С. 295-300.

5. A.Kovari, P.Dukan. KVM & OpenVZ virtualization based IaaS open source cloud virtualization platforms: OpenNode, Proxmox VE. 2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, Subotica, 2012, pp. 335-339. DOI: 10.1109/SISY.2012.6339540

6. L.Chen, W.Huang, F.Sui, D.Chen, and C.Sun. The online education platform using Proxmox and noVNC technology based on Laravel framework. 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), Wuhan, 2017, pp. 487-491. DOI: 10.1109/ICIS.2017.7960041

7. Manage virtual machines with virt-manager. URL:https://virt-manager.org (дата обращения: 02.11.2018).

8. Proxmox Virtual Environment. URL:https://pve.proxmox.com/wiki/Main_Page (дата обращения: 02.11.2018).

9. AQEMU a GUI for virtual machines using QEMU as the backend. URL:https://sourceforge.net/projects/aqemu (дата обращения: 02.11.2018).

10. GKVM. A Gnome user interface for KVM. URL:http://gkvm.sourceforge.net (дата обращения: 02.11.2018).

11. Cloudmin. URL:http://www.webmin.com/cloudmin.html (дата обращения: 02.11.2018).

12. Ovirt. Powerful open source virtualization. URL:https://ovirt.org (дата обращения: 02.11.2018).

13. МВС-10П МП PETASTREAM. URL:http://www.jscc.ru/resources/hpc/#item1587 (дата обращения: 02.11.2018).

Разработка метода передачи мультимодальных данных в условиях ограниченных каналов связи

Ю. Н. Морин

Межведомственный суперкомпьютерный центр РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail: ymorin@jscc.ru

Аннотация: В статье рассматриваются методы организации передачи мультимодальных данных в условиях ограниченных каналов связи. Подробно рассматривается метод передачи мультимодальных данных с использование кеширующих ретрансляторов.

Ключевые слова: поток, мультимодальные данные, сеть доставки.

Введение

В настоящее время в Интернет прослеживается тенденция вытеснения текстовых коммуникаций другими видами: изобразительными, аудио, видео. Объем видеоконтента постоянно возрастает в средствах массовой информации, в блогах, образовании, играх, рекламе. Даже средства моментального обмена сообщениями, ориентируются не передачу или включение в текст идеограмм, эмодзи (от яп. 絵 — картинка и 文字 — знак, символ), изображений, аудио и видео.

Согласно [1] средний объем интернет видео передаваемого в мире по сетям в 2018 г. составляет более 98 ПБ в месяц, при этом среднегодовой темп роста объема интернет видео в 2016-2021 гг. составляет 31%.

Увеличение объемов передаваемых мультимедийных данных увеличивает нагрузку на сети передачи данных. В таких условиях чрезвычайно важным является создание методов и алгоритмов позволяющих уменьшить нагрузку на сети передачи данных без ухудшения качества работы мультимедийных сервисов.

Обзор моделей передачи мультимодальных данных

Под мультимодальными данными мы понимаем данные, включающие несколько модусов (текстовых, аудиальных, пространственных, визуальных) образующих сообщение.

Классический метод передачи мультимодальных данных, является модель клиент–сервер [2, 3] (см. рисунок 1).

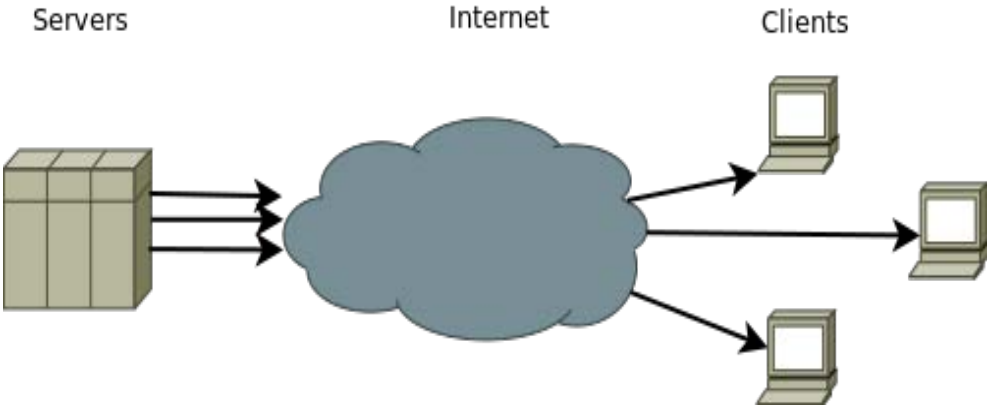


Рисунок 1. Модель клиент-сервер

В случае использования одиночного сервера данная модель обладает следующими недостатками: если производительности не хватает на обслуживание всех клиентов или сервер находится на ремонте, то это вызовет неработоспособность всей системы. При ис-

пользовании многоуровневой архитектуры клиент-сервер – кластера позволит избежать обозначенных проблем выше. Следует особо отметить, что такая модель требовательна к каналам связи кластера серверов.

Использование технологии peer-to-peer (P2P) [4] для передачи мультимодальных данных (см. рис. 2), когда клиент получивший фрагмент данных может выполнять функции по распространению фрагмента выступая в качестве сервера для других клиентов, позволяет существенно снизить требование к каналам связи и производительности сервера мультимодальных данных[5]. Не трудно показать, что недостаток данной схемы в том повышается требования к сети пользователя.

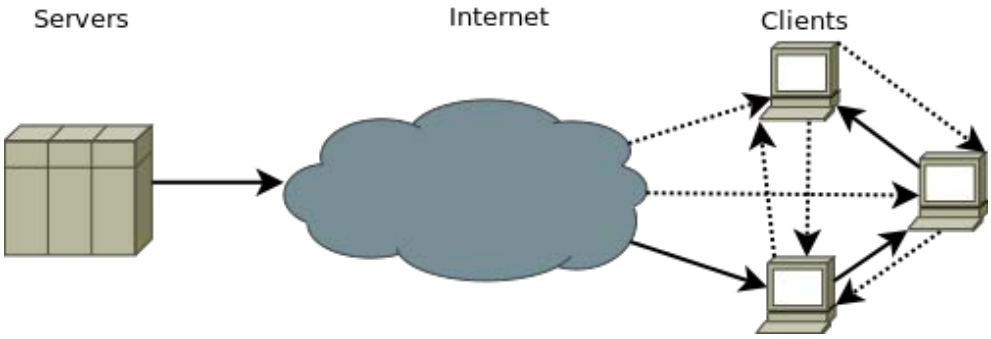


Рисунок 2. Модель peer-to-peer

Задача состоит в том, чтобы предложить метод, который позволит снизить требования к пропускной способности каналов связи: сервера/кластера, клиента, провайдера для осуществления передачи мультимодальных данных.

Рассмотрим расчет нагрузки на каналы связи в каждой из моделей передачи мультимодальных данных.

Клиент-серверная модель

В случаи использования клиент — серверной модели максимальная загрузка канала и длина трассы будут следующими.

Максимальная загрузка канала будет в точке подключения сервера и равна:

$$S_{cs} = \sum_{i=1}^{N} s_i = N * s, (1)$$

где N — количество клиентов; s_i — скорость одного подключенного клиента, в нашем рассмотрении мы допускаем, что все s_i равны.

Суммарная длина трассы, которую преодолевает трафик всех клиентов, равна:

$$L_{cs} = \sum_{i=1}^{N} l_{ai}, (2)$$

где l_{ai} — трасса от сервера до i-го клиента.

Модель Peer-to-Peer

Схема распространения трафика в модели peer-to-peer приведена на рисунке 3.

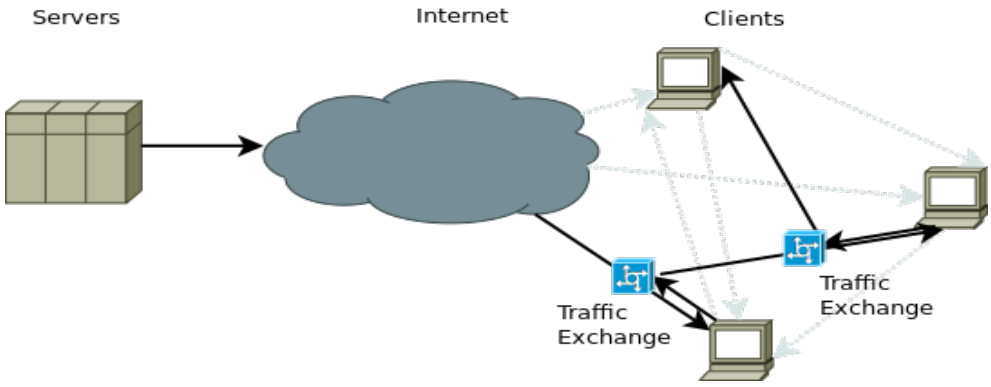


Рисунок 3. Распространение трафика в модели peer-to-peer

Полагаем, что не вызывает сомнения, что минимальное значение максимальной загрузки канала будет достигаться если фрагмент данных распространяется

следующим образом: $S \rightarrow C_1 \rightarrow C_2 \rightarrow \ldots \rightarrow C_N$, где S — сервер, C_i – клиент номер i, N – количество клиентов.

Максимальная загрузка канала будет в точках подключения клиентов i, $i \in [1, N-1]$ и равна:

$$S_{p2p} = 2 * s_i = 2 * s, (3)$$

Суммарная длина трассы, которую преодолевает трафик всех клиентов, равна:

$$L_{p2p} = \sum_{i=1}^{N-M} l_{ai} + \sum_{j=1}^{M} (l_{ix} + l_{xj}) \quad (4)$$
$$l_{ai} = l_{ax} + l_{xi}$$
$$l_{xi} = l_{ix}$$

где l_{ix} и l_{xj} – трасса от i и j клиента до общей для них точки обмена трафиком; M — количество клиентов выступающих в роли пиров.

Метод снижения требований к каналам связи на основе использования кеширующих ретрансляторов

Хотелось бы иметь метод передачи мультимодальных данных обладающий преимуществами каждой из моделей клиент–серверной и peer-to-peet.

В нашем иссле-довании установлено, что добиться этого возможно разместив в точках обмена трафика, рассмотренных в разделе «Модель peer-to-peer», кеширующих ретрансляторов (см. рис. 4).

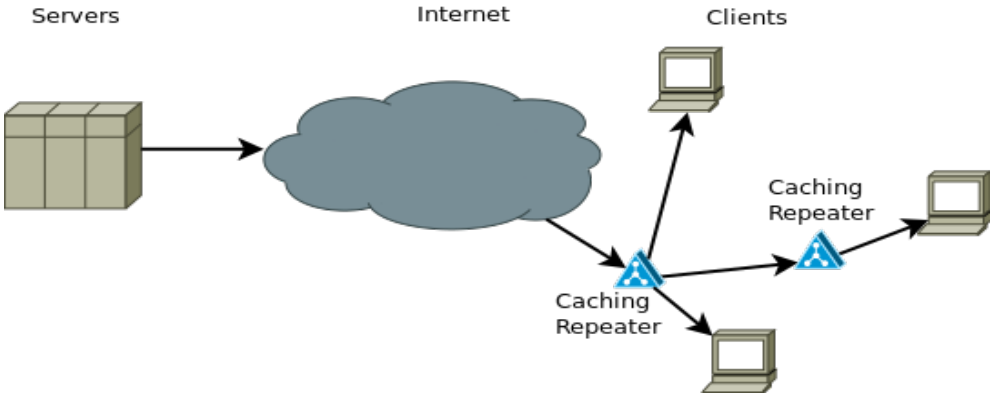


Рисунок 4. Модель с кеширующими ретрансляторами

В этом случаи суммарная длина трассы, которую преодолевает трафик всех клиентов, равна:

$$\begin{cases} L_{cr} = \sum_{j=1}^{M} l_{aj} + \sum_{j=1}^{M} \sum_{i=1}^{N_j} l_{ij} \quad (5) \\ l_{aj} + l_{ji} = l_{ai} \\ l_{ij} < l_{ai} \end{cases}$$

где l_{aj} — трасса до точки обмена; l_{ij} – трасса от j-ой точки обмена до i-клиента; l_{ai} – длина трасса от сервера до i-го клиента, как и в случаи рассмотренном в главе «клиент-серверная модель»; N_j – количество клиентов использующих точку ретрансляции j; M — количество точек ретрансляции.

Решая уравнения 2, 4 и 5 нетрудно показать, что

$$L_{cr} < L_{cs}$$
$$L_{cr} < L_{p2p} \quad (6)$$

и

$$L_{cr} \sim f(N_j, l_{ij}) \sim f(N_j, N_j) = f(N_j^2) \sim \frac{1}{N_j^2} \quad (7)$$

Отсюда следует увеличение количества кеширующих ретрансляторов квадратично уменьшает суммарную длину трассы, которую преодолевает трафик всех клиентов, а значит уменьшает нагрузку на сети.

Заключение

В нашем исследовании установлено, что кеширующие ретрансляторы мультимодального трафика уменьшают нагрузку на сети передачи данных без ухудшения качества работы мультимедийных сервисов, так как данные не подвергаются перекодированию и каким-либо другим изменениям. Установка кеширующих ретрансляторов в точках обмена трафика уменьшает дублирование трафика на магистральных каналах.

В настоящее время также решена задача выбора ближайшего ретранслятора для перенаправления трафика. В следующей статье будет рассмотрено, как направить трафик от клиента на ближайший кеширующий ретранслятор; вопросы обеспечения безопасности, возможности использования метода для передачи конфиденциального (зашифрованного) трафика; вопросы уменьшения задержек при передаче чувствительного к ним мультимодального трафика.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований по теме (проекту) «Исследование и разработка методов сетевой интеграции ресурсов и сервисов научных организаций» (0065-2018-0404).

Development of a method for transmitting multimodal data in conditions of limited communication channels

Y.N.Morin

Abstract: The article discusses methods for organizing transmission of multimodal data in conditions of limited communication channels. The method of transmitting multimodal data using caching repeaters is considered in detail.

Keywords: stream, multimodal data, Content Delivery Network, TCP, UDP.

Литература

1. Cisco Visual Networking Index: Forecast and Methodology, 2016–2021 // Cisco Systems Inc., 2017 [электронный ресурс] URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html (дата обращения 08.11.2018).
2. Client–Server Multimedia Streaming. (n.d.). Encyclopedia of Multimedia, pp. 61–63
3. D.P. Wu, Y.W. Hou, W. Zhu, Y.Q. Zhang, and J.M. Peha, “Streaming Video over the Internet: Approaches and Directions,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 11, No. 1, February 2001, pp. 282–300.
4. Liben-Nowell D., Balakrishnan H., Karger D., Analysis of the evolution of the peer-to-peer systems. // PODC '02 Proceedings of the twenty-first annual symposium on Principles of distributed computing , ACM, NY, 2002, pp.233-242
5. Magharei, N., Rejaie, R., Rimac, I., Hilt, V., Hofmann, M. ISP-Friendly Live P2P Streaming. IEEE /ACM Transactions on Networking, 22(1), 2014, pp.244–256. doi:10.1109/tnet.2013.2257840

Загрузка многопроцессорного вычислительного комплекса на базе коммуникационной среды RapidIO

Ю.М. Лазутин

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: lazutin@niisi.ras.ru

Аннотация: В работе описывается технологическое программное обеспечение, используемое в отечественных программно-аппаратных комплексах на базе коммуникационной среды RapidIO для загрузки исполняемых образов ОС РВ Багет в процессорные элементы комплекса и для контроля целостности ПО на этапе эксплуатации. Загрузчик использует подготовленную заранее схему распределения исполняемых образов по процессорным элементам.

Ключевые слова: ОС РВ Багет, RapidIO, загрузка многопроцессорного вычислительного комплекса, контроль целостности программного обеспечения

Введение

Разработанные в ФГУ ФНЦ НИИСИ РАН отечественные микропроцессоры 1890ВМ6 и 1890ВМ7 [1,2] имеют встроенные контроллеры интерфейса RapidIO [3,4]. Универсальный микропроцессор 1890ВМ6 – 1 канал параллельного RapidIO и 2 канала последовательного RapidIO. 1890ВМ7, имеющий специальный сопроцессор для цифровой обработки сигналов – один канал параллельного RapidIO. Коммутатор 1890ВГ3Я с 8-ю портами параллельного RapidIO позволяет создавать из этих микропроцессоров процессорные модули (до двух процессоров на плате каждый со своей памятью) и отдельные приборы / крэйты, а каналы последовательного RapidIO 1890ВМ6 – объединять приборы в общую RapidIO систему. На практике для связи между приборами используется также микросхема 1890ВГ18, имеющая 1 канал параллельного RapidIO, один или два (в зависимости от разрядности) канала последовательного RapidIO и коммутатор RapiIO пакетов. Эта же микросхема содержит интерфейс оптоволоконных линий и используется для ввода внешних данных.

Разрядность идентификатора оконечного устройства сети RapidIO у 1890ВМ7 составляет 8 бит (остальные микросхемы поддерживают 16 бит), поэтому в общем случае максимальное число процессорных элементов в системе, включающей в себя и 1890ВМ6, и 1890ВМ7 – 255. Разрабатываемые в настоящее время приложения из области цифровой обработки сигналов уже выходят на это ограничение. В качестве операционной системы используется ОС РВ Багет [5].

Разработка программного обеспечения (ПО) для системы из 255 процессорных элементов (ПЭ) является непростой задачей, особенно для приложений, работающих в режиме реального времени. Помимо разработки вычислительных алгоритмов и их распараллеливания по имеющимся вычислительным ресурсам [7] имеются и технологические проблемы с организацией процесса разработки ПО и его эксплуатации. Среди них:

- определение топологии сети RapidIO, ее инициализация, включая программирование коммутаторов;
- согласованная загрузка ПЭ в процессе разработки ПО и при его эксплуатации;
- подготовка приборов к штатной эксплуатации - запись во внутренние запоминающие устройства приборов (включая флэш-память каждого ПЭ) необходимых для работы в штатном режиме файлов;
- минимизация типов модулей ЗИП (Запасные части, Инструменты и Приспособления — ГОСТ 2.601-2013) путем обеспечения их взаимозаменяемости.

В работе [6] описывается реализованный в рамках проекта ОС РВ Багет программный инструментарий для конфигурирования программно-аппаратного комплекса на базе перечисленных выше аппаратных средств, при котором на специальном декларативном языке описывается конфигурация системы:

- аппаратная: связи между компонентами, типы процессорных модулей, идентификаторы ПЭ и маршруты коммутаторов;
- и программная: для каждого ПЭ указывается какой файл с загрузочным образом ОС РВ из какой файловой системы нужно в него загрузить, какие сетевые интерфейсы и с какими IP- адресами следует в ОС РВ инициализировтать.

Это Высокоуровневое Описание Системы (ВОС) транслируется в elf-файл (формат, используемый в ОС типа Linux для представления откомпилированных программ), называемый Целевое Описание Системы (ЦОС). ОС РВ предоставляет специальную библиотеку для работы с файлом ЦОС. ЦОС может быть включен в загрузочный образ ОС РВ и использоваться для инициализации сетевых интерфейсов и сети RapidIO. Предусмотрено также использование ЦОС отдельным технологическим ПО. В данной работе описывается такое технологическое ПО — Rio-загрузчик.

Rio-загрузчик используется на всех этапах жизни прикладного ПО:
- на этапе разработки ПО Rio-загрузчик обеспечивает загрузку ПЭ прибора с файлового сервера (технологическая ЭВМ под управлением ОС Линукс) в соответствии с конфигурацией ПО, описанной в файле ЦО;
- Rio-загрузчик позволяет также записать все необходимое ПО во внутренние хранилища прибора (флэш-память каждого ПЭ и флэш-диски с интерфейсом IDE) по конфигурации, описанной в ЦОС. Эта операция выполняется при серийном производстве разработанных программно-аппаратных комплексов, но отлаживается на финишном этапе разработки, когда появляется работоспособное ПО;
- загрузку приборов с внутренних носителей во время эксплуатации комплексов также обеспечивает Rio-загрузчик;
- наконец, во время эксплуатации комплексов Rio-загрузчик позволяет проконтролировать целостность ПО и корректность аппаратной конфигурации по сохраненным при записи контрольным суммам и различным сигнатурам.

Rio-загрузчик размещается во флэш-памяти каждого ПЭ. Запись во флэш-память выполняется с помощью программы, находящейся в постоянном запоминающем устройстве (ПЗУ) ПЭ — программы ПЗУ. Запуск Rio-загрузчиков в приборе также осуществляется с помощью программы ПЗУ. Программа ПЗУ содержит интерпретатор языка psl, созданный на базе проекта hush [8]. Сценарии, написанные на этом языке могут храниться во флэш-памяти ПЭ, что позволяет эффективно управлять процессом тестирования прибора и запуска системного ПО.

1. Инициализация RapidIO

Для того, чтобы сеть RapidIO могла функционировать, необходимо выполнить процедуру инициализации (enumeration), в процессе которой источникам и получателям пакетов данных (оконечным элементам/end points сети) присваиваются идентификаторы RapidIO (далее RioId), а в коммутаторы записываются маршруты следования пакетов. В разрабатываемых комплексах приборы являются относительно автономными подсистемами (их можно независимо включать и они автономно тестируются) и должны инициализироваться автономно.

Согласно спецификации RapidIO инициализация должна выполняться одним ПЭ, далее называемым главным. Для надежности (на случай выхода из строя главного ПЭ) рекомендуется наличие дублера, который, обнаружив после некоторого таймаута, что инициализация не выполнена, также начинает процесс инициализации. В рассматриваемых комплексах дублер отсутствует, поскольку потеря главного фатальна и требует ремонта прибора путем использования модулей ЗИП.

В ЦОС для каждого прибора указывается главный ПЭ (host). На рис. 1 приведен упрощенный фрагмент ЦОС для одного ПЭ. Фрагмент взят из промежуточного xml-представления ЦОС, получаемого при трансляции исходного высокоуровневого описания в elf-файл. Фрагмент находится в секции ЦОС, описывающей режим загрузки (текст в первой строке `Startup Mode="TDESC_SM_LOAD"`), подрежим `"NFS-osuser".`

Указание главного базируется на понятии «географического адреса». Модули и приборы, созданные на базе перечисленных во Введении микросхем разработаны таким образом, что каждый ПЭ в приборе имеет уникальный «географический адрес» (ГА), в ЦОС — поле GeoIdMain. Часть ГА определяется позицией ПЭ на модуле и позицией модуля в приборе (на рис. 1 - 02), часть — номером прибора (на рис. 0x1c). Номер прибора задается на внешнем разъеме прибора. В файле ЦОС все ПЭ комплекса однозначно

идентифицируются своим ГА. На рис. 1 HostGeoId — это ГА главного ПЭ в приборе 0x1c. Некоторые приборы резервируются (холодный резерв, включаемый при выходе из строя основного прибора). GeoIdMain сответствует основному прибору, GeoIdRes — резервному, если он имеется. Отличия — только в номере прибора.

```
<Startup Mode="TDESC_SM_LOAD" Submode="NFS-osuser">
  <Cpu GeoIdMain="0x1c02"
       GeoIdRes="0x0"
       RioId="1"
       HostGeoId="0x1c02"
    ...
    <Load Mode="TDESC_LM_SELF" Path="nfs//t16E00-00.u03" />
    <Ethernet>
      <Interface Name="sm0" Addr="10.21.17.31" Mask="255.255.255.0"  />
      <Interface Name="de9" Addr="192.168.57.10" Mask="255.255.255.0" />
    </Ethernet>
    ...
    <Mounts>
      <Mount Options="-t nfs 192.168.57.55:/home/osuser nfs" />
    </Mounts>
  </Cpu>
```

Рис. 1. Фрагмент целевого описания системы для режима загрузки

В рассматриваемых программно-аппаратных комплексах файл ЦОС хранится на флэш-дисках приборов в файловой системе vfat вместе с ПО комплекса. Конструктивно флэш-диски оформлены как PMC-мезонины, устанавливаемые на процессорные модули. В приборе может быть более одного модуля с аппаратной конфигурацией, совпадающей с модулем главного ПЭ и запасные модули из комплекта ЗИП, предназначенные для их замены должны допускать замену любого такого модуля без какого-либо вмешательства оператора в настройки программы ПЗУ и загрузчика. Более того, предъявляется требование и межприборной взаимозаменяемости таких модулей. Для обеспечения полной взаимозаменяемости модулей с конфигурацией главного на все флэш-диски комплекса записывается одно и то же содержимое и в конфигурационные параметры загрузчика помещается путь к файлу ЦОС.

Загрузчик при старте должен определить, является ли ПЭ, на котором он исполняется главным.

Алгоритм определения главного ПЭ упрощенно (только для штатной загрузки с флэш-диска) выглядит следующим образом:

- загрузчик, получив управление от программы ПЗУ определяет, доступен ли ПЭ, на котором он выполняется, флэш-диск и монтируется ли на него файловая система vfat; при наличии vfat загрузчик читает файл ЦОС, определяет свой ГА и, перебирая описания ПЭ, ищет в ЦОС описание ПЭ у которого GeoIdMain (или GeoIdRes) совпадает с его ГА;
- если ГА ПЭ совпадает в найденом описателе с HostGeoId, то загрузчик считает себя главным и с помощью специальной функции из библиотеки ОС РВ для работы с ЦОС выполняет процедуру инициализации прибора;
- если в предыдущих пунктах какое-либо из условий не выполнено, загрузчик переходит в режим подчиненного — дожидается завершения процесса инициализации RIO-сети (фиксированный таймаут);

После завершения инициализации RIO-сети все загрузчики активируют драйвер, эмулирующий на RIO-сети Ethernet-устройство с именем 'sm' и инициализируют сетевую подсистему ОС РВ, подключая к ней эмулятор с фиксированными IP и MAC адресами, в которых последний байт равен RioId ПЭ. IP-адрес, указанный в ЦОС для устройства 'sm' (рис. 1) не используется загрузчиком.

После инициализации главный загрузчик, руководствуясь ЦОС, опрашивает по локальной сети подчиненные загрузчики и принимает решение о возможности дальнейшей работы.

Отсутствие ответа от какого-либо ПЭ допустимо только для режима записи прикладного ПО в отдельные модули. Этот режим может использоваться на специальном стенде, на котором «прошиваются» отдельные модули. Такая конфигурация неполного прибора задается в конфигурационных параметрах загрузчика в главном ПЭ прибора.

2. Конфигурирование загрузчика

После завершения процесса инициализации RIO-сети подчиненные загрузчики ожидают команд от главного загрузчика по эмулируемой Ethernet-сети. Главный загрузчик читает свои конфигурационные параметры из выделенного сектора флэш-памяти ПЭ, делает настраиваемую паузу, чтобы дать оператору возможность перейти к процедуре изменения конфигурационных параметров путем нажатия клавиши 'Enter' и переходит в режим, указанный в параметрах.

Загрузчик имеет два режима работы:

- режим загрузки в ПЭ прибора указанных в ЦОС загрузочных образов ОС РВ. На рис. 1 указан подрежим `NFS-osuser`, что соответствует загрузке из инструментальной ЭВМ по протоколу NFS. Ключевое слово в начале текста указывает, откуда брать файлы. Штатной загрузке соответствует ключевое слово vfat-301. Текст после ключевого слова произволен. Подрежимов может быть произвольное число;

- режим записи указанных в ЦОС файлов во флэш-память ПЭ и во флэш-диски прибора. Соответствующая секция ЦОС устроена аналогично секции загрузки — см. рис. 3.

На рисунке 2 приведен пример конфигурационных параметров загрузчика (с некоторыми сокращениями). Конфигурационные параметры разбиты на две группы — основные и дополнительные.

```
Boot device            : de9
inet on ethernet       : 10.4.6.25:255.255.255.0
gateway                :
host inet              : 10.4.6.55
MountPath              : /home/PROJECTS/PRJ_1
COS                    : FLASH/COMMON_MNT/RIO/COS/bin/COS.o
Userfunc.tar           : FLASH/COMMON_MNT/RIO/CCS/uf.tar
rconfflag              : v
lmode                  : cos,NFS-SERVER-PRJ
rmode                  : NFS-301-FLASH:03
crsi                   :
rioverbose             : 0
```

Рис. 2. Конфигурационные параметры загрузчика

Boot device - указывает, откуда загрузчик читает все файлы. Может иметь три значения: de, de9 и ide. Первые два — различные виды контроллера Ethernet 10-100М бит/сек, т. е. файлы читаются с технологической ЭВМ — сервера загрузки. Устройство de9 встроено в процессор 1890ВМ6, de реализуется отдельной микросхемой 1890ВГ3, устанавливаемой либо на процессорный модуль (модули с микропроцессором 189ВМ7), либо на PMC-мезонин, устанавливаемый на модуль с главным ПЭ. Мезонин иногда используется потому, что имеет два интерфейса MII и позволяет организовать резервирование линий. Драйвер автоматически переключается на другую линию при получении сигнала (прерывание) о разрыве используемой линии.

Устройство загрузки ide обозначает флэш-диск, установленный на процессорный модуль с главным ПЭ (интерфейс - IDE). Поддерживается только файловая система vfat.

inet on ethernet, gateway и **host inet** — IP-адреса главного ПЭ, шлюза (если он нужен) и технологической ЭВМ. При необходимости указывается маска подсети. Эти параметры используются для чтения файла ЦОС и должны сответствовать тем, что указаны для главного ПЭ в файле ЦОС.

Mount Path — точка монтирования NFS в файловой системе технологической ЭВМ. ОС РВ Багет поддерживает сетевые протоколы FTP и NFS. При чтении по сети на практике используется только протокол NFS. Причина в том, что при записи ПО в приборы загрузчик сохраняет на технологической ЭВМ контрольную информацию. В ОС РВ Багет только протокол NFS обеспечивает запись данных на удаленный NFS-сервер.

COS — путь к файлу ЦОС (относительно точки монтирования).

userfunc.tar — путь к tar-файлу, содержащему функцию userfunc, используемую в режиме записи ПО в прибор для подсчета и контроля контрольных сумм, сигнатур, номеров процессорных модулей и т. д. Данная функция пишется разработчиками прикладного ПО и динамически подгружается и подключается

загрузчиком с помощью штатных средств ОС РВ.

rconfflag — параметр, включающий режим записи ПО в прибор или проверку его целостности. При пустом заначении — режим загрузки. Значения rconfflag управляют действиями загрузчика и поведением userfunc.

lmode — содержит два параметра. Текст cos указывает, что нужно использовать файл ЦОС. Когда формировалась исходная спецификация на функции загрузчика предполагалось, что будет и режим работы без ЦОС. Однако такой режим пока не понадобился. Второй параметр (после запятой) — подрежим режима загрузки.

rmode — подрежим режима записи ПО в прибор. Через символ ':' можно указать, для каких модулей выполняется операция записи. rmode на рис. 2 указывает, что запись выполняется только в ПЭ модуля, установленного в 3-й слот прибора.

crsi и **rioverbose** относятся к дополнительным параметрам, используемым при отладке всего технологического процесса, связанного с применением файла ЦОС. Crsi позволяет подменить реальный номер некоторого прибора на номер, указанный в crsi. Такая подмена поддерживается конфигуратором ОС РВ и пакетов поддержки модулей, обеспечивающих функционирование ОС РВ на конкретных модулях. Возможность подмены номера прибора очень удобна при удаленной работе со стендами. Rioverbose содержит три поля, задающих уровни отладочной трассировки для 3-х компонент загрузчика — собственно загрузчик, программа инициализации сети RIO и функция userfunc.

3. Загрузка процессорных элементов

На этапе разработки прикладного ПО, когда все ПЭ загружаются из технологических ЭВМ (для обеспечения согласованности загружаемого ПО обычно это один сервер загрузки, куда собираются все загружаемые файлы), может использоваться произвольное число подрежимов режима загрузки (на практике — до 10-ти). Подрежимы используются для фиксации промежуточных отлаженных версий прикладного ПО, для различных проектов, отлаживаемых на одном и том же стенде. Отличия — в путях к загружаемым файлам, IP-адресах. Штатный подрежим, соответствующий загрузке из файловой системы vfat с флэш-диска, всегда один — VFAT-301.

В режиме загрузки (параметр rconfflag пуст) загрузчик находит в ЦОС секцию указанного в параметре lmode подрежима, ищет в последовательности описателей ПЭ описатель, в котором либо поле GeoIdMain, либо GeoIdRes совпадает с географическим адресом ПЭ, на котором загрузчик исполняется и в цикле перебирает последующие описатели, пока номер прибора в GeoIdMain или GeoIdRes (смотря за что зацепились при поиске) не изменится. Для каждого описателя загрузчик читает указанный в атрибуте Path файл (загрузочный образ ОС РВ в elf-формате объемом обычно около 10 мегабайт), пересылает его ПЭ с указанным в текущем описателе RioId и дает ему команду на обработку переданного файла. Пересылка файла выполняется по специальному «быстрому» протоколу, команды и ответы на команды пересылаются по TCP-соединению через эмулируемую на RapidIO локальную сеть прибора. Использование TCP-протокола для взаимодействия загрузчиков и протокол этого взаимодействия — это наследство от VME-загрузчика, который широко используется для разработанных ФГУ ФНЦ НИИСИ РАН VME-модулей. В VME-приборах TCP на VME использовался и для загрузки файлов: главный загрузчик по очереди выдает подчиненным загрузчикам команду «отработай свои конфигурационные параметры» и подчиненный загрузчик монтирует указанную в конфигурационных параметрах файловую систему, читает по сети указанный файл с образом ОС РВ. При этом обычно главный ПЭ выполнял роль шлюза между внешней ethernet-сетью и внутренней VME-сетью. Однако пропускная способность эмулируемого TCP-протокола на порядок меньше специализированного быстрого. В реальных VME-приборах максимальное число ПЭ — 6. В RIO-приборах сейчас уже 20 ПЭ и разница во времени вполне ощутима.

Завершается процесс загрузки прибора рассылкой подчиненным ПЭ копии файла ЦОС — он размещается в фиксированном месте ОЗУ, наследуется ОС РВ (специальный механизм пакета поддержки модуля) и используется при инициализации RIO-сети и сетевой подсистемы ОС РВ.

4. Запись программного обеспечения в приборы

Режим записи ПО в приборы существенно сложнее режима загрузки. На рис. 3 приведен сокращенный элемент описания из секции записи ЦОС для

главного ПЭ прибора 0x1c. Тэги ReplFile содержат информацию о файлах, которые надо записать во флэш-память ПЭ с ГА 0x1c02 (если в тэге есть атрибут TargetAddress) или в файловую систему vfat (если есть атрибут TargetPath). Флэш-память есть у каждого ПЭ и она должна содержать набор сценариев для программы ПЗУ (файлы с расширением psl), необходимых для старта и тестирования прибора, tar-файл с тестовой системой и загрузчик соответствующего типа (для 1890ВМ6 и для 1890ВМ7). Файловая система vfat устанавливается на всех флэш-дисках прибора (PMC-мезонин на модулях с 1890ВМ6). В vfat записывается все ПО аппаратно-программного комплекса, а не только ПО, исполняемое на данном приборе. Это обусловлено требованием взаимозаменяемости модулей во время эксплуатации комплекса. В атрибуте TargetPath могут указываться и имена отдельных файлов, и имена каталогов.

```
<Startup Mode="TDESC_SM_RECONF" Submode="NFS-301-FLASH">
<Cpu GeoIdMain="0x1c02"
       GeoIdRes="0x0"
       RioId="1"
       HostGeoId="0x1c02"
        ...
<Mounts>
        <Mount Options="-t nfs -o uid=1005,gid=1005 192.169.57.55: nfs" />
        <Mount Options="-t msdos /dev/ide0 /vfat" />
</Mounts>
<Reconf>
  <ReplFile
   SourcePath="nfs/FLASH/COMMON_MNT/RIO/PMON/Common/Start/s_prib.psl"
   TargetAddress="0x240000" />
          ...
  <ReplFile
 SourcePath="nfs/FLASH/COMMON_MNT/RIO/DELIVERY/BOOTER/K64/work/btcprio64c.bin"
  TargetAddress="0x020000" />
  <ReplFile
   SourcePath="nfs/FLASH/COMMON_MNT/RIO/ETS/etsCOI.tar"
   TargetAddress="0x400000" />
  <ReplFile
SourcePath="nfs/FLASH/COMMON_MNT/RIO/DELIVERY/BOOTER/K128/work/bt128c.bin"
TargetPath="/vfat/FLASH/COMMON_MNT/RIO/DELIVERY/BOOTER/K128/work/bt128c.bin"/>
  <ReplFile
    SourcePath="nfs/FLASH/COMMON_MNT/RIO/ETS/etsCOI.tar"
    TargetPath="/vfat/FLASH/COMMON_MNT/RIO/ETS/etsCOI.tar" />
  <ReplFile
   SourcePath="nfs/FLASH/COMMON_MNT/RIO/PMON/Common/Start/s_prib.psl"
   TargetPath="/vfat/FLASH/COMMON_MNT/RIO/PMON/Common/Start/s_prib.psl" />
        ...
  <ReplFile
    SourcePath="nfs/FLASH/COMMON_MNT/RIO/CCS/uf.tar"
    TargetPath="/vfat/FLASH/COMMON_MNT/RIO/CCS/uf.tar" />
  <ReplFile
    SourcePath="nfs/FLASH/COMMON_MNT/RIO/COS/bin/COS.o"
    TargetPath="/vfat/FLASH/COMMON_MNT/RIO/COS/bin/COS.o" />
  <ReplFile
    SourcePath="nfs/FLASH/RIO/01"
    TargetPath="/vfat/FLASH/RIO/01" />
  <ReplFile
        ...
</Reconf>
</Cpu>
```

Рис.3. Фрагмент целевого описания системы для режима записи ПО

Режим записи ПО это технологическая операция, ориентированная на нужды производства, устанавливающего на получаемые от производителя аппаратуры приборы и отдельные модули (модули ЗИП или модули после ремонта) требуемое ПО. Для записи отдельных модулей используется специальный стенд, в который устанавливаются прописываемые модули и загрузчику в параметре rmode указывается, какие именно модули надо прописать. Загрузчик при просмотре файла ЦОС пропускает описания для неустановленных модулей. В процессе отладки и

тестирования ПО возникает потребность переписать в отдельных модулях отдельные файлы. В этом случае в параметре rmode можно указать имя файла, содержащего информацию о переписываемых файлах и модулях.

5. Контроль целостности программного обеспечения и аппаратуры

На этапе эксплуатации описываемых в статье программно-аппаратных комплексов регламент технического обслуживания требует периодической проверки целостности записанного в приборы ПО и соответствия аппаратной конфигурации документации на приборы. Целостность ПО обычно проверяется контрольными суммами различного типа:

- есть суммы, зафиксированные в документе;
- средствами встроенного ПО выполняется суммирование в действующем приборе и результат сравнивается с документом.

В случае комплекса, где в одном приборе находится до 20 ПЭ, а во всем комплексе их более 200 необходимо, следуя положениям концепции контролируемого выполнения [9], предоставить средства для автоматизации этого процесса. Потребители разработанных ФГУ ФНЦ НИИСИ РАН аппаратных средств на базе RapidIO пришли к такому же выводу и сформулировали дополнительные требования к выполняемым RIO-загрузчиком функциям. Принимать решения о том, что и как суммировать, где хранить суммы, какие сигнатуры использовать для идентификации модулей и где их размещать должны разработчики программно-аппаратного комплекса. В ходе разработки и в процессе сопровождения разработанного ПО принятые решения могут пересматриваться и их программная реализация должна быть отделена от кода загрузчика. Отделение полезно также и для разделения зон ответственности. Документация на код, реализующий контрольные функции, разрабатывается и хранится у разработчиков комплекса и может модифицироваться независимо при условии сохранения реализованного в загрузчике интерфейса. ОС РВ Багет содержит штатные средства (таблица символов/имен и динамический загрузчик), позволяющие во время выполнения исполняемого образа ОС РВ (например, RIO-загрузчика) загрузить объектный код программы, содержащей вызовы функций, входящих в образ ОС РВ. В свою очередь, программа в исполняемом образе может найти в таблице символов предопределенное имя головной функции загруженной программы, получить из талицы ее адрес и вызывать эту функцию при каких-то действиях.Конфигурационный параметр загрузчика userfunc.tar содержит путь к tar-файлу, содержащему функцию userfunc, выполняющую все действия по контролю целостности программно-аппаратного комплекса. Формат tar-файла обусловлен тем, что среди файловых систем, поддерживаемых ОС РВ, есть файловая система tar. Tar-файл читается в ОЗУ, на соответствующий адрес монтируется файловая система tar и из нее функцией load ОС РВ загружается файл userfunc.o. В режиме записи ПО загрузчик после чтения из технологической ЭВМ очередного файла, указанного в тэге ReplFile (см. рис. 3), всегда вызывает userfunc. При этом кроме значений атрибутов в тэге ReplFile в userfunc передаются и значения атрибутов тэга cpu. Часть этих атрибутов, на рис. 3 не приведенная, нужна только userfunc. Значения конфигурационного параметра rconfflag специфицируют выполняемые функцией userfunc и загрузчиком действия:

- ws — запись файлов в прибор, подсчет контрольных сумм и их сохранение в доступных файловых системах; запись контрольной информации во флэш-память;
- w — запись без подсчета контрольных сумм, но какие-то действия userfunc при этом выполняет;
- s – те же действия, что и для ws, но без записи прочитанных из технологической ЭВМ данных;
- v — проверка целостности ПО и контрольной информации.

Userfunc выполняется только в главном ПЭ прибора. Интерфейс взаимодействия загрузчика и userfunc описан в документации на загрузчик. Загрузчик при вызове передает userfunc всю необходимую ей информацию из файла ЦОС и данные о прочитанных из технологической ЭВМ файлах. Следует отметить, что, как и замышлялось, разработчики загрузчика не обладают полной информацией о действиях, реализуемых userfunc. Для их выполнения загрузчик предоставляет userfunc функции для:

- чтения/стирания/записи флэш-памяти в любом ПЭ прибора;
- чтения/записи/удаления файлов для любой файловой системы vfat прибора;
- чтения/записи/удаления файлов в файловой системе NFS.

Заключение

Рассмотрено технологическое программное обеспечение - RIO-загрузчик, используемый при разработке и эксплуатации распределенных программно-аппаратных комплексов на базе коммуникационной среды RapidIO, выполненных на отечественных микросхемах семейства Комдив. Используя структурированное описание аппаратуры и программного обеспечения комплекса, полученное с помощью программных инструментов, реализованных в рамках проекта ОС РВ Багет, RIO-загрузчик выполняет инициализацию коммуникационной среды, загрузку исполняемых образов ОС РВ в процессорные элементы комплекса на этапах разработки в эксплуатации, запись программного обеспечения во флэш-память и флэш-диски на этапе подготовки комплекса к штатной эксплуатации. RIO-загрузчик также предоставляет требуемую среду выполнения для программы контроля целостности программного обеспечения и аппаратуры комплекса. Данная программа разрабатывается специалистами организаций, разрабатывающих прикладное программное обеспечение комплекса.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований.ГП 14 по теме (проекту) «Исследование методов и средств реализации контролируемого выполнения приложений, функционирующих в реальном масштабе времени». (№0065-2018-0011).

Booting of multiprocessor RapidIO-system

Y.M. Lazutin

Abstract: The paper describes software for booting multiprocessor RapidIO system with proper executable RTOS-modules according to predefined mapping scheme. The booter also provides operating environment for tools checking integrity of resident software.

Keywords: RT OS Baget, RapidIO, multiprocessos system booting, software integrity checking

Литература

1. С.Г.Бобков. Импортозамещение элементной базы вычислительных систем – “Вестник Российской Академии Наук”, 2014, том 84, № 11, с. 1010 –1016.
2. О.В.Сердин, С.Г.Бобков, Н.В.Кондратьева, А.А.Еремин. Разработка высоконадежных многопроцессорных модулей на базе высокоскоростных каналов RapidIO. «Программные продукты и системы». 2013, №4, 49-55
3. RapidIO Interconnect Specification. Part1: Input/Output Logical Specification. Revision 3.1 – RapidIO Trade Association, 2014.
4. RapidIO Interconnect Specification. Part 2: Message Passing Logical Specification. Revision 3.1 – RapidIO Trade Association, 2014.
5. А.Н.Годунов, В.А.Солдатов, Операционные системы семейства Багет (сходства, отличия и перспективы) – «Программирование», Москва, 2014, №5, с. 68 -76.
6. А.Н.Годунов, В.А.Солдатов, Конфигурирование многопроцессорных систем в операционной системе реального времени Багет – «Программная инженерия», Москва, 2016, №6, с. 243 -251.
7. Г.О.Райко, Ю.А.Павловский, В.С.Мельканович. Технология программирования многопроцессорной обработки гидроакустических сигналов на вычислительных устройствах семейства «КОМДИВ» – Научно -технический сборник «Гидроакустика», СПб.: Концерн «Океанприбор», 2014, вып. 20(2), с. 85 -92.
8. https://git.busybox.net/busybox/tree/shell/hush.c
9. Бетелин В. Б., Галатенко В. А., Костюхин К. А., Дзабраев М. Д. Контролируемое выполнение сложных систем // Научный сервис в сети Интернет: труды XIX Всероссийской научной конференции (18-23 сентября 2017 г., г. Новороссийск). — М.: ИПМ им. М. В. Келдыша, 2017. — С. 63-71. — URL: http://keldysh.ru/abrau/2017/72.pdf doi:10.20948/abrau-2017-72

Программная реализация поддержки передачи данных специальных типов IEC61131-3 в библиотеке мониторинга

А.И. Грюнталь[1], К.Г. Нархов[2], А.М. Щегольков[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,.

E-mail's: [1]grntl@niisi.ras.ru, [2]kostas@niisi.ras.ru, [3]ashch@niisi.ras.ru

Аннотация: В статье рассматривается возможность использования библиотеки мониторинга для контроля выполнения и отладки прикладных программ для программируемых логических контроллеров (ПЛК). Использование библиотеки мониторинга для ПЛК требует поддержки специальных типов языка программирования Structured Text (ST) стандарта IEC61131-3 на уровне средств приема-передачи команд, входящих в состав библиотеки мониторинга. В данной статье рассмотрены варианты расширения функциональных возможностей библиотеки мониторинга, а также предложены решения, реализующие средства отладки прикладных программ.

Ключевые слова: контролируемое выполнение, исключительная ситуация, исключение, библиотека мониторинга, многопоточная программа, поток управления, сигнал, операционная система реального времени, многопроцессорная система.

Введение

Библиотека мониторинга (БМ) [2, 5, 6] реализует функции, которыми должна обладать система, спроектированная в соответствии с принципами контролируемого выполнения [3, 4]. В соответствии с [3], программа с контролируемым выполнением – это программа со встроенными механизмами, обеспечивающими выполнение программы в критических условиях (например, при поступлении в программу некорректных данных, ошибок в среде исполнения или в самой прикладной программе). Средства контролируемого выполнения прикладной программы содержат механизмы, обеспечивающие контроль состояния выполняемой программы, сравнение параметров состояния прикладной программы с эталоном, а также генерацию управляющих воздействий в случае отклонения поведения программы от эталона.

Библиотека мониторинга обеспечивает выполнение контроля работоспособности, корректности и целостности прикладной программы на основании данных, получаемых от агентов мониторинга, аккумулирующих данные о выполнении прикладной программы (контрольные параметры состояния).

В статье рассматривается возможность использования библиотеки мониторинга для контроля выполнения и отладки прикладных программ, выполняющихся в среде операционной системы реального времени Багет 2.6, для программируемых логических контроллеров (ПЛК). ОС РВ Багет 2.6 является отечественной операционной системой, которая разработана в ФГУ ФНЦ НИИСИ РАН и предназначена для создания программного обеспечения программно-аппаратных комплексов, работающих в режиме реального времени [0].

1. Алгоритм функционирования библиотеки мониторинга

Библиотека мониторинга предназначена для реализации контролируемого выполнения многопоточных приложений распределенных многопроцессорных вычислительных систем [3, 4]. Библиотека мониторинга обеспечивает сбор данных о состоянии пользовательских потоков управления, генерацию и передачу в пользовательские потоки управляющих воздействий в том случае, когда требуется изменение режима выполнения приложения, не предусмотренное алгоритмом функционирования приложения [2, 5, 6].

Библиотека мониторинга – это инфраструктурная программа, представляющая

собой расширение приложения, контролирующее прикладную программу. Встраивание библиотеки мониторинга в готовое приложение требует минимальных изменений в исходном тексте приложения и в случае штатного исполнения не меняет алгоритм приложения. Состав БМ: агенты, монитор, средства сбора данных, средства передачи команд. Агенты представляют собой встроенные в приложение функции, регистрирующие данные – параметры выполнения. Номенклатура параметров выполнения определяется разработчиком в зависимости от функций и назначения приложения. Средства передачи команд – это совокупность средств синхронизации, входящих в ОС РВ Багет 2.6 и протокола передачи данных верхнего уровня, реалиизованного в библиотеке мониторинга.

Реализация протокола передачи данных содержит принципиальные ограничения, исключающие использование библиотеки мониторинга для отладки прикладных программ ПЛК, написанных в соответствии со стандартом IEC61131-3 [7]. В этой статье предложены варианты расширения функциональных возможностей библиотеки мониторинга, необходимые для снятия этих ограничений. Параметры состояния прикладной программы передаются средствами библиотеки мониторинга в монитор для анализа параметров и, в случае необходимости, выработки управляющих воздействий – команд, влияющих на режим выполнения приложения. Наряду с командами, изменяющими режим (или алгоритм) приложения, в БМ реализованы команды, которые могут выполнять общесистемные действия, например – приостанавливать или повторно запускать потоки управления. Вычислительная система, на которой устанавливается библиотека мониторинга, представляет собой однопроцессорную или многопроцессорную вычислительную систему. Средства передачи монитору параметров состояния инвариантны относительно физического уровня, используемого для межпроцессорной передачи данных. Поэтому в качестве среды передачи данных могут использоваться интерфейсы RS232, RS422, RS485, Ethernet и RapidIO.

2. Описание протокола передачи данных

Агенты библиотеки мониторинга, регистрирующие и передающие контрольные параметры состояния, представляют собой функции, выполняемые в контексте прикладного потока управления. Параметры состояния содержат идентификационную информацию о прикладном потоке управления, в контексте которого исполняется агент, о группе функциональных потоков управления, в состав которой входит поток управления, и контрольный идентификатор состояния программы. Один поток управления может содержать несколько агентов, поэтому параметры состояния содержат также идентификационную информацию об агенте, сгенерировавшем эти параметры [2]. Агент мониторинга – это функция библиотеки мониторинга, вызов которой размещается в контрольной точке приклад-ной программы. Прототип функции агента библиотеки мониторинга:

```
int checkpoint_agent(
  struct gfp_thread_pool *tpool,
  int usercheckpoint_id,
  int userdata);
```

Этой функции передаются следующие аргументы:

– **tpool** – указатель на структуру с атрибутами потока управления, в контексте которого выполняется агент;

– **usercheckpoint_id** – идентификатор агента (номер контрольной точки);

– **userdata** – данные для передачи в монитор.

При вызове агента мониторинга параметры состояния прикладной программы (переданные в агент как аргументы функции) помещаются в очередь сообщений. Каждое сообщение в очереди сообщений имеет следующий формат [2]:

– битами **0-7** кодируется идентификатор процессора, на котором выполняется группа функциональных потоков управления;

– битами **8-15** кодируется идентификатор группы функциональных потоков управления, в которую входит поток управления;

– битами **16-23** кодируется числовой идентификатор (тег) потока управления;

– битами **24-31** кодируется идентификационная информация об агенте потока управления;

– битами **32-39** кодируются пользовательские данные – это может быть номер команды или значение переменной **errno**.

Распаковка сообщения выполняется монитором в структуру типа **agentmag_t** с помощью вызова входящей в состав программного модуля **monitor.c** библиотеки мониторинга функции

```
void doMessDecode(
 char * message,
 struct agentmsg_t * mess_block,
 unsigned int PRINT_DEBUG));
```

Структура **agentmag_t** содержит поля типа **char**, соответствующие заданным частям (байтам) сообщения. На основании полученных данных от агента мониторинга монитор формирует управляющую реакцию.

3. Особенности протокола передачи данных

Ограничением рассмотренного выше протокола является размер пользовательских данных. Реализация протокола обеспечивает передачу в одном сообщении 1 байта данных (целое от 0 до 255). Данное ограничение обусловлено назначением и типовым использованием библиотеки мониторинга. Как было отмечено ранее, из агента мониторинга посредством пользовательских данных могут быть переданы номер команды и значение переменной **errno**. Эта переменная используется системными вызовами и некоторыми библиотечными функциями ОС РВ при ошибках для указания того, что именно произошло [0]. Реализация библиотеки мониторинга поддерживает 4 команды (завершение потока управления, останов потока управления, запуск остановленного потока управления, перезапуск потока управления), при этом переменная **errno** в ОС РВ Багет 2.6 принимает значения от нуля до 95. Следует отметить, что в коротких сообщениях, передаваемых из потока управления средствами библиотеки мониторинга, используются данные размером не более 1 байта (биты **32-39** в сообщении).

4. Применение библиотеки мониторинга для программируемых логических контроллеров

В силу универсального характера заложенных в библиотеку мониторинга технических решений возможна модификация библиотеки мониторинга с целью её применение в задачах управления ПЛК.

ПЛК представляют собой автономно функционирующие ЭВМ, управляющие техническими объектами по заранее заданному (для конкретных условий эксплуатации) алгоритму. ПЛК осуществляет сбор данных с датчиков, устанавливаемых на техническом объекте, анализ данных и передачу управляющих воздействий.

В типовом случае несколько ПЛК взаимодействуют по сети с управляющей ЭВМ (УЭВМ). Цель управления – контроль за работой ПЛК, оперативное изменение режима работы ПЛК, сбор статистики о выполнением производственных процессов. Таким образом, ПЛК (один или несколько) представляет собой объект управления со стороны удалённой ЭВМ.

Возможны более сложные схемы взаимодействия ПЛК и УЭВМ. ПЛК, УЭВМ взаимодействующие по сети, специальное программное обеспечение ПЛК и УЭВМ образуют АСУ технологическим процессом (АСУ ТП).

4.1. Специальные типы данных IEC61131-3

Библиотека мониторинга может быть использована для контроля выполнения программ для ПЛК. Исходный код прикладной программы для ПЛК разрабатывается на языке программирования Structured Text (ST) стандарта IEC61131-3.

Особенностью этого языка программирования является наличие специальных типов данных **BYTE**, **WORD**, **DWORD** и **LWORD**, соответствующих по длине следующим типам языка Си: **char**, **short**, **int** и **long**. Поддержка специальных типов ST на уровне протокола передачи данных библиотеки мониторинга позволяет обеспечивать контроль состояния прикладной программы и, в случае необходимости, генерацию управляющих воздействий.

В соответствии с реализацией текущего протокола передачи данных и вытекающими из нее ограничениями, которые кратко описаны выше, библиотека мониторинга поддерживает только специальный тип **BYTE** языка ST.

Библиотека мониторинга может использоваться для удаленной отладки прикладной программы. В текущей реализации отладка предполагает просмотр и протоколирование значений переменных: в контрольных точках прикладной программы устанавливаются агенты мониторинга, в которые передаются значение отлаживаемых переменных. Трасса вызовов агентов мониторинга протоколируется монитором и впоследствии исследуется прикладным программистом. Текущая реализация библиотеки мониторинга не предполагает отладку в реальном времени.

В текущей реализации библиотека мониторинга используется только для отладки **char** (или **BYTE** в терминах ST) переменных.

Расширение функциональных возможностей библиотеки мониторинга, которые следует реализовать для устранения рассмотренных выше ограничений:

– поддержка специальных типов данных **WORD**, **DWORD** и **LWORD** языка программирования ST стандарта IEC61131-3;

– возможность удаленной модификации значения переменной в заданной контрольной точке;

– возможность удаленной отладки (просмотр и модификация значений переменных) прикладной программы в реальном времени.

4.2. Поддержка передачи данных специальных типов IEC61131-3

Структура прикладной программы, работающей совместно с библиотекой мониторинга на многопроцессорной системе, представлена на рисунке 1. При работе библиотеки мониторинга на многопроцессорной системе используются удалённый (главный) и локальный (подчиненный) мониторы. Главный монитор назначается при конфигурировании библиотеки мониторинга, при этом прочие мониторы считаются подчиненными [5]. На рисунке 1 представлены компоненты библиотеки мониторинга, которые функционально связаны с передачей данных и требуют изменения:

– агент мониторинга: в исходном тексте прикладной программы размещены вызовы функций агентов мониторинга, посредством которых выполняется связь с локальным монитором;

– локальный монитор: в однопроцессорной системе выполняет функции удаленного монитора, в многопроцессорной системе выполняет прием-передачу данных из удаленного монитора (в удаленный монитор) через подпрограмму связи с удаленным монитором;

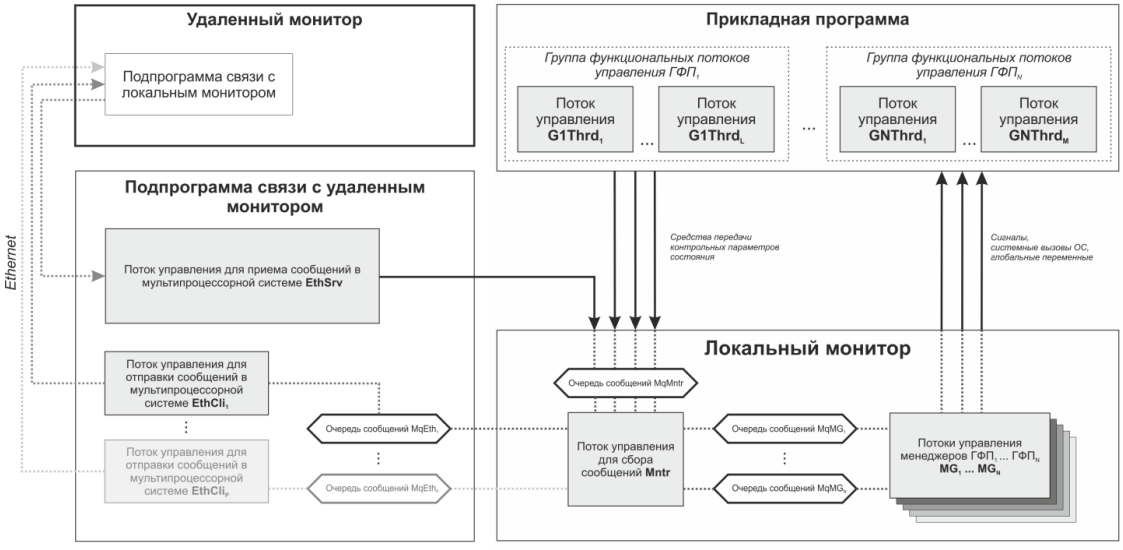


Рисунок 1. Библиотека мониторинга в многопроцессорной системе

– подпрограмма связи с удаленным монитором: обеспечивает передачу данных между локальным и удаленным монитором по протоколу TCP/IP;

– удаленный монитор: принимает данные из локального монитора и генерирует управляющие воздействия, которые передаются обратно в локальный монитор.

4.3. Новый формат сообщения

Поддержка специальных типов данных **WORD**, **DWORD** и **LWORD** языка программирования ST стандарта IEC61131-3 требует изменение формата сообщения библиотеки мониторинга:

– битами **0-7** кодируется идентификатор процессора, на котором выполняется группа функциональных потоков управления;

– битами **8-15** кодируется идентификатор группы функциональных потоков управления, в которую входит поток управления;

– битами **16-23** кодируется числовой идентификатор (тег) потока управления;

– битами **24-31** кодируется идентификационная информация об агенте потока управления приложения;

– битами **32-39** кодируется информация о пользовательских данных: специальный тип данных (**BYTE**, **WORD**, **DWORD** и **LWORD**), тип сообщения (отладочное сообщение, команда, цепочка команд или данные), количество команд в цепочке, флаг ожидания (агент, передавший сообщение, приостанавливает поток управления, в контексте которого он выполнен);

– битами **40-127** кодируются пользовательские данные.

Таким образом, сообщение состоит из заголовка (0-39 бит) и данных (40-127) бит и имеет длину 16 байт.

4.4. Использование цепочек команд и команд с параметром при мониторинге прикладной программы

Сообщения нового формата позволяют библиотеке мониторинга не только пересылать пользовательские данные специальных типов **WORD**, **DWORD** и **LWORD** языка программирования ST стандарта IEC61131-3, но и использовать в сообщении несколько команд.

Командами в библиотеке мониторинга называются управляющие воздействия, передаваемые из монитора в заданный поток управления прикладной программы.

Размер команды составляет 1 байт, поэтому в сообщении нового формата можно уместить до 11 команд (биты **40-127** в сообщении).

Следует отметить, что новый формат сообщения позволяет доставлять в поток управления команду с параметром. В качестве параметра могут использоваться следующие значения:

– один аргумент специального типа **LWORD**;

– два аргумента специального типа **DWORD**;

– четыре аргумента специального типа **WORD**;

– восемь аргументов специального типа **BYTE**.

Команда с параметром необходима для реализации возможности удаленной модификации значения переменной в заданной контрольной точке, а также для удаленной отладки.

5. Оценка производительности передачи данных специальных типов IEC61131-3

Для оценки производительности передачи данных специальных типов IEC61131-3 библиотекой мониторинга была использована контрольная задача, состоящая из шести групп функциональных потоков управления. В каждой группе работало по три потока управления, и в каждом потоке управления было установлено не более пяти контрольных точек (агентов мониторинга). Группы функциональных потоков управления были развернуты на двух процессорах по схеме 50/50: три группы на один процессор, три – на другой.

Каждый из потоков управления выполнил 100 тысяч итераций. При выполнении потоков управления монитор реагировал на искусственно порождаемые в потоках управления сбои и направлял в эти потоки управления необходимые команды, соответствующие сбоям.

По результатам выполнения контрольной задачи отмечено, что экспериментальная версия библиотеки мониторинга со встроенной поддержкой специальных типов IEC61131-3 функционально совместима с версией библиотеки без поддержки специальных типов. Однако является менее производительной. Отмечено замедление выполнения контрольной задачи на 15% по сравнению с версией библиотеки без поддержки специальных типов [2, 6, 7].

Исследование причин, лежащих в основе замедления работы библиотеки, предлагаются определило следующие способы ускорения работы библиотеки мониторинга со встроенной поддержкой специальных типов IEC61131-3:

– оптимизация алгоритма упаковки/ распаковки сообщения;

– реализация кэширования сообщений, например, если агент мониторинга фиксирует на итерации n состояние идентичное ранее переданному на итерации

n-1 состоянию, то данные в монитор не передаются и агент мониторинга использует последнюю полученную от монитора команду (без ожидания приема-передачи этой команды от монитора);

– децентрализация многопроцессорной системы: часть функций удаленного монитора делегировать локальному монитору, например, использовать локальный монитор для протоколирования работы группы функциональных потоков управления (сохранение протокола работы по протоколу сетевого доступа к файловым системам nfs).

6. Использование библиотеки мониторинга для отладки прикладных программ для ПЛК

Под отладкой прикладных программ для ПЛК понимается работа по просмотру и модификации значений переменных прикладной программы, выполняемой на целевой ЭВМ (ПЛК). Эта работа выполняется с помощью интегрированной среды разработки, установленной на инструмен-тальной ЭВМ. Для реализации функциональных возможностей отладки следует расширить функциональные возможности библиотеки мониторинга следующим образом:

– портировать функционал удаленного монитора в интегрированную среду разработки на инструментальной ЭВМ (инструментальный монитор);

– доработать библиотеку мониторинга в части конфигурирования: на этапе конфигурирования библиотеки требуется задать значения для доступа к ИЭВМ (IP-адрес, порт и т. д.);

– реализовать маршрутизацию сообщений в инструментальный монитор из удаленного монитора;

– реализовать графический интерфейс для отображения отладочной информации и ввода значений для отлаживаемых переменных.

7. Заключение

В результате изменения формата и расширения размера сообщения в протоколе передачи данных, используемом в библиотеке мониторинга, удалось реализовать поддержку специальных типов данных стандарта IEC61131-3 **`WORD`**, **`DWORD`** и **`LWORD`** языка программирования ST стандарта IEC61131-3.

Поддержка специальных типов данных стандарта IEC61131-3 позволяет использовать библиотеку мониторинга для контроля выполнения прикладных программ на программируемых логических контроллерах.

В библиотеку мониторинга встроена функциональная возможность выполнения приема-передачи в одном сообщении групп команд (цепочек команд), а также команд, сопровождаемых пользовательскими данными.

Выполнена оценка производительности библиотеки мониторинга с новыми функциональными возможностями и предложены способы её ускорения.

Планируется расширить библиотеку мониторинга средствами удаленной отладки прикладных программ для программируемых логических контроллеров.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований по теме (проекту) «39. Архитектура, системные решения, программное обеспечение, стандартизация и информационная безопасность информационно-вычислительных комплексов и сетей новых поколений.

Разработка методов и средств контролируемого выполнения приложений, функционирующих в реальном масштабе времени (№ 0065–2018–0011), АААА–А18–118041190171–0.

.

Handling Exceptions Using the Monitor Library

A. Gryuntal, K. Narkhov, A. Shchegolkov

Abstract: The article discusses the methods of using the monitoring library for controlling and debugging programmable logic controllers (PLCs). The monitoring library requires the support of special types, provided by the programming language Structured Text (ST) of IEC 61131-3 standard, at the level of tools for receiving and transmitting commands included in the monitoring library. This article discusses options for expanding the monitoring library functionality, and proposes solutions that implement the tools for debugging applications.

Keywords: controlled execution, exception, monitor library, multithread program, thread, signal, real-time operating system, multiprocessing system.

Литература

1. А.Н.Годунов, В.А.Солдатов. Операционные системы семейства Багет (сходства, отличия и перспективы) – «Программирование», Москва, 2014, №5, с. 68-76.
2. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Реализация принципа контролируемого выполнения для прикладных программ реального времени. // Труды НИИСИ РАН. – Том 5 № 2. Построение программ реального времени. ISSN 2225-7349, Москва, 2015, с. 113-121.
3. В.А. Галатенко, Контролируемое выполнение / В.А. Галатенко, К.А. Костюхин, К.А., Н.В. Шмырев – М: НИИСИ РАН, 2012. – 157 с.
4. В.Б. Бетелин, Контролируемое выполнение с явной моделью / В.Б. Бетелин, В.А. Галатенко, К.А. Костюхин – ПРОГРАММИРОВАНИЕ, 2014, N- 6, с. 45-55.
5. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Библиотека мониторинга для многопоточных программ. // Труды НИИСИ РАН. – Том 7 № 1. Построение программ реального времени. ISSN 2225-7349, Москва, 2017, с. 70-74.
6. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Обработка исключительных ситуаций с использованием библиотеки мониторинга. // Труды НИИСИ РАН. – Том 7 № 4. Построение программ реального времени. ISSN 2225-7349, Москва, 2017, с. 96-101.
7. International Standard IEC 61131-3, Programmable controllers –Part 3: Programming languages, Second edition, 2003, IEC Central Office, Geneva, ISBN 2-8318-6653-7.

Визуализация параметров состояния прикладной программы на персональной ЭВМ Багет Р2А

А.И. Грюнталь[1], К.Г. Нархов[2], А.М. Щегольков[3]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]grntl@niisi.ras.ru, [2]kostas@niisi.ras.ru, [3]ashch@niisi.ras.ru

Аннотация: Статья посвящена вопросам реализации средств визуализации параметров состояния многопоточной прикладной программы в среде операционной системы реального времени, функционирующей на многопроцессорной вычислительной системе. В данной статье рассмотрены варианты расширения функциональных возможностей библиотеки мониторинга, а также предложены решения, реализующие средства отладки прикладных программ в интерактивном режиме.

Ключевые слова: контролируемое выполнение, библиотека мониторинга, многопоточная программа, поток управления, очередь сообщений, сигнал, операционная система реального времени, многопроцессорная система, отладка.

Введение

Актуальной задачей при использовании библиотеки мониторинга (БМ) [1, 2, 3] является визуализация параметров состояния потоков управления прикладной программы, выполняющейся в многопроцессорной системе под управлением операционной системы реального времени семейства Багет (ОСРВ Багет 2.6 [4]).

В предыдущей статье [3] был рассмотрен встроенный в БМ механизм протоколирования параметров состояния в файл, а также методы анализа и интерпретации сохраненного протокола. Однако стоит отметить, что эти методы являются относительно трудоемкими и требуют специальных навыков, поэтому не могут быть использованы в рамках так называемого «быстрого старта» использования БМ.

Для упрощения использования БМ, в том числе для мониторинга работоспособности многопроцессорной системы и сокращения времени ее отладки, предлагается использование удаленной инструментальной ЭВМ (ИЭВМ) с предустановленными инструментами визуализации параметров состояния (ИВПС) – программным обеспечением, реализующим интерактивный оконно-диалоговый режим взаимодействия с программистом и написанным с использованием свободно распространяемых графических библиотек [5].

В качестве ИВПС может использоваться интерактивная среда разработки технологических средств автоматизированной генерации специального программного обеспечения (ТСАГ СПО) [6] или среда Eclipse для ИЭВМ на базе микропроцессора Intel x86/x64 под управлением ОС Fedora 27 NIISI. В качестве ИВПС для ИЭВМ архитектуры mips64 под управлением ОС Astra Linux (персональная ЭВМ Багет Р2А) используется приложение визуализации параметров состояния (ПВПС).

Использование ПЭВМ Багет Р2А является приоритетным решением для мониторинга, так как этот программно-аппаратный комплекс, включающий исследуемую многопроцессорную систему и средства мониторинга, полностью построен на отечественных аппаратуре и ПО и обеспечивает требование импортозамещения для многопроцессорных систем.

1. Требования к приложению визуализации параметров состояния

Для визуализации параметров состояния прикладной программы требуется реализация функциональной возможности маршрутизации данных удаленным монитором БМ. Под маршрутизацией данных понимается прием-передача данных удаленным монитором без анализа,

модификации или протоколирования, а монитором из локальных мониторов многопроцессорной системы, передаются (маршрутизируются) на ИЭВМ, а данные, полученные от ИЭВМ удаленным монитором, передаются (маршрутизируются) в локальные мониторы.

Для реализации маршрутизации данных удаленным монитором БМ следует реализовать следующие функциональные возможности:

– расширение параметров конфигурирования БМ: флаг установки режима маршрутизации на удаленном мониторе, IP адрес и порт ИЭВМ, локальный порт для приема данных из ИЭВМ, политика маршрутизации (передача всех команд/данных, только ненулевых команд/данных, передача заданных программистом команд/данных);

– установка алгоритма маршрутизации в головной функции удаленного монитора.

Для визуализации параметров состояния прикладной программы в ПВПС требуется реализация монитора для ПВПС с функцией визуализации данных в интерактивном оконно-диалоговом режиме. Монитор для ПВПС – это многопоточное приложение, состоящее из:

– потока управления монитора, который функционирует по алгоритму удаленного монитора БМ без маршрутизации;

– потока управления интерактивным диалоговым режимом;

– потока синхронизации данных, передаваемых монитором ПВПС в многопроцессорную систему, и данных, которые вводит программист в интерактивном оконно-диалоговом режиме.

2. Маршрутизация данных удаленным монитором

Для визуализации параметров состояния прикладной программы удаленный монитор должен маршрутизировать данные на ИЭВМ с ПВПС. Параметры маршрутизации данных задаются на этапе конфигурирования БМ с помощью структуры **`libcontrol_mps_conf`**, поля которой используются для активации функции маршрутизации данных:

– **`routing`** – включение/выключение маршрутизации данных удаленным монитором;

– **`route_from_port`** – порт удаленного монитора, с которого будут маршрутизироваться данные; именно: данные, принятые удаленным

– **`route_to_ip`** – IP адрес ИЭВМ, на который будут маршрутизироваться данные удаленным монитором;

– **`route_to_port`** – порт ИЭВМ, на который будут маршрутизироваться данные удаленным монитором;

– **`route_policy`** – политика маршрутизации: передача всех данных, только ненулевых данных, передача заданных программистом данных.

Конфигурирование БМ может быть выполнено в диалоговом режиме с помощью технологического расширения (плагина) ТСАГ СПО для интегрированной среды разработки Eclipse.

Удаленный монитор маршрутизирует данные на ПЭВМ Багет Р2А, на которой установлено ПВПС.

В состав ПВПС входит подсистема связи с библиотекой мониторинга (ПСБМ). Интерактивный интерфейс ПВПС состоит из главного окна и одного или нескольких подчиненных окон.

На главном окне отображается многопроцессорная система в виде следующей иерархической структуры: идентификатор процессора – группа функциональных потоков управления – поток управления (рисунок 1).

При двойном нажатии правой кнопкой мыши на заданном потоке управления появляется окно с таблицей параметров состояния этого потока управления.

Параметры состояния визуализируются в соответствующей ячейке таблицы (рисунок 2) и меняются динамически в соответствии с поступлением от удаленного монитора.

Если темп поступления параметров состояния превышает время, установленное в настройках ПВПС, то таблица обновляется в соответствии со временем, заданным в настройках ПВПС.

3. Визуализация параметров состояния данных на ПЭВМ Багет Р2А

Визуализация параметров состояния в интерактивном режиме позволяет оценить корректность работы многопроцессорной системы, а также проанализировать работоспособность заданных потоков управления в реальном времени.

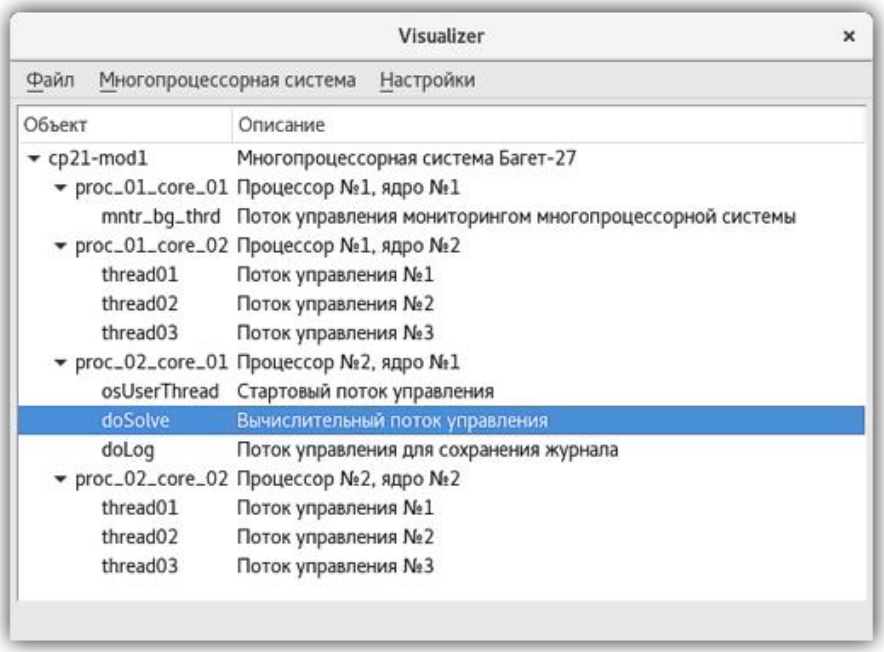


Рисунок 1. Главное окно ПВПС

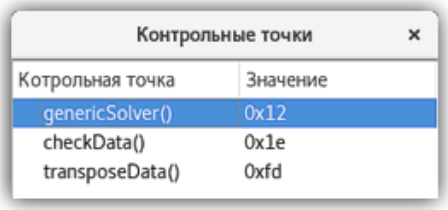


Рисунок 2. Подчиненное окно визуализации параметров состояния

Визуализация выполняется на ПЭВМ Багет Р2А на базе микропроцессора Комдив64 с предустановленной ОС Astra Linux, включающей средства разработки Qt4. Для компиляции ПВПС для ОС Astra Linux следует загрузить в файловую систему ИЭВМ репозиторий с исходными текстами подсистемы, выполнив команду:

```
git clone
https://gitlab.com/pheix-
qt/visualizer
```

Инструкции по компиляции, сборке и запуску ПВПС приведены в файле **README.md**.

Для начала работы с удаленной многопроцессорной системой следует загрузить в ПВПС файл конфигурации БМ для этой многопроцессорной системы, используя подпункт «Открыть конфигурацию» пункта «Файл» главного меню ПВПС. После успешного открытия и импорта данных из файла конфигурации многопроцессорная система будет отображена в таблице на главном окне ПВПС (рисунок 1).

Далее следует активировать ПСБМ, которая выполняет прием-передачу данных из многопроцессорной системы, используя подпункт «Включить ПСБМ» пункта «Многопроцессорная система» главного меню ПВПС. Для деактивации ПСБМ следует использовать подпункт «Отключить ПСБМ» пункта «Многопроцессорная система» главного меню ПВПС. Если ПСБМ не может быть включена или выключена, то соответствующий подпункт меню «Многопроцессорная система» будет неактивен.

После успешной активации ПСБМ следует выбрать некоторый поток управления из списка на главном окне ПВПС и с помощью двойного щелчка левой клавиши мыши перейти на подчиненное окно визуализации параметров состояния заданного потока управления (см. пример на рисунке 2). На этом окне отображается таблица с контрольными точками заданного потока (строки). Для каждой контрольной точке в столбце «Значение» отображается последний принятый ПСБМ параметр состояния. Время обновления таблицы

задается в настройках ПВПС (пункт «Настройки» главного меню ПВПС) и по умолчанию составляет 100 мс.

Для визуализации параметра состояния для заданной контрольной точки в виде графика (зависимость значение-время) следует выбрать контрольную точку из списка на подчиненном окне визуализации параметров состояния ПВПС и с помощью двойного щелчка левой клавиши мыши перейти на окно визуализации параметра состояния в заданной контрольной точке (рисунок 3).

Рисунок 3. Окно визуализации параметра состояния в заданной контрольной точке

Следует отметить, что все параметры состояния многопроцессорной системы, принимаемые и отправляемые ПСБМ, сохраняются в файл-протокол, который может быть проанализирован и интерпретирован спустя некоторое время после завершения работы многопроцессорный системы (так называемый “черный ящик”).

4. Выводы

В статье приведена типовая архитектура мониторинга и отладки многопроцессорных систем, функционирующих в среде операционной системы реального времени. В качестве операционной системы реального времени использовалась операционная система ОС РВ Багет 2.6. Возможно использование других версий операционных систем реального времени семейства ОС РВ Багет 2.0.

Цель мониторинга – отладка прикладной программы, достижение устойчивой работы программы путем визуализации параметров состояния многопроцессорной системы в реальном времени.

Предложена программа визуализации параметров состояния, работающая в интерактивном оконно-диалоговом режиме на отечественной персональной ЭВМ Багет Р2А.

Рассмотрены методы и особенности работы с программой.

Дальнейшая работа связана с реализацией механизма визуализации параметров состояния, поиска по параметрам и фильтров отображения параметров (списков отображения).

The Application Program State Parameters Visualization on the Personal ComputerBaget R2A

A. Gryuntal, K. Narkhov, A. Shchegolkov

Abstract: Data monitoring plays a key role in multi-threaded applications. Visualization is an important approach to helping multiprocessing system get a complete view of execution status and discover data values in real time. This article discusses options for expanding the monitoring library functionality and proposes solutions for interactive debugging and data visualization.

Keywords: controlled execution, exception, monitor library, multithread program, thread, signal, real-time operating system, multiprocessing system, debugging, data visualization.

Литература

1. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Реализация принципа контролируемого выполнения для прикладных программ реального времени. // Труды НИИСИ РАН. – Том 5 № 2. Построение программ реального времени. ISSN 2225-7349, Москва, 2015, с. 113-121.
2. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Библиотека мониторинга для много-поточных программ. // Труды НИИСИ РАН. – Том 7 № 1. Построение программ реального времени. ISSN 2225-7349, Москва, 2017, с. 70-74.
3. А.И. Грюнталь, К.Г. Нархов, А.М. Щегольков, Обработка исключительных ситуаций с использованием библиотеки мониторинга. // Труды НИИСИ РАН. – Том 7 № 4. Построение программ реального времени. ISSN 2225-7349, Москва, 2017, с. 96-101.
4. А.Н.Годунов, В.А.Солдатов. Операционные системы семейства Багет (сходства, отличия и перспективы) – «Программирование», Москва, 2014, №5, с. 68-76.
5. Alan Ezust. An introduction to design patterns in C++ with QT / Alan Ezust, Paul Ezust. Pearson Education, 2012. 764 с.
6. К.Г.Нархов. Генератор текста программ в исходном виде для систем реального времени, Тверь: Программные продукты и системы. 2010. № 4. С. 24-30.

Исполнение программного кода, хранящегося во флэш-памяти

Н.Д. Байков[1], А.Н. Годунов[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail's: [1]nknikita@niisi.ras.ru, [2]nkag@niisi.ras.ru

Аннотация: В работе изучается целесообразность исполнения хранящегося во флэш-памяти (вместо оперативной памяти в условиях её дефицита) программного кода. Предлагается способ оценки снижения производительности в худшем случае. На его основе проводится анализ результатов для нескольких отечественных процессорных модулей, предназначенных для эксплуатации в экстремальных условиях.

Ключевые слова: ОСРВ Багет, масштабирование, конфигурирование, производительность, флэш-память, оперативная память.

Введение

Важным направлением развития вычислительной техники и микроэлектроники является разработка встраиваемых систем, предназначенных для эксплуатации в экстремальных условиях. К подобным системам предъявляются повышенные требования надежности по целому ряду показателей. В случае процессорных модулей это могут быть требования сохранения работоспособности в широком диапазоне температур, при сильной вибрации и перегрузках, в условиях радиационного излучения, а также многие другие. Вполне естественно, что учет всех этих факторов накладывает определенные ограничения на технические характеристики модулей. Одним из проявлений этих ограничений может стать малый объем (в пределах одного или нескольких мегабайт) доступной для работы оперативной памяти. Готовым решением для построения управляющей системы служит операционная система реального времени (ОСРВ) Багет 2.x [1, 2], разработанная в ФГУ ФНЦ НИИСИ РАН. Одним из ее достоинств является гибкая система конфигурирования, позволяющая сократить объем занимаемой операционной системой памяти за счет исключения из целевого образа неиспользуемых частей [3]. Для дальнейшей экономии ресурсов приходится прибегать к нестандартным решениям. Например, редактор связей ОСРВ позволяет поместить секцию программного кода во флэш-память модуля, что в свою очередь расширяет доступный для данных пул оперативной памяти. Однако это оказывает влияние на общую производительность исполняемой программы, поскольку скорость работы с флэш-памятью ниже скорости работы с оперативной памятью. Целью настоящей работы является количественная оценка влияния на производительность программы использования флэш-памяти для хранения секции программного кода на примере процессорных модулей, предназначенных для использования в экстремальных условиях.

Технические характеристики оборудования

Тестирование проводилось при помощи двух экспериментальных процессорных модулей на базе 32-разрядных RISC-процессоров 1907ВМ014 и 1890ВМ2Т архитектуры КОМДИВ-32.

Процессор 1907ВМ014 является примером высоконадежного процессора, предназначенного для эксплуатации в экстремальных условиях. В процессе тестирования использовался процессор, работающий с тактовой частотой 100 МГц. Процессор оснащен кэш-памятью данных первого уровня объемом 8 Кб и кэш-памятью инструкций первого уровня объемом также 8 Кб. Длина строки кэш-памяти инструкций равна 16 байтам, а кэш-память данных может использовать строки длиной 4 или 16 байт. Кэш-память второго уровня отсутствует. Инструкции обрабатываются 8-стадийным конвейером: 2 стадии получения инструкции, чтение регистрового файла, вычисление адреса операнда, выполнение, доступ к памяти, анализ исключений и запись в регистры.

Также изучалась работа близкого к 1907ВМ014 по архитектуре (но предназначенного для использования в более мягких

условиях) процессора 1890ВМ2Т в составе экспериментального процессорного модуля БТ23-205. Второй процессор обладает идентичной структурой кэш-памяти и также работал с тактовой частотой 100 МГц. Существенным для дальнейшего рассмотрения отличием процессора 1890ВМ2Т от процессора 1907ВМ014 является структура конвейера, которая состоит из 5 последовательных стадий: получение инструкции, ее декодирование, выполнение, доступ к памяти и запись в регистры.

В качестве программного обеспечения использовалась экспериментальная версия ОСРВ Багет 2.x.

Методика тестирования

Поскольку архитектура КОМДИВ-32 совместима с архитектурой MIPS, для построения системы тестов применялись те же самые принципы, которые были предложены авторами данной работы в [4]. Далеко не все из них удалось заимствовать без изменений в силу ограничений архитектуры и особенностей реализации. Наиболее простым примером отличия служит механизм очистки кэш-памяти, который вместо неподдерживаемой инструкции `cache` основан на совершенно ином наборе команд. Так как алгоритм тестирования из [4] удается приспособить к сравнению скорости работы оперативной и флэш-памяти лишь частично, в результатах измерений наблюдаются некоторые расхождения. Поскольку они не влияют на порядок оцениваемых величин, мы не будем фокусировать на них свое дальнейшее внимание. Также отметим, что поскольку используются экспериментальные версии модулей, то при реальной эксплуатации результаты измерений в определенной степени могут отличаться от результатов данной работы.

Напомним, что в основе алгоритма лежит изучение времени выполнения тестовых функций с многократным повторением одних и тех же инструкций. Анализ зависимости длительности теста от количества повторений инструкции или количества итераций внешнего цикла позволяет выявить вклад отдельных событий в общую длительность выполнения теста. В грубом приближении можно считать, что время выполнения функции складывается из времени получения данданных из оперативной памяти при промахе в кэш-память данных, времени получения инструкций из оперативной или флэш-памяти при промахе в кэш-память инструкций, а также времени выполнения инструкций из кэш-памяти процессора. Игнорирование параллелизма выполнения инструкций конвейером является очевидным недостатком такого подхода. При анализе полученных результатов мы будем по возможности учитывать указанную особенность, однако следует понимать, что все формулируемые предположения относительно работы процессора носят качественный характер и являются сильным упрощением действительности.

В отдельную группу можно вынести накладные расходы, связанные с выполнением инструкций тестовой системы, которые включают в себя опрос счетчиков времени, инициализации используемых регистров процессора, включение/отключение обработки прерываний и т.д. При увеличении количества повторений тестируемой инструкции влияние накладных расходов на итоговый результат асимптотически уменьшается.

Наибольшее внимание при анализе результатов будет уделяться времени обработки промаха в кэш-память инструкций.

Результаты измерений

В качестве единицы измерения времени будем использовать такты процессора, что более удобно при оценке длительности выполнения отдельных процессорных инструкций, а также сравнения полученных результатов с результатами [4]. Поскольку при тестировании оба процессора работали на одной частоте, преобразование полученных результатов, например, в наносекунды не представляет сложности.

Описание результатов измерений разделим на два этапа. В ходе первого из них будет получена оценка снижения производительности при использовании флэш-памяти в худшем случае. Второй этап будет состоять из синтетического теста производительности, что позволит оценить падение производительности «в среднем».

На первом этапе выполнялись следующие тесты:

- выполнение 1024 последовательных инструкций `addi`;
- выполнение 16 итераций цикла, содержащего 1024 последовательных инструкций `addi`;

	1907BM014				1890BM2T			
	RAM		FLASH		RAM		FLASH	
	К-	К+	К-	К+	К-	К+	К-	К+
`addi`	20 479	4 383	35 558	6 987	5 075	1 350	20 625	2 325
`addi x16`	35 821	19 867	51 054	22 496	20 675	16 875	36 100	17 600
`mul.s`	20 392	6 442	35 654	8 842	6 600	3 425	21 900	4 300
`div+mflo`	23 500	21 404	35 658	23 500	19 025	18 825	29 750	19 600

Табл. 1. Длительность тестов (такты)

– выполнение 1024 последовательных инструкций `mul.s`;
– выполнение по 512 чередующихся инструкций `div` и `mflo`.

Длительность выполнения каждого из тестов, независимо от процессорного модуля и выбора между оперативной и флэш-памятью для хранения секции программного кода, измерялась в двух ситуациях: тестируемая функция отсутствовала (К-) или была размещена (К+) в кэш-памяти инструкций. Между последовательными инструкциями в каждом из тестов присутствовала зависимость по данным. Обращений к кэш-памяти данных при непосредственном выполнении повторяющихся в тесте инструкций не требовалось, поскольку все необходимые данные умещались на регистрах процессора. Результаты измерений приведены в таблице 1.

Представленные результаты позволяют сформулировать цепочку предположений относительно работы процессорных модулей. Начнем с оценки длительности выполнения рассматриваемых инструкций, под которой мы будем подразумевать промежуток времени между началом стадии выполнения инструкции конвейером до момента, когда ее результат станет доступен для следующих инструкций.

Прежде всего, заметим, что даже в тех случаях, когда тестируемые функции находятся в кэш-памяти каждого процессора, результаты измерений для оперативной и флэш-памяти отличаются. По-видимому, расхождение результатов объясняется промахами в кэш-память при выполнении инструкций тестовой системы, т.е. связано с накладными расходами измерений. Чтобы исключить их влияние, мы будем анализировать не абсолютные значения длительности каждого теста, а приращения при переходе от одного теста к другому.

В качестве точки отсчета будем использовать верхнюю строку таблицы 1, в которой указаны результаты измерений для 1024 повторений инструкции `addi`. Инструкция `addi` является одной из наиболее простых процессорных инструкций, для выполнения которой из кэш-памяти на большинстве процессоров достаточно одного такта. Не являются исключением и рассматриваемые процессоры. Убедиться в этом можно путем сравнения друг с другом первых двух строк таблицы 1. Каждая пара значений отличается приблизительно на 15 500 тактов, что соответствует выполнению из кэш-памяти (все инструкции были загружены в кэш-память при выполнении первой итерации цикла) 15 дополнительных итераций цикла по 1024 инструкции `addi`.

Далее выделим в таблице 1 две группы значений. К первой группе отнесем ячейки таблицы, находящиеся в пересечении первой строки со столбцами с названием «К+». Данные значения соответствуют длительностям выполнения 1024 инструкций `addi` из кэш-памяти процессоров. Ко второй группе отнесем аналогичные ячейки из третьей строки таблицы 1. Они соответствуют длительностям выполнения 1024 инструкций `mul.s` из кэш-памяти процессора. Заметим, что значение в каждой из ячеек второй группы превышает значение в соответствующей ячейке из первой группы приблизительно на 2 000 тактов, из чего можно сделать вывод, что для выполнения одной инструкции `mul.s` из кэш-памяти каждого из процессоров требуется по 3 такта.

Аналогичным образом можно оценить длительность выполнения одной пары инструкций `div` и `mflo` из кэш-памяти процессора. На каждом из процессоров для этого потребуется приблизительно 35-36 тактов.

Воспользуемся полученными оценками для измерения длительности обработки промаха в кэш-память инструкций. С этого момента будем рассматривать столбцы таблицы 1 с названием «К-».

Начнем с процессорного модуля на основе 1907BM014. Заметим, что время выполнения первого теста совпадает в пределах

погрешности со временем выполнения третьего теста как при выборке инструкций из оперативной памяти, так и при выборке из флэш-памяти. Попытаемся обосновать полученный результат за счет структуры конвейера. Обработка промахов в кэш-память при получении инструкции осуществляется в самом начале конвейера, тогда как ее выполнение начнется лишь на пятой стадии. Так как в одной строке кэш-памяти помещается ровно 4 инструкции, можно предположить, что стадия выполнения первой инструкции в очередной строке кэш-памяти начнется одновременно с выборкой следующей строки из памяти. Если суммарное время выполнения 4 инструкций `addi` или 4 инструкций `mul.s` не превышает время выборки строки из памяти, длительность первого и третьего тестов без учета накладных расходов равна времени обработки 256 промахов в кэш-память процессора. Под обработкой в данном случае подразумевается выборка строки из оперативной или флэш-памяти, а также продвижение первой ее инструкции до начала стадии выполнения на конвейере. Величина накладных расходов оценивается при помощи результатов выполнения тестов из кэш-памяти процессора по известным значениям длительности выполнения каждой из инструкций. Используя первую строку таблицы 1, получаем, что для обработки промаха в кэш-память в случае оперативной памяти на процессоре 1907ВМ014 требуется приблизительно

$$[(20\,479 - (4\,383 - 1\,024)) / 256] = 66$$

тактов, а при работе с флэш-памятью –

$$[(35\,558 - (6\,987 - 1\,024)) / 256] = 115$$

тактов процессора.

На процессоре 1890ВМ2Т стадия выполнения инструкции конвейером идет третьей по счету, поэтому выборка следующей строки кэш-памяти, вероятно, будет инициирована только после того, как первая половина строки уже будет выполнена. Это позволяет объяснить, например, увеличение времени выполнения теста при замене инструкции `addi` на инструкцию `mul.s`, чего не наблюдалось на процессоре 1907ВМ014. В этом случае можно предположить, что длительность теста без учета накладных расходов примерно оценивается временем выполнения 256 одинаковых циклов, длительность каждого из которых складывается из времени выполнения первых двух инструкций в каждой строке кэш-памяти и времени обработки промаха в кэш-память. Таким образом, из первой строки таблицы 1 получаем, что время обработки промаха в кэш-память при работе с оперативной памятью приблизительно оценивается величиной в

$$[(5\,075 - (1\,350 - 1\,024)) / 256 - 2] = 16$$

тактов, а с флэш-памятью –

$$[(20\,625 - (2\,325 - 1\,024)) / 256 - 2] = 73$$

тактами.

Наконец, рассмотрим результаты последнего теста для пары инструкций `div` и `mflo`. Для них справедливы аналогичные `addi` и `mul.s` рассуждения с той лишь разницей, что `div` является примером инструкции, выполнение которой требует от процессора значительных затрат вычислительных ресурсов. Например, рассмотрим результаты для процессора 1890ВМ2Т.

Для обращения к оперативной памяти на этом процессорном модуле требуется меньше времени, чем для выполнения одной пары инструкций `div` и `mflo`, поэтому длительность выполнения теста из оперативной памяти превосходит длительность выполнения из кэш-памяти только на величину накладных расходов.

Однако при использовании флэш-памяти процессора наблюдается существенное увеличение времени выполнения теста, т.к. промежуток времени между началом стадии выполнения для первой инструкции деления в строке для двух последовательных строк кэш-памяти складывается из времени выполнения первой пары инструкций `div` и `mflo`, а также времени обращения к флэш-памяти. С учетом полученных ранее оценок время выполнения с учетом накладных расходов оценивается величиной

$$(2\,325 - 1\,024) + (36 + 73) * 256 = 29\,205$$

тактов, что достаточно хорошо согласуется с данными таблицы 1.

Проведенные эксперименты позволяют сделать вывод, что при обращении в оперативную память за данными или при выполнении медленных инструкций относительные показатели быстродействия при сравнении флэш-памяти и оперативной памяти сближаются. Следовательно, для наблюдения наибольшего снижения производительности от использования флэш-памяти необходимо выполнение следующих условий:

- инструкции выполняемой функции отсутствуют в кэш-памяти процессора;
- используемые данные находятся в кэш-памяти данных;
- функция состоит из инструкций, «выполняющихся» за 1 такт при загрузке из кэш-памяти процессора.

Размер буферов	1907ВМ014		1890ВМ2Т	
	RAM	FLASH	RAM	FLASH
512	9 904	14 925	2 350	4 225
1 024	13 483	18 670	3 850	5 725
2 048	19 892	24 821	6 850	8 750
4 096	35 433	40 416	12 875	14 775
8 192	64 845	69 642	24 950	26 850

Табл. 2. Длительность копирования данных (такты)

Сформулированным условиям удовлетворяет тест для инструкций `addi`. Таким образом, коэффициент снижения производительности за счет использования флэш-памяти на процессорном модуле с процессором 1907ВМ014 не превышает 115/66 ≈ 1,74, а на процессорном модуле с процессором 1890ВМ2Т - (73+2) / (16+2) ≈ 4,17. Отметим, что больший размер коэффициента во втором случае объясняется относительно высокой скоростью обращения к оперативной памяти.

Перейдем к результатам второго этапа тестирования. Рассмотрим задачу копирования данных из одного буфера в другой при помощи функции `memmove`, которая уже рассматривалась ранее в работе [4]. При рассмотрении ограничимся случаем, когда буфера отсутствуют в кэш-памяти данных процессора, а инструкции функции `memmove` отсутствуют в оперативной или флэш-памяти. Длительности копирования буферов различного размера (в байтах) приведены в таблице 2.

С ростом размеров буферов влияние флэш-памяти на общую производительность закономерно снижается. Главной причиной этого выступает увеличение количества обращений к оперативной памяти для загрузки данных, тогда как количество промахов в кэш-память инструкций фиксировано. Если при работе с буферами малого объема результаты могут существенно отличаться, то уже для буферов размера 8 192 байт падение производительности оказывается в пределах 8%.

Заключение

Проведенное тестирование формально подтверждает гипотезу о том, что использование флэш-памяти для хранения инструкций способно вызвать существенное падение производительности. Были получены оценки коэффициентов снижения производительности из-за ее использования в худшем случае, которые составили приблизительно 1,74 и 4,17 для процессорных модулей на основе процессоров 1907ВМ014 и 1890ВМ2Т соответственно. Тем не менее, результаты задачи копирования демонстрируют, что за счет использования кэш-памяти процессора и активной работы с данными влияние флэш-памяти может быть практически устранено. Использование флэш-памяти можно считать оправданным при выполнении больших пользовательских программ, имеющих сопоставимую с размерами кэш-памяти инструкций часть, выполнение которой занимает больше всего процессорного времени. При этом от прикладного программиста в этом случае требуется осмысленно подходить к размещению функций в памяти, чтобы избегать случайного вытеснения часто использующихся инструкций из кэш-памяти.

Публикация выполнена в рамках государственного задания по проведению фундаментальных научных исследований (ГП14) по теме (проекту) «Исследование принципов построения компактной операционной системы для отечественных радиационностойких процессоров» (№ 0065-2018-0021).

Execution of program code stored in flash memory

N.D. Baykov, A.N.Godunov

Abstract: This paper studies the reasonability of execution of program code stored in flash memory (instead of RAM) under conditions of limited RAM. An approach to the estimation of performance loss in the worst case is proposed. On its basis, analysis of the results for several Russian processor modules designed for extreme environment is performed.

Keywords: RTOS Baget, scaling, configuring, performance, flash memory, RAM.

Литература

1. В.Л. Безруков, А.Н. Годунов, П.Е. Назаров, В.А. Солдатов, И.И. Хоменков. Введение в ос2000 // Вопросы кибернетики, М.: НИИСИ РАН. 1999. с. 76-106.

2. А.Н. Годунов, В.А. Солдатов. Операционные системы семейства Багет (сходства, отличия и перспективы) // Программирование. 2014. № 5. с. 68-76.

3. А.А. Асонов, А.Н. Годунов, В.А. Солдатов. Методы снижения ресурсоемкости операционной системы жесткого реального времени // Труды НИИСИ РАН. 2018. Т. 8. № 1. с. 4-13.

4. Н.Д. Байков, А.Н. Годунов. Сравнение производительности отечественных и импортных микропроцессоров // Программные продукты и системы. 2017. Т. 30. № 3. с. 409-419.

Доверенная загрузка уровня внутреннего программного обеспечения в СнК с архитектурой КОМДИВ

К.А. Пшеничный[1], С.А.Сидоров[2]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's:[1]pshen@cs.niisi.ras.ru, [2]sidorov@niisi.msk.ru

Аннотация: В статье приведены основные понятия доверенных вычислений и доверенной загрузки как элементов предотвращения несанкционированного доступа к данным, умышленного или неумышленного их повреждения. Кратко описаны методы и средства доверенной загрузки, а также технические решения для средств доверенной загрузки, реализуемые в процессорах КОМДИВ.

Ключевые слова: доверенная загрузка, внутреннее программное обеспечение, система на кристалле, архитектура КОМДИВ.

Введение

Способность гарантировать целостные и достоверные данные на сегодняшний день является важнейшим качеством вычислительных систем. Законодательство большинства развитых стран устанавливает принципы обеспечения безопасности критической компьютерной инфраструктуры, определяя необходимые мероприятия и ответственность субъектов. В РФ в 2017 году был принят Федеральный закон от 26.07.2017 № 187-ФЗ "О безопасности критической информационной инфраструктуры Российской Федерации" [1], в котором даны определения терминов и сформулированы основные понятия безопасности, принципы ее обеспечения, полномочия должностных лиц и субъектов критической инфраструктуры, а также дано категорирование объектов инфраструктуры по социальной, политической, экономической и прочим видам значимости.

Закон формулирует базовое понятие «компьютерный инцидент» как «факт нарушения и (или) прекращения функционирования объекта […], и (или) нарушения безопасности обрабатываемой таким объектом информации, в том числе произошедший в результате компьютерной атаки». Наряду со стоящей на переднем плане задачей обеспечения безопасности вычислительной инфраструктуры, обрабатывающей конфиденциальную и секретную информацию, не такой заметной, но от этого не менее важной, является задача обеспечения надежности встроенных систем управления.

Повсеместное использование встроенных компьютерных систем, вплоть до бытовых приборов, а также усложнение аппаратных и программных реализаций приводит к тому, что проблемы, связанные с обеспечением целостности и достоверности данных, выходят за рамки просто обеспечения кибербезопасности, ассоциируемой с компьютерными атаками, кибертерроризмом.

Не менее опасными для человека становятся и технические сбои, которым в первую очередь подвержены устройства хранения данных, особенно в суровых условиях эксплуатации под воздействием внешних факторов (высокие или низкие температуры, нестабильное напряжение питания, механические нагрузки, излучение и пр.).

Вопросы надежности, защищенности от сбоев, дополняют понятие кибербезопасности.

Использование поврежденных данных может, в лучшем случае, привести просто к неработоспособности управляемого объекта, в худшем же опасно изменить функционирование объекта, вплоть до угрозы жизни людей. В интернете легко найти сотни историй о катастрофических последствиях программных ошибок [2], но те же последствия могут наступить и в результате искажения кода программы или используемых ею данных.

Доверенная загрузка, гарантирующая использование целостных, не поврежденных умышленно или неумышленно данных, является базовым компонентом обеспечения кибербезопасности и надежности.

Доверенные вычисления и доверенная загрузка

Для обозначения нового понятия введен термин доверенного вычисления (Trusted Computing) [3]. Trusted Computing — это технология безопасности, при которой «компьютер будет вести себя ожидаемым образом, и это поведение будет обеспечиваться компьютерным оборудованием и программным обеспечением» [4]. Реализация этого поведения достигается путем загрузки программы в память оборудования с проверкой уникального ключа шифрования, недоступного для остальной части системы – доверенной загрузки. Доверенная загрузка – процесс загрузки компьютера, предотвращающий несанкционированный его запуск, несанкционированную загрузку операционной системы и получение доступа к конфиденциальной информации, а также предотвращающий загрузку искаженного кода и данных. Другими словами, доверенные вычисления – это комплекс аппаратных средств, программного обеспечения, средств и методов разработки, алгоритмов, обеспечивающих достоверность результатов вычислений (учитывая важность этого понятия, журнал Military & Aerospace Electronics переименовал периодический электронный бюллетень Cyber Security (кибербезопасность) на Trusted Computing (доверенные вычисления) [5]).

В соответствии с Информационным сообщением ФСТЭК России № 240/24/405 от 06.02.2014 г. «Об утверждении Требований к средствам доверенной загрузки» [6], а также Приказом ФСТЭК России № 119 от 27.09.2013 г., средства доверенной загрузки (СДЗ) подразделяются на три типа:

1) Средства доверенной загрузки уровня базовой системы ввода-вывода;

2) Средства доверенной загрузки уровня платы расширения;

3) Средства доверенной загрузки уровня загрузочной записи.

Первый тип – это программные средства, встроенные в микропроцессор, систему на кристалле (СнК), или в электронный модуль (BIOS и материнская плата в терминологии персональных компьютеров). Для этого типа СЗД не требуется установка дополнительных плат или чипов расширения.

СДЗ уровня платы расширения наиболее распространены и реализуются обычно программно-аппаратным способом. Наряду с функциями блокировки загрузки несанкционированного программного обеспечения и проверки целостности программно-аппаратной среды, эти средства позволяют протоколировать события безопасности. Широко применяются в вычислительных сетях.

Принцип действия СДЗ уровня загрузочной записи состоит в шифровании загрузочных секторов жестких дисков, в результате чего становится недоступной информация о логических разделах при несанкционированной загрузке компьютера.

Первичной, и поэтому основной, в доверенной загрузке является загрузка уровня базовой системы ввода-вывода, то есть загрузка внутреннего программного обеспечения (ВПО) – кода, выполняемого процессором после подачи питания, обеспечивающего инициализацию, начальное тестирование и, в итоге, загрузку операционной системы [7]. ВПО хранится в энергонезависимой памяти, как правило это флеш-память.

Считается, что первым вирусом уровня ВПО, широко распространенным по всему миру и нанесшим существенный вред, можно считать обнаруженный в 1998 году вирус CIH (Chernobyl) [8].

Его обнаружение и необходимость обеспечения верификации процесса загрузки привели к тому, что приблизительно в то же время компания IBM начала поставки материнских плат со встроенной микросхемой смарт-карты публичного ключа [9] шифрования.

Эта технология нашла широкое применение, в связи с чем возникла необходимость в определении стандартов.

В начале двухтысячных годов был учрежден консорциум Trusted Computing Group (TCG). TCG – консорциум, объединяющий сегодня более ста крупнейших производителей персональных компьютеров, микропроцессоров и программного обеспечения (среди них Intel, AMD, Hewlett-Packard, IBM, Microsoft). Консорциум TTG разработал и предложил спецификацию TPM (Trusted Platform Module - Модуль Доверенной Платформы) [10].

Спецификация определяет отраслевые стандарты доверенных вычислений в целом, в том числе и доверенной загрузки.

Для поддержки доверенной загрузки в данном подходе используется отдельная микросхема (рис. 1), выполняющая следующие функции:

- Аутентификация публичных ключей. Формирование пары ключей, используя внутренний генератор случайных чисел, создание цифровой подписи публичного ключа, верификацию, кодирование-декодирование.

- Проверка целостности. Вычисление и сохранение хеш-суммы по алгоритму SHA-1 (Secure Hash Algorithm 1) [11] каждого объекта при загрузке в отдельных регистрах PCR(i) (Platform Configuration Register) модуля TPM, в соответствии со спецификацией.

При этом PCR(n+1)=SHA1(PCR(n) + SHA1(объект)).

Регистры PCR хранят зашифрованную информацию о целостности метрик доверенной системы. Изменить их невозможно, можно лишь добавить новые записи, зависимые от предыдущих, что и показывает приведенная формула.

- Аттестация – сохранение хеш-сумм всех элементов загрузки и передачи данных вовне с возможностью подписи данных частным ключом.

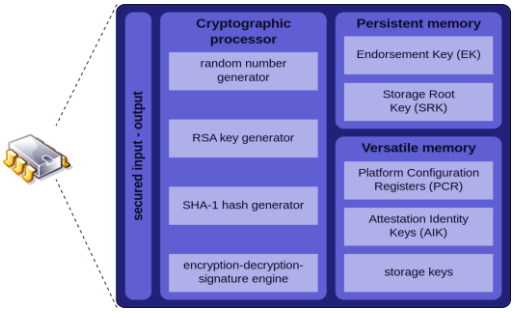


Рис.1. Схема TPM (Trusted Platform Module - Модуль Доверенной Платформы)

В состав микросхемы (рис. 1) входит блок безопасного (шифрованного) ввода-вывода (secured input-output) и криптографический процессор (Cryptographic processor), в задачу которого входит генерация случайных чисел и RSA-ключей [12], вычисление хеш-сумм, кодирование-декодирование шифрованной информации.

В постоянной памяти (Persistent memory) хранится ключ подтверждения и корневой ключ хранилища, а в отдельной области памяти (Versatile memory) хранятся временные ключи и конфигурационные записи в регистрах PCR.

Один из заслуживающих внимания подходов к организации доверенных вычислений состоит в том, что программа оперирует не оригинальными, а зашифрованными данными, так называемое «слепое» исполнение. Суть метода в том, что данные (шире – состояние вычислительного процесса) хранятся и, что самое главное, обрабатываются в виде, полученном гомоморфным преобразованием оригинальных данных. Обратное преобразование результатов доступно только владельцу приватного ключа.

Таким образом надежно блокируется возможность злонамеренного осмысленного искажения данных. И хотя при использовании гомоморфного шифрования эффективность исполнения программы значительно снизится [13], для таких программ, как загрузчик, это может играть непринципиальную роль.

Доверенная загрузка с использованием процессоров КОМДИВ

В ФГУ ФНЦ НИИСИ РАН проводятся исследования по разработке ВПО для обеспечения доверенной загрузки электронных модулей и ЭВМ на базе СнК с архитектурой КОМДИВ [14] (СнК КОМДИВ). Целью является:

- обеспечение загрузки только образов ПО, подписанных ключом разработчика;

- предотвращение загрузки образов ПО, целостность которых нарушена.

Для создания электронных модулей и ЭВМ, поддерживающих доверенную загрузку, используются системы на кристалле (СнК – СБИС, объединяющая на одном кристалле одно или несколько процессорных ядер, устройства и контроллеры внешних устройств, память) с ядром микропроцессора общего назначения с архитектурой КОМДИВ64 (800 МГц), в составе которой имеется кеш-память первого уровня инструкций и данных (32+16 Кбайт) и сбоеустойчивые контроллеры кеш-памяти второго уровня (встроенная в СнК КОМДИВ, объем 512 Кабйт) и внешней оперативной памяти. Этими средствами обеспечен сбоеустойчивый тракт данных, позволяющий в значительной степени снизить риск искажения данных, в том числе и кода программы, под воздействием внешних факторов. Сбоеустойчивость обеспечивается контролем передачи данных по коду Хэмминга с коррекцией одиночных ошибок [15]. В СнК КОМДИВ также входят контроллеры внешних устройств (Gigabit Ethernet, USB 2.0, CAN 2.0, SATA 3.0, PCI Express, SerRapidIO 4X/1X, SPI, I2C и пр.)

СнК КОМДИВ допускает использование двух источников загрузки – внутреннее или внешнее ПЗУ, выбор которого определяется внешним сигналом. Внешнее ПЗУ используется при отладке изделий или в применениях, не требующих доверенной загрузки.

Внутреннее ПЗУ (постоянное запоминающее устройство) объемом 32 Кбайта представляет собой классическое энергоне-

зависимое масочное ПЗУ [16], которое создается на производстве в процессе изготовления микропроцессора. Данные в ПЗУ также записываются на стадии производства и после этого технологически не могут быть перезаписаны в условиях производства. В процессе работы СнК КОМДИВ ПЗУ доступно только для чтения данных, таким образом несанкционированная запись в ПЗУ невозможна. А также, в отличие от отдельных микросхем ПЗУ, которые интегрируются на системную плату, невозможен подлог внутреннего ПЗУ, что гарантирует достоверность содержащихся в нем данных. Во внутреннее ПЗУ записывается начальный загрузчик, функционирующий от момента включения питания.

Для хранения ключей электронной цифровой подписи (ЭЦП - реквизит документа, позволяющий подтвердить принадлежность подписи ее владельцу, а также зафиксировать состояние данных, т.е. наличие либо отсутствие изменений, в электронном документе с момента его подписания) имеется однократно записываемая память (OTP – One Time Programmable) объемом 1 Кбайт. Память XPM™ (Extra Permanent Memory) от Kilopass Techology [17] построена на стандартной КМОП-логике. OTP в незапрограммированном виде содержит логические «0». Физический принцип процесса записи в ячейку памяти логической «1» основан на пробое подзатворного окисла транзистора высоким напряжением, получаемым схемой накачки напряжения или отдельной линией, выведенной на высокое напряжение. Механизм NVM™ (non-volatile memory) гарантирует полную защищенность данных.

Результаты и перспективы

На базе макетных образцов СнК КОМДИВ в ФГУ ФНЦ НИИСИ РАН реализована процедура доверенной загрузки. Технология доверенной загрузки состоит в следующем.

После включения питания или подачи сигнала Сброс, управление передается во внутреннее ПЗУ предзагрузчику, выполняющему следующие функции:

- минимальная инициализация аппаратуры СнК КОМДИВ (процессор, память);
- считывание из внешнего источника и разбор заголовка контейнера, содержащего основной загрузчик;
- загрузку контейнера в память;
- вычисление значение хэш-функции от ключа проверки ЭЦП;
- при совпадении (значение храниться в памяти OTP) распаковку контейнера и передачу управления загрузчику.

Отработанная на макетах технология показывает работоспособность выбранного подхода и открытость к расширению.

Дальнейшее развитие средств доверенной загрузки в СнК КОМДИВ в работах НИИСИ РАН включает реализацию следующих возможностей:

- контроль версий для обновления программного обеспечения, предусматривающий запись (специальными средствами) дополнительной информации в OTP-память о доверенных версиях программного обеспечения;
- различные режимы старта (блокировка запуска неподписанных ЭЦП образов, блокировка записи в память OTP, выбор источника загрузки и пр.). Эта возможность необходима для построения вычислительных средств на базе СнК КОМДИВ различного назначения и конфигурации.

Заключение

В СнК КОМДИВ реализованы основные возможности СДЗ уровня базовой системы ввода-вывода, необходимые для построения доверенной вычислительной системы. Развитие этих возможностей расширяет сферу применения СнК КОМДИВ в разработках на основе отечественной элементной базы.

Firmware level trusted bootloading KOMDIV-architecture system-on-chip

K.A.Pshenichny, S.A.Sidorov

Abstract: Described firmware level trusted bootloading process in KOMDIV-architecture system-on-chip.

Keywords: firmware, trusted bootloading, KOMDIV-architecture, system-on-chip

Литература

[1] Федеральный закон от 26 июля 2017 г. № 187-ФЗ «О безопасности критической информационной инфраструктуры Российской Федерации» // http: // ivo.garant.ru/ #/ document/ 71730198/ paragraph/1:0

[2] https://habr.com/company/mailru/blog/370153

[3] Eimear Gallery, Chris J. Mitchell. Trusted Computing: Security and Applications // Cryptologia. — Taylor & Francis, 2008.

[4] Chris Mitchell. Trusted Computing. IET, 2005. ISBN 978-0-86341-525-8.

[5] www.militaryaerospace.com

[6] ФСТЭК. Информационное сообщение об утверждении требования к средствам доверенной загрузки от 06 февраля 2014 г. № 240/24/405. https://fstec.ru/component/attachments/download/663

[7] Е.Н.Лякина, С.А.Сидоров. Внутреннее программное обеспечение. // Информационная безопасность. Микропроцессоры. Отладка сложных систем. Сб. статей под ред. Акад. РАН В.Б.Бетелина. М.: НИИСИ РАН, 2004, с.55-75 (21 стр.). ISBN 5-93836-019-7.

[10] Trusted Computing Group, TPM Основные Спецификации https://trustedcomputinggroup.org/resource/tpm-library-specification

[9] https://www.emaro-ssl.ru/blog/public-and-private-key

[8] W95.CIH Technical Details. Symantec. 25 April 2002. http://www.symantec.com/security_response/writeup.jsp?docid=2000-122010-2655-99

[11] https://tools.ietf.org/html/rfc3174

[12]A.J. Menezes, P.V.Oorschot, S.A.Vanstone. Handbook of Applied Cryptography — CRC Press, 1996. — 816 p. — (Discrete Mathematics and Its Applications) — ISBN 978-0-8493-8523-0

[13] V.Sidorov and W.K.Ng, "Towards Performance Evaluation of Oblivious Data Processing Emulated with Partially Homomorphic Encryption Schemes," 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), New York, NY, 2016, pp.113-115. doi: 10.1109/BigDataSecurity-HPSC-IDS.2016.36

[14] С.Г.Бобков. Импортозамещение элементной базы вычислительных систем. //Вестник Российской академии наук, 2014, том 84, № 11, с.1010-1016. ISSN: 0869-5873.

[15] У.Питерсон, Э.Уэлдон. Коды, исправляющие ошибки: Пер. с англ. М.: Мир, 1976, 600 с.

[16] Ю.В.Новиков. «Основы цифровой схемотехники» -М.: «Мир», 2001.

[17] https://www.techonline.com/directory/kilopass-technology

Исследование влияния накопленной дозы ионизирующего излучения на характеристики высокотемпературных HVLDMOS транзисторов.

С.И.Бабкин[1], С.И.Волков[2], А.С.Новоселов[3], С.В.Румянцев[4]

ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1]Sergey_Babkin@srisa.ru, [2]Sviatoslav_Volkov@srisa.ru,

*[3]Anton_Novoselov@srisa.ru, [4]*Sergey_Rumyancev@srisa.ru

Аннотация: Исследованы характеристики высокотемпературных высоковольтных LDMOS транзисторов, изготовленных на КНИ структуре, в диапазоне накопленной дозы ионизирующего излучения до 600 кРад.. Показано, что транзисторы сохраняют свою работоспособность при максимальной дозе в 600кРад. Напряжение пробоя n-канальных LDMOS транзисторов увеличивается с набором дозы от 38 до 43В. Сдвиг порогового напряжения n-канальных LDMOS транзисторов составляет около 50мВ на 100 кРад, p-канальных LDMOS транзисторов составляет около 100 мВ на 100 кРад. Пороговое напряжение донного (паразитного) n-канального LDMOS транзистора составило 10В при максимальной дозе в 600 кРад.

Ключевые слова: LDMOS, КНИ, высоковольтный транзистор, высокая температура, накопленная доза

1. Введение

В настоящее время в автомобильной электронике, системах промышленного контроля, телекоммуникациях и космической электронике широкое применение находят интеллектуальные монолитные интегральные схемы (ИМИС). ИМИС состоят из низковольтных и высоковольтных МОП транзисторов. Последние изготавливают с использованием технологии High Voltage LDMOS далее HV LDMOS. Такие ИМИС повышают надежность, уменьшают объем и вес, а также увеличивают эффективность системы в целом [1, 2]. Нередко к HV LDMOS предъявляется требование сохранения работоспособности в экстремальных условиях (ионизирующее излучение, высокие температуры (>125 °C)).

Формирование HV LDMOS транзисторов сочетается с традиционной КМОП КНИ технологией [3]. Эти преимущества позволяют использовать монолитную интеграцию нескольких силовых устройств, низковольтных схем управления и процессорных ядер на одном чипе. Схемы, использующие КНИ технологию, обладают повышенной стойкостью к ионизирующему облучению и к воздействию нейтронов высокой энергии и тяжелых заряженных частиц [4,5]. В ФГУ ФНЦ НИИСИ РАН разрабатывается технология формирования высоковольтных LDMOS КНИ транзисторов на основе базового КМОП КНИ процесса с проектными нормами 0,5 мкм. В работе [6] рассмотрены особенности технологии формирования высоковольтных LDMOS транзисторов, исследованы ВАХ различных конструктивно-технологических вариантов LDMOS транзисторов.

Исследована возможность использования высоковольтных LDMOS транзисторов в условиях высоких температур (до 225°C).

Сравнение характеристик высоковольтных LDMOS транзисторов с характеристиками низковольтных КМОП КНИ транзисторов, изготовленных в едином технологическом цикле, проведено в работе [7].

В данной работе ставилась задача исследовать влияние ионизирующего излучения на характеристики n (NLDMOS) и p (PLDMOS) канальных HVLDMOS транзисторов в диапазоне дозы облучения до 600 кРад.

2. Тестовые образцы и методы их исследования

На рис. 1 представлена конструкция тестовых высоковольтных LDMOS транзисторов, сформированных на структуре КНИ со следующими параметрами: толщина рабочего слоя кремния 190 нм; толщина слоя изолирующего окисла 150 нм, толщина подзатворного окисла 14 нм. Для испытаний выбраны n- и p-канальные LDMOS транзисторы со следующими параметрами: длина канала 0,5 мкм; ширина канала 5 мкм;

длина области толстого полевого окисла F=6 мкм; длина протяженного стока DRIFT C = 6 мкм; величина перекрытия затвором (GATE) области толстого окисла (F) H = 3 мкм.

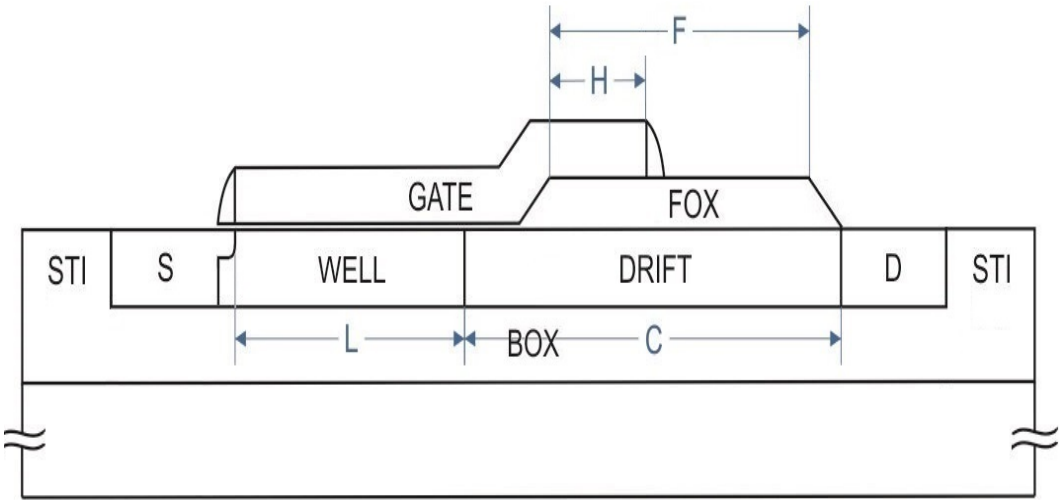


Рисунок 1. Конструкция исследуемого LDMOS КНИ транзистора.

Таблица 1.

Характеристики аппаратно-программного комплекса.

Измеряемые параметры	Диапазон	Разрешение
Температура	от - 60 до +300 °C	0,5°C
Напряжение	от -100 до + 100 В;	0,5 мкВ
Ток	от-100 мА до+100 мА	1 фА
Частота	от 1 кГц до 5 МГц	1 мГц
Емкость	от 1 фФ	0,1 фФ
Макс. мощность на канал	2 Вт	100мА при V<20В

Облучение транзисторов проводилось на установке ГУТ-200М РНЦ КИ. Набор дозы проходил в пять этапов, 4 этапа по 100 кРад, 5-й – 200 кРад. После каждого этапа облучения проводились измерения характеристик LDMOS транзисторов.

Для измерения характеристик транзисторов после воздействия ионизирующего излучения использовался автоматизированный аппаратно-программный комплекс. Аппаратная часть комплекса состоит из трех подсистем: измерительной – на основе оборудования фирмы Keysight Technologies (параметрический анализатор B1500A, матричный коммутатор E5250, осциллограф DSO6104L); системы контактирования – на основе зондовой станции фирмы Suss Microtech; температурной – на основе термосистемы фирмы ATT. Для интеграции комплекса, генерации и исполнения команд и обеспечения хранения результатов измерений используется управляющий компьютер. Управление комплексом, создание измерительных программ производится с помощью программного обеспечения Easy Expert и среды программирования VEE фирмы Keysight Technologies.

Возможности созданного аппаратно-программного комплекса и диапазон измеряемых параметров представлены в таблице 1.

3. Основные результаты

Влияние накопленной дозы оценивалось по изменению следующих характеристик LDMOS транзисторов:

1) пороговое напряжение транзистора (определялось по максимуму проводимости при Vds=0,1В, Vgs в диапазоне от 0В до 3,3В (шаг 0,05В)

2) пороговое напряжение донного (паразитного) транзистора (определялось по току стока в 1 мкА при Vds=3,3В, Vgs=0В)

3) напряжение пробоя сток-исток (определялось по току стока в 1 нА при Vgs=0В).

Зависимости указанных параметров от накопленной дозы представлены на рисунках 2,3,4

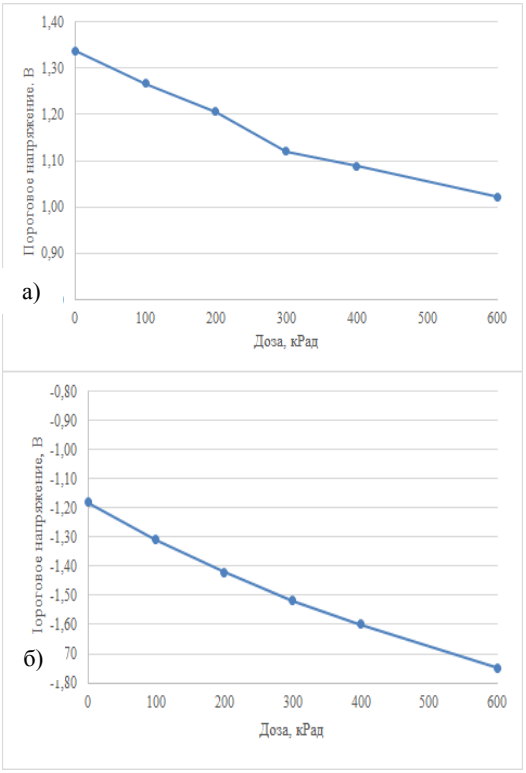


Рисунок 2. Зависимость порогового напряжения NLDMOS- а и PLDMOS- б транзисторов от накопленной дозы.

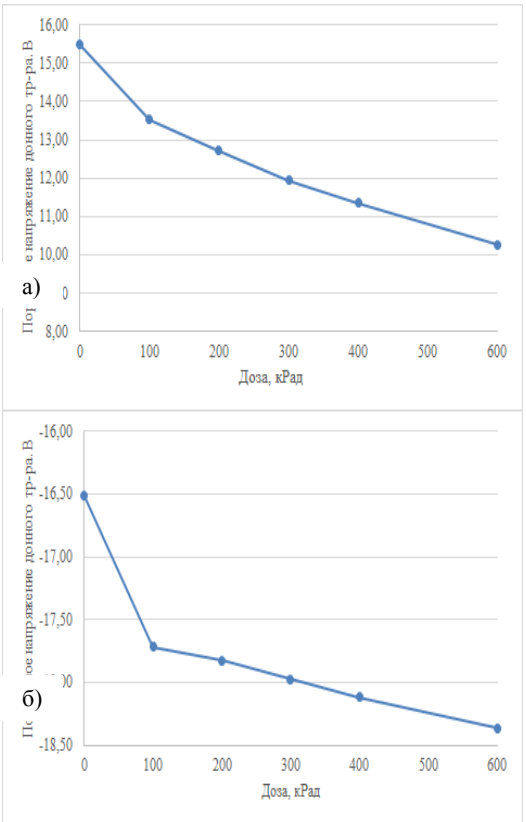


Рисунок 3. Зависимость порогового напряжения донного транзистора для NLDMOS - а и PLDMOS – б транзисторов от накопленной дозы.

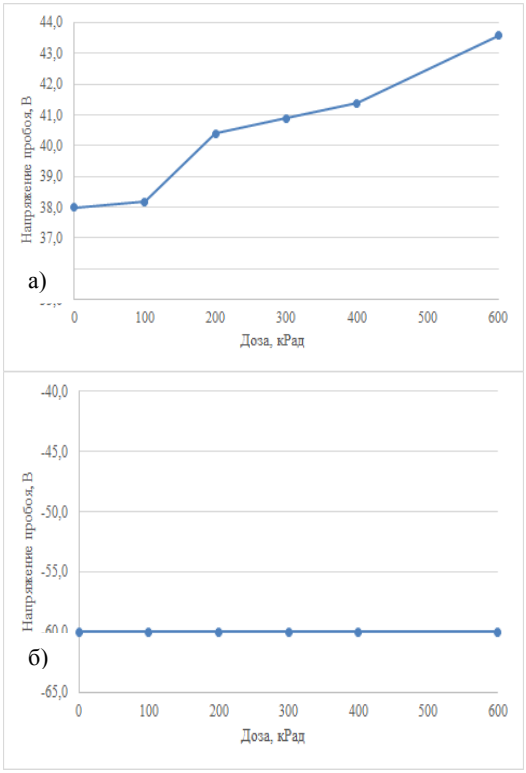


Рисунок 4. Зависимость напряжения пробоя сток-исток NLDMOS - а и PLDMOS – б транзисторов от накопленной дозы.

Из представленных данных следует, что как n-канальные, так и p-канальные LDMOS транзисторы работоспособны до накопленной дозы в 600 кРад. Пороговое напряжение n-канального транзистора уменьшилось на 300 мВ до 1 В, пороговое напряжение p-канального транзистора увеличилось на 600 мВ до 1,7 В. Порог открытия донного транзистора уменьшился с 15 В до 10 В для NLDMOS транзистора, и увеличился с 16,5 В до 18,5 В для PLDMOS транзистора. С набором дозы напряжение пробоя сток-исток n-канальных LDMOS транзисторов увеличивается от 38 до 43 В и остается более 60 В для p-канальных LDMOS транзисторов.

Заключение

Высокотемпературные HVLDMOS транзисторы сохраняют работоспособность в исследуемом диапазоне накопленной дозы.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН (выполнение фундаментальных научных исследований ГП 14) по теме № 0065-2018-0117 "Исследование и построение моделей и конструкций элементов микроэлектроники в расширенном диапазоне температур (-60С 300С). "(№ АААА-А18-118042790079-0).

Study of the effect of accumulated ionizing radiation dose on the characteristics of high-temperature HVLDMOS transistors

S.I. Babkin, S.I. Volkov, A.S. Novosyolov, S.V. Rumyancev

Abstract: The characteristics of high-temperature high-voltage LDMOS transistors manufactured on a SOI structure are investigated in the range of accumulated ionizing radiation dose up to 600 kRad. It is shown that transistors maintain their operation at the maximum dose of 600 kRad. The breakdown voltage of n-channel LDMOS transistors increases with a set of doses from 38 to 43V. The shift of the threshold voltage of n-channel LDMOS transistors is 50mV per 100 kRad, p-channel LDMOS transistors is 100 mV per 100 kRad. The threshold voltage of the backgate (parasitic) n-channel LDMOS transistor was 10V with a maximum dose of 600 kRad.

Keywords: LDMOS, SOI, CMOS, high-voltage transistor, high temperature, total dose

Литература

1. B. Murari, F. Bertotti, G.A. Vignola, Smart Power ICs, Springer, Germany, 1996.
2. A. Nakagawa, Recent Advances in High Voltage SOI Technology for Motor Control and Automotive Application, Proceedings of International on Bipolar/BiCMOS Circuits and Technology Meeting, 1996, pp. 69 – 72.
3. G..Toulon, I,Cortes, F.Morancho . Analysis and Optimization of LUDMOS Transistors on a 0.18 um SOI CMOS Technology. International Journal of Microelectronics and computer science, Vol.1, No.1, 2010.
4. J.P. Colinge, Thin-film SOI technology: the solution to many submicron CMOS problems, in: Electron Devices Meeting, 1989. IEDM'89. Technical Digest., International, IEEE, 1989, pp. 817 - 820.
5. J.R. Schwank, V. Ferlet-Cavrois, M.R. Shaneyfelt, P. Paillet, P.E. Dodd, Radiation effects in SOI technologies, IEEE Trans. Nucl. Sci. 50 (3) (2003) 522-538.
6. С.И.Бабкин, Д.А.Байдаков, С.И.Волков, А.А.Глушко, С.А.Морозов, А.С.Новоселов, А.А.Столяров. Разработка технологии формирования высоковольтных LDMOS КНИ транзисторов для экстремальной электроники // Труды НИИСИ РАН, 2018, т. 8, № 3, с. 31-36.
7. С.И.Бабкин, С.И.Волков, С.А.Морозов, А.С.Новосёлов, С.В.Румянцев. Исследование параметров высоковольтных LDMOS транзисторов при высоких температурах // Труды НИИСИ РАН, 2018, т. 8, № 3, с. 25-30

Построение отказоустойчивых систем для проведения олимпиад по программированию

Н.О.Бесшапошников[1], А.Г.Кушниренко[2], А.Г.Леонов[3], К.А.Прокин[4]

[1,2,3,4] ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

[3] ФГБОУ ВО «Московский государственный университет имени М.В. Ломоносова: механико-математический факультет», Москва, Россия и ФГБОУ ВО «Московский педагогический государственный университет», Москва, Россия, E-mail: dr.l@vip.niisi.ru

Аннотация: В настоящее время большое внимание уделяется цифровизации образования и, в частности, цифровой школе. В рамках этого движения существенная часть образовательного процесса трансформируется в электронную форму, с возможностью использования дистанционного доступа к цифровым образовательным ресурсам. Управление образовательным процессом, в частности, контрольные опросы и олимпиады также проводятся с использованием цифровой среды, в автоматическом режиме. Эта статья посвящена некоторым вопросам построения надежной отказоустойчивой базовой цифровой системы для проведения олимпиад. Рассмотрены известные решения в этой области. Предложен набор компонент и основные алгоритмы функционирования открытой отказоустойчивой системы. Рассмотрены вопросы масштабируемости и устойчивости к высоким нагрузкам.

Ключевые слова: Программирование, фреймворк, кластер, олимпиада.

Потребность в автоматизированных цифровых системах для проверки достижений, учащихся в образовательном процессе сейчас высока как никогда [8].

Автоматизированные практикумы, включающие online цифровой контроль разрабатывались, как для высшей школы [9],[10],[11], так и для школы и дошкольных образовательных учреждений. Так система ПктоМир [7], созданная специалистами ФГУ ФНЦ НИИСИ РАН создавалась как цифровая среда для обучения младшеклассников и дошкольников с возможностью автоматизированного контроля достижений и проведение олимпиад.

В мире также существует порядка десятка систем проведения олимпиад и соревнований по программированию для старших школьников, студентов, профессиональных программистов в том числе и для соревнований командных [5],[4].

Рутинная техническая задача, решаемая подобными системами - ведение базы данных, хранящих всю информацию о конкретном соревновании. Основная содержательная задача - проверка правильности программ, с составляемых участниками соревнований. Вспомогательная задача – ранжирование участников соревнований, начисление баллов по той или иной системе.

Недостатки существующих систем

В описании известной российской системы ejudge [1] декларируется, что ejudge - это система для проведения различных мероприятий, в которых необходима автоматическая проверка программ. Ее недостатками являются:

- нестабильная работа;
- сложность настройки;
- недружественный интерфейс;
- отсутствие масштабирования.

Moodle [2] – свободно распространяя система обучения. При всех ее достоинствах, к ее недостаткам можно отнести:

- сложность системы;
- требование специальных навыков настройки от преподавателя;
- настройка и установка модулей для тестирования.

Ниже рассмотрен проект отказоустойчивой системы для проведения олимпиад.

Система для проведения командных соревнований и олимпиад состоит из нескольких компонент:

1. Система аутентификации пользователей;
2. Система проверки решений учеников;
3. API сервер для выдачи и приема решений задач;

4. WEB-сервер для создания, редактирования индивидуальных заданий и комплектов задания для олимпиад, отслеживания и хранения результатов выполнения олимпиадных заданий.

Системы аутентификации, проверки, API и WEB сервера реализованы в Node.JS [6] с помощью фреймворков Express + Pug. В качестве базы данных используется NoSQL БД MongoDB[3].

Система устроена таким образом, чтобы её разворачивание собственными силами было максимально простым.

Реализация системы проверки решений

Система проверки решений является автономной и легко масштабируемой, как вертикально, так и горизонтально.

Это обусловлено выбором технологий и программных продуктов, поддерживающих масштабируемость и отказоустойчивость.

Система проверки решений состоит из двух основных частей.

Часть I, реализованная на Node.JS[13], заключается в обработке запроса на проверку. При этом обеспечивается:

a. Связь с БД MongoDB, где хранится информация о тестируемом решении и самом задании для тестирования;

b. Получение сообщения из очереди проверки с помощью библиотеки BusMQ (модуль Node.JS), основанной на Redis [12];

c. Подготовка файла, необходимого для проверки (система компилирует или дополняет решение, генерирует или использует входные\выходные данные);

d. Сбор и тестирование данных после выполнения программы решения.

В базе данных MongoDB хранится коллекции, содержащие данные о задании, которое нужно проверять, настройки языков программирования (и соотв. им компиляторов) и проверяющих систем, и сами решения.

Задание содержит информацию:

1. Доступные языки программирования для сдачи решения и ключи компиляции для соответствующих им компиляторов (если такой есть);

2. Ограничения по времени исполнения и памяти;

3. Тесты, которые, в свою очередь, содержат:

- файлы для тестирования и соответствующие им эталонные результаты выполнения (включая стандартные потоки ввода/вывода);
- или программы (могут быть реализованы на любом доступном языке), которые генерируют входные данные и тестируют выходные;

4. Настройки проверки тестов (сравнение побайтово, игнорируя пробельные символы, или программа на JavaScript для собственной проверки);

5. Дополнительные файлы для компиляции (компилируются вместе с решением).

Решение содержит:

1. Файл с программой (может быть архив) для тестирования;

2. Статус тестирования (загружен/тестирование/сдано/ошибка);

3. Отметку времени о загрузке и тестирования;

4. Результат тестирования, содержащий файлы, полученные программой из решения (логи компиляции, информация о пройдённых и не пройденных тестах).

Все файлы, необходимые для работы системы, хранятся в GridFS – модулем MongoDB для работы с файлами.

Этот модуль позволяет эффективно работать с файлами, а также использовать стандартные механизмы базы данных для обеспечения отказоустойчивости и масштабирования.

Очередь BusMQ обладает гарантированной доставкой и поддерживает кластеризованный вариант Redis сервера - Redis Sentinel.

Помимо этого, Redis Sentinel в случае отказа любой из систем сохраняет информацию об очереди, что гарантирует отсутствие потерь сообщений о тестировании.

Часть II является *исполняющей частью*, основанной на Docker [14].

В исполняющей части в Docker-контейнерах происходит компиляция и запуск программ решений.

Выбор Docker’а обусловлен кроссплатформенностью, а также меньшими затратами на виртуализацию для создания изолированных сред проверки одного конкретного задания.

Контейнеры для тестирования собраны на базе образов CentOS 7 Linux, содержат пакеты и специально подготовленные скрипты для тестирования решений на различных языках программирования.

Процесс проверки задания ученика описан на диаграмме ниже:

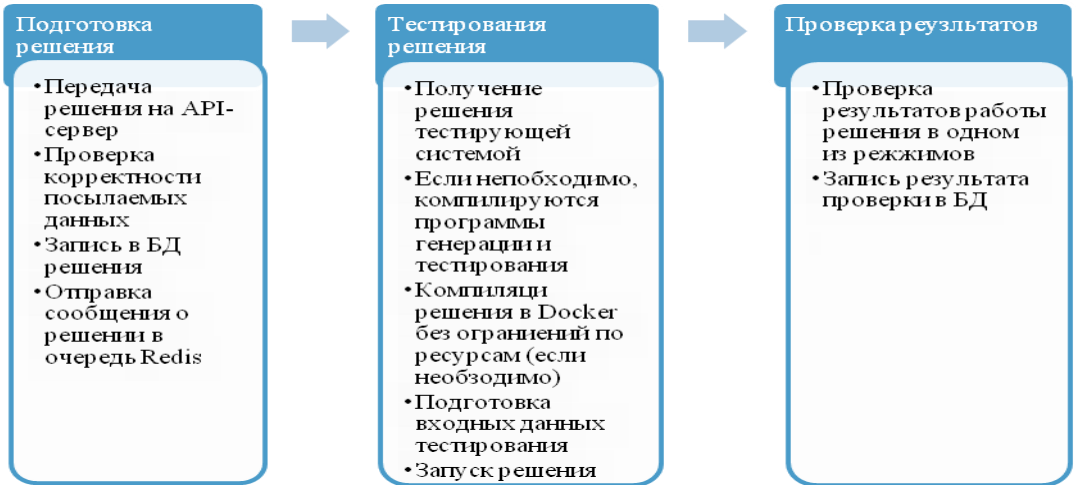


Рис.1. Полный процесс проверки решений

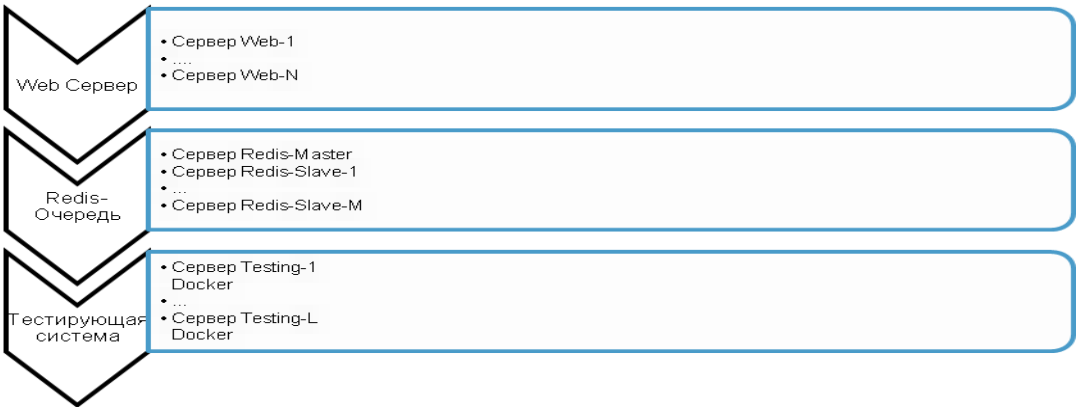


Рис.2. Процесс проверки решений с точки зрения серверов

Масштабируемость и отказоустойчивость данной тестирующей системы обеспечена:

1. Stateless архитектурой web-компоненты. При повышении нагрузки, достаточно создания новой копии одного и того же эталонного сервера и, например, балансировать с помощью прокси-серверов (nginx, Apache) или DNS. Недоступность одного из серверов не приведёт к недоступности всей компоненты.
2. Автономностью тестирующих серверов. Каждый такой сервер подключается к очереди сообщений, и получает задания на тестирование из неё. Если сервер достигает максимальной нагрузки (определяется конфигурацией сервера, а именно максимально единовременно тестируемых решений), то он пропускает сообщения из очереди, их получает другой. Если не один сервер не смог обработать сообщение, то оно будет храниться в очереди, в ожидании, пока оно не будет обработано.
3. Масштабируемостью и отказоустойчивостью Redis серверов за счёт технологии Redis Sentinel. Он подразумевает под собой наличие одного master сервера, который обрабатывает текущие запросы и несколько slave серверов, которые содержат копию текущих данных. При этом каждый тестирующий сервер может содержать сервис redis-sentinel, который будет выступать в качестве арбитра, если возникает вопрос выбора нового redis-master сервера.

Работа выполнена по теме 0065-2018-0017 госзадания 2018 года в отделе учебной информатики ФГУ ФНЦ НИИСИ РАН.

Building fault-tolerant systems for programming contests

N.O.Beshaposhnikov, A.G.Kushnirenko, A.G.Leonov, K.A.Prokin

Abstract: Much attention is paid to the digitalization of education and, in particular, the digital school now. As part of this movement, an essential part of the educational process is being transformed into an electronic form, with the possibility of using remote access to digital educational resources. Management of the educational process, in particular, contests and olympiads are also conducted using the digital environment, in automatic mode. This article is devoted to some questions of building a reliable, fault-tolerant basic digital system for conducting competitions. Authors are considered known solutions in this area. A set of components and the basic algorithms of functioning of an open fault-tolerant system are proposed. The issues of scalability and resistance to high loads are including in this article.

Keywords: programming, framework, cluster, contest.

Литература

1. Ejudge – система для проведения различных мероприятий, в которых необходима автоматическая проверка программ. [Электронный ресурс] / Режим доступа: https://ejudge.ru/wiki/index.php/Система_ejudge свободный.
2. Moodle – система управления курсами (электронное обучение). [Электронный ресурс] / Режим доступа: https://moodle.org свободный.
3. База данных с открытым кодом MongoDB. NoSQL БД MongoDB. [Электронный ресурс] / Режим доступа: https://www.mongodb.com/ свободный.
4. Т.И.Ведерникова, К.Б.Чепченко. Система проведения соревнований и проверки решений задач по программированию. «BAIKAL RESEARCH JOURNAL», т. 6 (2015), №5, 19. [Электронный ресурс] / Режим доступа: https://cyberleninka.ru/article/v/sistema-provedeniya-sorevnovaniy-i-proverki-resheniy-zadach-po-programmirovaniyu свободный.
5. Википедия. Олимпиады по программированию. [Электронный ресурс] / Режим доступа: https://ru.wikipedia.org/wiki/Олимпиады_по_программированию свободный.
6. Википедия. SpiderMonkey — первый в истории «движок» JavaScript. [Электронный ресурс] / Режим доступа: https://ru.wikipedia.org/wiki/SpiderMonkey свободный.
7. А.Г.Кушниренко, А.Г.Леонов, М.В.Райко, И.Б.Рогожкина. Методические указания по проведению цикла занятий «Алгоритмика» в подготовительных группах дошкольных образовательных учреждений с использованием свободно распространяемой учебной среды ПиктоМир. [Электронный ресурс] / Режим доступа: https://www.niisi.ru/piktomir/m2018.pdf свободный.
8. А.Г.Леонов, Ю.А.Первин. Качественные оценки эффективности методики обучения элементам информатики в пропедевтическом курсе. «Ярославский педагогический вестник», (2015), № 5.
9. А.Г.Леонов, А.А. Прилипко. Автоматизированный практикум по машинным языкам – основа изучения языков программирования. «Сборник научных статей: Интеграция отечественной науки в мировую: структурные преобразования и перспективные направления развития – Санкт-Петербург 2016 г.», СПб., (2016).
10. А.Г.Леонов, А.А. Прилипко. Разработка и внедрение компьютерных практикумов в учебные курсы программирования в школе и вузе. «Сборник научных статей по итогам международной научно-практической конференции – Санкт-Петербург 2015 г.», СПб., 2015, 116 – 120.
11. А.Г.Леонов. ЭВМ-Практикум. «Первое сентября», (2011), № 11.
12. Резидентная в памяти база данных Redis с открытым кодом. [Электронный ресурс] / Режим доступа: https://redis.io/ свободный.
13. Свободно распространяемая система Node.js серверной поддержки исполнения Java Script кодов. [Электронный ресурс] / Режим доступа: https://nodejs.org/ свободный.
14. Средство виртуализации Docker. [Электронный ресурс] / Режим доступа: свободный.

Разработка программных исполнителей системы ПиктоМир

Н.О.Бесшапошников, К.А. Мащенко, А.Е. Орловский

ФГУ ФНЦ НИИСИ РАН, Москва, Россия, E-mail: dr.l@vip.niisi.ru

Аннотация: Школьный и пропедевтический курсы по информатике, использующие соответственно школьный алгоритмический язык и язык Пикто базируются на педагогических программных средствах КуМир и ПиктоМир. Ученики изучают предмет, создавая алгоритмы для управления широким спектром программных исполнителей и роботов. При этом каждый программный исполнитель системы КуМир фактически представляют собой отдельный программный модуль, методы подключения которого к КуМир фиксированы и описаны в открытых источниках. Таким образом, педагоги-инноваторы могут самостоятельно проектировать и создавать новые программные исполнители к системе КуМир и использовать их в своей педагогической практике.

В статье описан аналогичный подход к созданию программных исполнителей в системе ПиктоМир. Приведены унифицированные методы подключения вновь создаваемых программных исполнителей и роботов.

Ключевые слова: программные исполнители, модульность, ПиктоМир, КуМир, алгоритмика.

Учебники и учебные пособия, который используют для составления алгоритмов придуманный академиком А.П.Ершовым алгоритмический язык [5][6], предполагают использование на уроках Информатики и ИКТ учебной программной системы КуМир [10]. Программные исполнители системы КуМир, такие как Водолей, Черепашка и др.[12] разработаны и интегрированы в КуМир с использованием единой технологии и единым программным интерфейсом. Такой подход позволяет педагогам или программистам, обладающим опытом работы на C++ в среде Qt [2], самостоятельно проектировать и создавать программных исполнителей в системе КуМир. Методически, эти исполнители должны поддерживать два режима работы, пультовый [9], и программный. Выполнение этого требования позволяет ученику знакомиться с системой команд нового программного исполнителя или робота, не используя на первых порах ни языка, ни системы программирования. Все команды, выполненные в пультовом режиме для решения поставленной перед ребенком алгоритмической задачи, сохраняется в виде протокола и могут быть потом помещены в качестве готового линейного фрагмента в Кумир-программу [7].

В учебной бестекстовой среде программирования ПиктоМир [11], используемой в пропедевтических курсах по алгоритмике для дошкольников и младших школьников [3], также используется принцип программного управления исполнителями (роботами) [8]. Система ПиктоМир с успехом применяется для детей, которые еще не умеют или не очень любят читать и писать. Для дошкольников и младших школьников в продолжительных циклах занятий целесообразно использовать обширный набор различных исполнителей и роботов. Продолжительная работа с одним исполнителем снижает мотивацию детей. Хотя в настоящее время ПиктоМир поддерживает пять типов подвижных роботов и два типа стационарных исполнителей, некоторым педагогам, проектирующим свои курсы, может оказаться полезна возможность создавать своих исполнителей (роботов). Скажем для развития межпредметных компетенций. (Например, педагог может захотеть разработать исполнитель Вычислитель для наглядного представления основных арифметических действий). Разработчику нового программного исполнителя приходится решать две задачи.

Во-первых, необходимо реализовать и визуализировать функциональность нового исполнителя автономно. Для этого разработчику нужно обладать навыками программирования с использование широко распространенной свободно распространяемой системы программирования cocos2d-x [1]. Документация по этой системе доступна в интернете.

Во-вторых, отлаженный автономно программный модуль требуется подключить к ПиктоМиру. Для решения этой задачи разработан специальный интерфейс, позволяющий провести интеграцию нового исполнителя в ПиктоМир, не вдаваясь в детали реализации ядра ПиктоМира.

Этот интерфейс в общих чертах описан ниже. Для желающих оценить трудозатраты на разработку нового исполнителя и интеграцию его в ПиктоМир по описанной ниже методике, на сайте www.piktomir.ru более детально описано подключение к ПиктоМи-

ру простейшего программного исполнителя «Робот».

Система ПиктоМир имеет четко выраженную модульную архитектуру.

В данном случае модуль - это набор исходных файлов на языке JavaScript и ресурсов, делящих на следующие три категории:

1. Файлы реализации исполнителя и обстановки.

В них содержатся классы, наследники базовых классов из ядра системы ПиктоМир (например, **AbstractRobot**, **MapLayer** и т.д.) [4], реализующие необходимый для работы API.

2. Ресурсы - наборы картинок (в формате SpriteSheet) и анимаций, используемых для визуализации исполнителей и обстановки.

3. Файл описания модуля, в котором содержится наследник класса **pm.ModuleInfo**, в котором указано, как именуются функции, их назначение и спецификация возвращаемых данных.

Набор функций-методов у всех модулей одинаковый.

В классе модуля также должны содержаться все уникальные идентификаторы исполнипелей, обстановок, объектов и заданий.

Описание класса (модуля) и используемых констант заканчивается вызовом специфичной функции для регистрации модуля:

```
pm.moduleUtils.register(
      new <название модуля>
)
```

pm.moduleUtils - это объект, отвечающий за регистрацию и использование модулей внутри ядра системы ПиктоМир. В нем содержится словарь **_modules**, который содержит соответствия между идентификатором модуля и объектом: реализующим модуль. Метод **register** данного класса в качестве параметра получает объект регистрируемого модуля, получает его идентификатор (далее – тип модуля) и заносит в словарь.

Любая функция из **pm.moduleUtils** содержит первым параметром тип модуля, поэтому в классах обстановки и исполнителя, с помощью функции getType возвращается непосредственно тип модуля:

```
pm.moduleUtils.function(
      object.getType(),
      <other parameters>
)
```

Любая функция из **pm.moduleUtils** использует функцию, которая лежит в соответствующем модуле, называются они для удобства так же, как и в самих модулях:

```
return
this._modules[type].function(
      <other parameters>
)
```

Таким образом все файлы, особенности, особенные функции исполнителя сами встраиваются в нужные места проекта, в которых вызываются функции именно того исполнителя, с которым в данный момент идет работа.

Ниже рассматривается пример создания и подключения к системе нового исполнителя, называемого здесь Робот.

Сначала создается класс модуля, который является наследником **pm.ModuleInfo**, содержащим и описывающим базовый функционал модуля

```
pm.RobotLevelModule=
pm.ModuleInfo.extend({
```

Пусть имя модуля будет таким:

```
name: «Robot»;
```

Дальше идет описание важной функции, возвращающей уникальный идентификатор модуля (тип), которая необходим для регистрации модуля:

```
getType:function(){
 return pm.RobotLevelModule.Type
}
```

Следующая функция генерирует пустую карту (обстановку: в которой перемещается робот) в редакторе:

```
generateEmptyMap(level, robot),
```

в качестве параметров которой выступаю объект уровня, для которого генерируется карта, и робот, который на ней будет расположен. Внутри функции создается новая карта:

```
    map          =          new
pm.data.RobotMap();
```

указывается ее размер и формируется через свойство **originalType** необходимое наполнение карты, а также позиция Робота

свойство **startForRobot** с указанием начального положения - свойство **startRobotData.direction**. Функция возвращает созданную карту.

Необходимо имплементировать функцию, которая возвращает пустой уровень:

```
generateEmptyLevel:function (){
var level=
        new pm.data.RobotLevel();
},
```

которая создает уровень; настраивает его, создает Робота и записывает в информацию уровня, вызывает **generateEmptyMap** от этого уровня и робота, создает список заданий и кладет туда задания этого исполнителя, возвращает уровень.

Далее следует набор функций, которые возвращают файлы роботов и объектов:

getRobotSpriteSheet(robotType)
возвращает путь до ресурсов исполнителя.

GetObjectsInfo()
возвращает словарь, который ставит в соответствие идентификатору объекта его изображение

getRobotsInfo()
возвращает словарь, который ставит в соответствие идентификатору робота его изображение (если тип робота один, то в словаре будет один элемент)

Далее идут функции для использования модулей в редакторе:

getSettings ()
создает визуальное представление с настройками робота

getMapElementsType()
возвращает типов элементов карты исполнителя

getRobotDisplayedName() и **getDisplayedNames()** возвращают локализованное название уровня

getMapTileset()
возвращает путь до ресурсов обстановки
getResources(robot)
возвращает набор ресурсов исполнителя

Требуется также имплементировать функции создания робота и объектов:

generateObject(objectType) создает и возвращает объект получаемого типа

generateRobot(robotType) создает и возвращает робота соответствующего типа

Для редактора необходимы функции, которые возвращают флаг, в каких случаях следует останавливать изменение размера карты:

stopResizingWidth(map)

stopResizingHeight(map)

Для локализации необходимы функции, которые возвращают соответствие

getLocalization(language)

которая в соответствии от языка системы ставит в соответствие названиям объектов, которые вы указали в модуле (такие как название уровня, робота и так далее) соответствующий им перевод

Далее набор функция для заданий робота:

generateTaskLayer(level, task) создает непосредственно визуальное представление задания

getTaskLabel()
возвращает название задания исполнителя

addLevelTask(level)
добавляет в список заданий уровня задание исполнителя

Ниже пример набора констант для Робота. Сам набор располагается непосредственно за описаниями функций класса:

```
pm.RobotLevelModule.RobotType =
"robot-robot";
pm.RobotLevelModule.Type = "ro-
bot";
pm.RobotLevelModule.Lamp =
"lamp";
pm.RobotLevelModule.Object =
"tree";
pm.RobotLevelModule.ObjectTypes=
{
    RobotLamp0:"RobotLamp0",
    RobotLamp1:"RobotLamp1",
    RobotObject0:"RobotObject0",
    RobotObject1:"RobotObject1"
};
```

И вызываем функцию регистрации модуля:

```
pm.moduleUtils.register(
      new pm.RobotLevelModule
);
```

Результатом создания описания - файла с точки зрения разработчика исполнителя Робот будет включение исполнителя в систему ПиктоМир.

Работа выполнена по теме 0065-2018-0017 госзадания 2018 года в отделе учебной информатики ФГУ ФНЦ НИИСИ РАН.

Development of software executors of the PiktoMir system

N.O.Besshaposhnikov, K.A. Maschenko, A.E. Orlovsky

Abstract. School and propaedeutic courses in computer science, using respectively school algorithmic language and Pikto's language, are based on the KuMir and PiktoMir pedagogical software. Pupils learn the subject by creating algorithms to control a wide range of software executers and robots. At the same time, all software executors of the KuMir system are in fact a separate software module, methods for connecting it to the KuMir are fixed and described in open sources. Thus, modern teachers can independently design and create new software executors for the KuMir system and use them in their teaching practice. The article describes the approach to creating software implementers in the PiktoMir system. The unified methods of connecting the newly created software executors and robots are given.

Keywords: Programming executors, modularity, PiktoMir, Kumir, algorithmic.

Литература

1. Cocos2d-x - open-source game framework, with a thin platform dependent layer. [Электронный ресурс] / Режим доступа: http://www.cocos2d-x.org/cocos2dx свободный.
2. Qt - cross-platform application framework and widget toolkit. [Электронный ресурс] / Режим доступа: https://www.qt.io свободный.
3. I.B.Rogozhkina, A.G.Kushnirenko. PiktoMir: Teaching Programming Concepts to Preschoolers with a New Tutorial Environment. «World Conference of Educational Technology and Researches», (2011), 216 – 220.
4. Н.О.Бесшапошников, А.Н.Дедков, Д.Б.Еремин, А.Г.Леонов. ПиктоМир как кооперативная среда для обучения основам программирования дошкольников и младших школьников. «Труды НИИСИ РАН», т.5 (2015), № 1, 138 – 141.
5. А.Г.Кушниренко. Информатика. 7—9 классы. Рабочая программа. Методические комментарии: учебно-методическое пособие. М., Дрофа, 2017.
6. А.Г.Кушниренко, Г.В.Лебедев, Я.Н.Зайдельман. Информатика, 7-9 классы. М., Дрофа, 2000.
7. А.Г.Леонов. Тенденции объектно-ориентированного программирования в разработке системы КуМир. «Программные продукты и системы», (2012), № 4, 245 – 249.
8. А.Г.Леонов, Ю.А.Первин. Логическое проектирование педагогических программных средств. «Ярославский педагогический вестник», (2013), № 4.
9. А.Г.Леонов, Ю.А.Первин. Переход от непосредственного управления исполнителями к составлению программ в пропедевтическом курсе информатики. «Ярославский педагогический вестник», (2013), № 3.
10. А.Г.Леонов. Обучение информатике: программные исполнители в системе КуМир. «Труды большого московского семинара по методике раннего обучения информатике. – Москва, 2008 г. (тезисы)», М., Изд-во Российский государственный социальный университет, 2008, т.1, 17 – 30.
11. Пиктомир. [Электронный ресурс] / Режим доступа: https://www.niisi.ru/piktomir/m2018.pdf свободный.
12. Д.В.Хачко, В.В.Яковлев, Н.М.Субоч, Т.Р.Джелядин, М.А.Ройтберг, А.Г.Кушниренко, А.Г.Леонов. Кумир 1.9. Практикумы и исполнители. Средства интенсификации обучения. «Свободное программное обеспечение в высшей школе: VII конференция. – Москва, 2015 г. (тезисы)», М., Изд-во Институт логики, 2012, 36.

От Робота к Роботору. Олимпиадные задачи в системе ПиктоМир

А.Г. Леонов[1], Ю.А. Первин[2]

[1,2] ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

[1] ФГБОУ ВО «Московский государственный университет имени М.В. Ломоносова: механико-математический факультет», Москва, Россия и ФГБОУ ВО «Московский педагогический государственный университет», Москва, Россия, E-mail: dr.l@vip.niisi.ru

Аннотация: В настоящее время все более актуальным является процесс понижения возраста учащихся для первого знакомства с основами алгоритмизации и программирования. Этот закономерный процесс предвосхитили великие ученые современности в области информатики и естественно-научных дисциплин. Однако, понижение возраста начала изучения основных понятий программирования нужно осуществлять вдумчиво и последовательно. Обучение основам алгоритмизации и программирования должно происходить как непрерывный процесс, начиная от знакомства с множествами предметов в дошкольные образовательные учрежденья до высших образовательных организаций. В старшей школе и в университетах обучение программированию должно происходить с получением обучаемым глубоких фундаментальных знаний в области математики и смежных естественно-научных дисциплин. В статье приводится опыт работы со студентами МГУ им. М.В. Ломоносова и МПГУ, на примере решения олимпиадных школьных и университетских задач по программированию с использованием систем КуМир и Роботор. Авторы предлагают рассмотреть подходы к конструированию исполнителей с аналогичным функционалом в системе ПиктоМир, открывающей двери в мир пропедевтического курс информатики. Предлагается методика подготовки к решению таких задач и для проведения олимпиад по информатике в начальной школе с использованием системы ПиктоМир. Олимпиады по информатике на заданиях с математическим наполнением и командные турниры в рамках дистанционного обучения усиливают межпредметные связи и мотивируют детей к освоению сквозного курса школьной информатики. Основной вывод из параллельного формирования знаний и навыков в области информатики и информационных коммуникационных технологий: к ним разумно приступать не в университете и общеобразовательных учреждениях, а в более раннем возрасте – в пропедевтическом курсе.

Ключевые слова: информатика, олимпиадные задачи по программированию, Роботор, Робот, ПиктоМир, начальная школа, профильные классы, алгоритм, программа, язык пограммирования.

Современный общество уже фактически живет в информационной эпохе, которая, в первую очередь, характеризуется основополагающим влиянием процесса цифровизации, иногда именуемым цифровой трансформацией, всех областей жизнедеятельности человека: производства, управления, образования и т.п. [1]. Цифровая трансформация диктует определенные, естественно, высокие требования к уровню компетенции человека в области информационных и коммуникационных технологий. «Тысячи профессий меняют свое лицо. Миллионы людей - операторов производства, наладчиков, машинисток, банковских служащих, продавцов-контролеров, библиотекарей, монтажников, секретарей, сборщиков на конвейере - садятся за полностью переоборудованные рабочие места, на которых ЭВМ становится их партнером и собеседником. И если даже этот партнер дружествен и надежен, у работника должна произойти глубокая психологическая и квалификационная перестройка для того, чтобы сохранить свою целостность и достоинство в этой новой обстановке. Мы уже сейчас говорим о миллионах людей, вовлеченных в этот процесс (в одной Западной Европе число терминалов ЭВМ и линий передачи данных приближается к миллиону), а через пару поколений это коснется практически каждого человека, вовлеченного в общественное производство», так в далеком 1981 году академик А.П.Ершов предвосхитил возникновение цифровой революции и указал направление развития современного человека постиндустриальном обществе [3, с. 9]. Не случайно, не только ведущие отечественные ученые [2], но и известные зарубежные фонды [21] и издания указывают, что формирование знаний и навыков в области информатики и ИКТ, разумно начинать не в университетах и общеобразовательных учреждениях, а в более раннем, дошкольном возрасте [16] [19].

Так в MIT Media Laboratory (США) группа энтузиастов разрабатывают программно-

аппаратную систему, при помощи которой дети в возрасте от четырех до восьми лет смогут программировать различных интерактивных роботов, просто выбирая и прикрепляя наклейки, изображающие определенные действия роботов, к листам бумаги. Для ученых программирование детьми интерактивных роботов служит исследовательским инструментом. Важно понять, в каком возрасте и каким способом наилучшим образом интегрировать этот новационный курс в учебные программы дошкольников и младших школьников, при этом, естественно, что целью является обучение, знакомство детей с основными принципами программирования [18].

Именно в этом возрасте, относящемся к старшим годам стадии так называемого операционального мышления, где-то между пятью и семью годами, когда происходит интенсивное развитие речи и активизируется процесс интериоризации внешних действий с предметами, для детей появляется окно возможностей знакомства с основными понятиями алгоритмики и программирования, которые, все равно, им будет необходимо освоить ранее или позже. Несмотря на то, что в это время у ребенка проявляется определенный эгоцентризм мышления, что, подчас, выражается в сложностях принятия позиции учителя, при коллективной работе (работе в группе или парами) попытка ребенка объяснить постановку и(или) решение задачи, именно «эгоцентрическим характером детского стиля» изложения является эффективным орудием освоения материала, когда в ребенок, работая в коллективе, рассказывает для самого себя, не стремясь к тому, чтобы добиться понимая у собеседника [17]. Тем самым, через свой эгоцентризм ребенок лучше воспринимает новые базовые понятия алгоритмики и элементов программирования, часть из которых хорошо ложится на уже сформированный понятийный аппарат дошкольника. Если для старшей школы используется понятия программно-управляемых исполнителей [11], то для ребенка уместно замена столь сложной для понимания и всеобъемлющей сущности на привычные понятия, уже знакомые из повседневной жизни: Робота [22] или Черепахи [23]. При этом существует масса эффективных методик, позволяющих оценить достижения учеников. Так групповые тесты, могут существенно сократить время получения результатов обработки тестов. Использование современных автоматизированных систем, использующих достижения передовых цифровых информационных технологий, упрощает методики и процедуры проведения тестов, а также обработки результатов с возможностью накопления результатов, формирования портфелей достижения учащихся. Это позволяет в школьном и дошкольном образовании получать регулярную картину оценки успешности усвоения материала учащихся, что, конечно не заменяет итоговых экзаменационных аттестаций [12].

Необходимо отметить, что уже на протяжении достаточно большого периода времени, начиная с 90-х годов накоплен положительный опыт, не только обучения младшеклассников основам информатики и ИКТ, но и включения этого пропедевтического курса в системный, непрерывный курс по указанной дисциплине с первого по одиннадцатый класс [9] [14].

В этом курсе в общей школе используется учебный алгоритмический язык с национальной лексикой, придуманный академиком А.П.Ершовым и названный им школьным алгоритмическим языком. Не смотря на свою ориентацию как учебного языка программирования, школьный алгоритмический язык, имеет ряд неоспоримых преимуществ, позволяющих эффективно использовать его в курсе информатики и ИКТ в общей школе на протяжении 30 и более лет [5]. В современном виде алгоритмический язык нашел свою нишу между начальным знакомством с основными понятиями и конструкциями процедурного программирования в начальной школе и профильными курсами для старшеклассников, где возможно и закономерно использование производственных языков программирования [8].

Интересно, что использование школьного алгоритмического языка программирования оказалось уместным не только в школе, но и в университетах (МГУ, МПГУ) в качестве стартового комплекта для освоения основ программирования и алгоритмизации, алгоритмов и базовых структур данных. Эффективное использование алгоритмического языка в университетских курсах стало возможно благодаря системе программирования КуМир [20], которая не только позволяет быстро и удобно освоиться новичку с новой дисциплиной, но и содержит широкий набор практикумов с автоматической проверкой, направляющих обучаемого по методически продуманному пути изучения курса с широкими возможностями самоконтроля. Система КуМир может использоваться как для проведения курсов по информатике и программированию, так и для подготовки и решения олимпиадных задач по программирование [13]. Междисциплинарные связи между информатикой и математикой были продемонстрированы добавлением к набору базовых программно-управляемых исполнителей, исполнителя Роботор [4].

Обычный Робот, входящий в базовый набор исполнителей КуМира, расположен на

ограниченной прямоугольной клетчатой плоскости, ячейки которой могут быть разделены непроходимыми для Робота стенами. Исполнитель Робот имеет набор простых, для понимания ученика, команд: “вверх”, “вниз”, “вправо”, “влево”, “закрасить”. По командам движения, Робот, если нет стены, перемещается в соседнюю клетку. По команде “закрасить” Робот закрашивает клетку, в которой стоит. В распоряжении программиста еще имеется ряд команд обратной связи с Роботом, когда исполнитель может сообщить, что вверху/внизу/справа/слева есть или нет проход (стоит стена или стена отсутствует), а также ответить, стоит ли Робот на закрашенной клетке или нет.

Мотивирующей легендой для обучающихся может быть рассказ о необходимости управлять Роботом, находящимся на далекой планете за многие миллионы и миллиарды километров от Земли.

Управляющий радиосигнал до этой планеты идет достаточно продолжительное время, что не позволяет управлять Роботом интерактивно, непосредственно посылая ему с Земли команду за командой, наблюдая за результатом выполнения.

Тем самым, возникает необходимость составлять план будущих действий Робота, то есть программу, для решения задачи. При этом обстановка, в которой действует Робот не фиксирована, а это значит, что от ученика требуется составлять универсальный алгоритм действия Робота, при котором в программе выполняемые команды Роботом будут зависеть от конкретной обстановки.

Для более математически интересных задач был разработан преемник Робота – Роботор, используемый на более сложных связанных поверхностях: тор, конечный или бесконечный цилиндр, лист Мебиуса, бутылка Кляйна и т.п.

Легенда появления задач для Роботора говорит, что это обычный Робот с тем же самым стандартным набором команд движения и закрашивания, как и уже знакомого учащимся обыкновенного Робота, расположенного на плоскости, только Роботор располагается на, например, тороидальной поверхности космического корабля, изготовленного из почти квадратных плит. Размер и свойства поверхности корабля Роботору, как правило, заранее не известны.

Исполнитель Роботор включает следующий набор команд:

- команды перемещения на одну клетку - влево, вправо, вверх, вниз;
- команды закрашивания клетки – закрасить;
- команды-вопросы, возвращающие логическое значение - слева свободно, справа свободно, сверху свободно, снизу свободно, и их отрицания - слева стена, справа стена, сверху стена, снизу стена;
- команды-вопросы про состояние клетки - клетка закрашена, клетка чистая;
- и некоторые другие.

Задача. РобоТор высадился на поверхность тороидального межзвездного корабля (на поверхности корабля нет перегородок и нет закрашенных клеток). Требуется определить площадь поверхности корабля (площадь измеряется количеством клеток) и вернуться в стартовую клетку.

Алгоритм решения задачи понятен старшекласснику, знакомому со школьным алгоритмическим языком и приводится здесь в сокращенном виде без комментариев. Подробнее можно прочитать статье [4].

алг площадь
нач цел высота, ширина
- закрасить
- верх; высота:=1
- нц пока клетка не закрашена
- вверх; высота:= высота +1
- кц
- вправо; ширина:=1
- нц пока клетка не закрашена
- вправо; ширина:= ширина +1
- кц
- вывод «Площадь тора равна», ширина * высота, нс

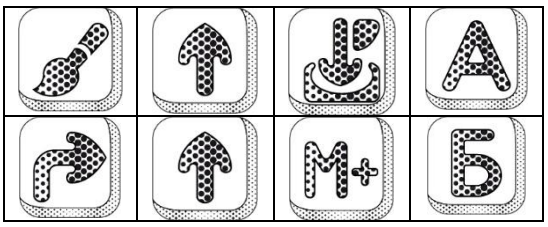
кон

Для Роботора были придуманы математические задачи, которые интересны не только старшим школьникам, желающим получить дополнительные знания и межпредметные компетенции в смежных дисциплинах математики и информатики, но и для студентов университетов математической направленности. Тороическая поверхность сложнее, чем плоскость. Так, любая несамопересекающаяся замкнутая ломаная на плоскости гарантированно разделяет плоскость на две, когда, напротив, на торической поверхности замкнутая несамопересекающаяся стена может разделить его поверхность на две части, а может и не разделять. Известно математическое доказательство, что стена разделяет поверхность клетчатого тора на две части, только в случае, когда она не «наматывается» на тор ни по меридиану, ни по параллели. Ниже приведен пример трудной задачи.

Роботор находится на поверхности тороидального межзвездного корабля. Вверху от Роботора находится замкнутая несамопересекающаяся стена. Определить, делит ли эта стена поверхность корабля на две части.

Несмотря на алгоритмическую сложность некоторых математических задач на торе и цилиндре, простейшие задачи могут быть использованы в качестве олимпиадных, а также дополнительного внеурочного материала для формирования межпредметных компетенций и в начальной школе [15]. Однако, наибольшую, неалгоритмическую сложность ученики могут испытать при работе с текстовыми языками программирования, даже если будет использоваться система программирования КуМир на школьном алгоритмическом языке. Эта проблема была решена созданием бестекстовой системы программирования ПиктоМир, которая прошла успешную апробацию в дошкольных образовательных учреждениях [6]. Работа малышей в пиктографической системе программирования проста для их понимания и наглядна. Дети знакомятся с основами алгоритмики и основными понятиями алгоритмических языков при составлении простейших программ для робота Вертуна. Аналогично своему старшему брату – Роботу из курса информатики и ИКТ в системе КуМир, Вертун также «живет» на клетчатом поле, часть которых может быть отделена стенами друг от друга. Система команд Вертуна, который в отличие от Робота ориентирован на поле, еще более упрощена:

- команда перемещения на одну клетку вперед;
- команды поворота направо/налево;
- команда закрашивания клетки – закрасить;
- команды-вопросы, впереди свободно/впереди стена;
- команды-вопросы про состояние клетки - клетка закрашена, клетка не закрашена.



По легенде Вертун находится на космодроме одной из планет, и после очередного взлёта космических кораблей, восстанавливает покрытие поля, ремонтируя (закрашивая) разрушенные клетки [7].

В рамках реализации программы ФГОС для начальной школы учащиеся младших классов на уроках информатики и математики должны овладеть основами логического и алгоритмического мышления, пространственного воображения, приобрести необходимые вычислительные навыки применять математические знания и представления для решения учебных задач, а также начальный опыт применения математических знаний в повседневных ситуациях.

Логические задачи для Вертуна, расположенного на циклически замкнутом поле космодрома (фактически торе, но не формализуя это геометрическое понятие для детей) приобретают самостоятельный интерес для учащихся, формируя межпредметные связи в смежных областях [10].

Школьникам можно пояснить, что Вертун при достижении края платформы-космодрома переносится фантастическим образом на другой край по вертикали или горизонтали соответственно.

Можно, также предусмотреть визуальную реализацию перемещения Вертуна, исчезающего с одной стороны и появляющегося с другой.

При этом часть задач, решение которых было сопровождено с определёнными трудностями, имеет простое решение для Вертуна на торе при использовании счетчика – Волшебного кувшина (см. *Примечание*), встроенного исполнителя в системе ПиктоМир.

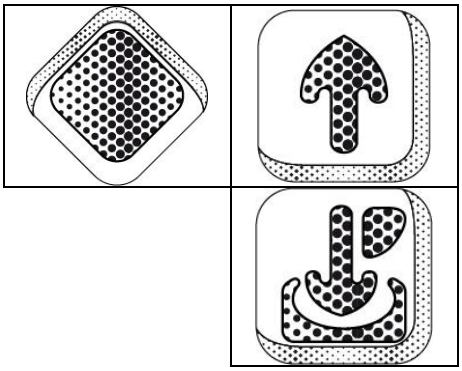
Примечание: В ПиктоМире имеются Счетчики – Волшебный кувшин и Кувшин с памятью.

В кувшин можно добавлять камни (по одному), брать (также по одному), выбрасывать все камни из кувшина, использовать в программе условия «Кувшин пуст?», «Кувшин не пуст?», а также работать с памятью кувшина, добавляя или вычитая содержимое кувшина из памяти (аналогично работе с памятью на простейшем калькуляторе).

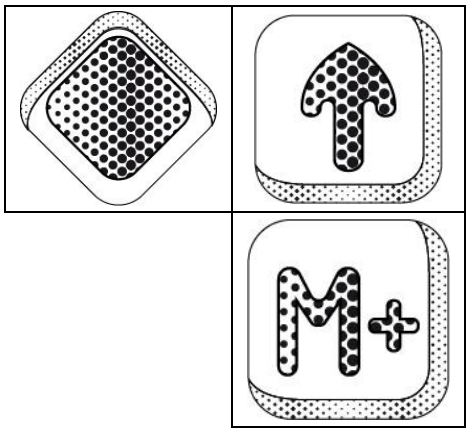
Задача. Космодром планеты свободен от стен. Вертун смотрит на север. Требуется определить площадь космодрома (площадь измеряется количеством клеток) и вернуться в стартовую клетку. Алгоритм решения задачи аналогичен приведенному выше алгоритму, записанному на школьном алгоритмическом языке и приводится здесь примерно в том виде, как это видит ученик на экране планшета:

Основной алгоритм.

А



Б



Вертун, вначале, закрашивет клетку, на которой стоит, чтобы, по завершению алгоритма вернуться на прежнее место. Далее смещается на один шаг на север, положив камень в кувшин. Во вспомогательном алгоритме А происходит подсчет «высоты» поля (условие выполнения цикла – клетка не закрашена). Пока Вертун не вернется в исходную клетку, он движется вперед, добавляя в кувшин камень (увеличивая счетчик на 1). Тем самым по окончании алгоритма А, Вертун снова будет в исходной клетке, при этом в кувшине будет столько камней, сколько шагов сделал Вертун. Так как Вертун находится на торе (космотдром замкнут по вертикали), то алгоритм А завершится и в кувшине будет «высота» поля. Далее Вертун поворачивается направо, добавляет к пустой памяти «высоту» поля из кувшина и делает один шаг. Во вспомогательном алгоритме Б происходит финальный подсчет площади поля (условие выполнения цикла – клетка не закрашена). Пока Вертун не вернется в исходную клетку, он движется вперед, добавляя в память число камней из кувшина (увеличивая память на «высоту»). Тем самым по окончании алгоритма Б, Вертун снова будет в исходной клетке, при этом памяти столько раз увеличивалась на «высоту», сколько шагов сделал Вертун. Здесь суммирование заменит умножение. По окончанию работы алгоритма в окне памяти кувшина будет находиться площадь космодрома. Описание пиктографических команд приведено ниже:

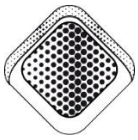
Закрасить

Вперед

Направо

Алгоритм А

Алгоритм. Б



Пока клетка НЕ закрашена

Положить камень в кувшин

Прибавить содержимое кувшина к памяти

На школьном алгоритмическом языке этот алгоритм выглядел бы, примерно, следующим образом:

алг
- закрасить
- вперед
- положить камень в кувшин
- А
- направо
- вперед
- добавить содержимое кувшина к памяти
- Б

кон
алг А
- нц пока клетка не закрашена
- вперед
- положить камень в кувшин
- кц

кон
алг Б
- нц пока клетка не закрашена
- вперед
- добавить содержимое кувшина к памяти

. кц
кон

Можно заметить, что в пиктографическом виде алгоритм выглядит достаточно прозрачно для понимания его учениками младших классов. Решение аналогичных задач олимпиадного программирования в начальной школе с использованием системы ПиктоМир позволит учащимся познакомиться с новыми геометрическими формами и овладеть способами измерения длин и вычисления площадей, что без сомнения, повысит их компетентность в области математики и информатики.

Авторы благодарны А.Г. Кушниренко и А.А. Ефремову за полезные обсуждения.

Работа выполнена по теме 0065-2018-0017 госзадания 2018 года в отделе учебной информатики ФГУ ФНЦ НИИСИ РАН.

From Robot to Robotor. The competition problems in PiktoMir

A.G. Leonov, U.A. Pervin

Abstract: At this present, the process of lowering the age of students for the first acquaintance with the basics of algorithmization and programming is becoming more and more urgent. This natural process was anticipated by the great scientists of our time in the field of informatics and natural sciencees. However, the lowering of the age of the beginning of the study of the basic concepts of programming must be carried out in a thoughtful and consistent manner. Learning the basics of algorithmization and programming should take place as a continuous process, starting with getting to know lots of subjects from pre-school educational institutions to higher education organizations. In high school and in universities, programming training must occur with the acquisition of deep fundamental knowledge in the field of mathematics and related natural science disciplines. The article shows the experience of working with students of the Moscow State University. M.V. Lomonosov and MPGU, on the example of solving the olympiad school and university problems in programming using KuMir and Robotor systems. The authors propose to consider approaches to the design of performers with similar functionality in the PictorMir system, which opens the door to the world of the propaedeutic informatics course. The method of preparation for solving such problems is proposed, and for the olympiad in informatics in elementary school using the PictorMir system. Olympiad in informatics on problems with mathematical content and team tournaments within the framework of distance learning strengthen interdisciplinary connections and motivate children to master the end-to-end course of school computer science. The main conclusion from the parallel formation of knowledge and skills in the field of informatics and information communication technologies: it is reasonable to start them not at the university and general educational institutions, but at an earlier age - in the propaedeutic course.

Keywords: Informatics, competition problems in programming, Robotor, Robot, PictoMir, elementary school, profile classes, algorithm, program, programming language.

Литература

1. В.Б.Бетелин. О новой технологической революции и готовности к ней экономики России. "Экономист", (2018), №2, 3 – 9.
2. Е.П.Велихов, В.Б.Бетелин, А.Г.Кушниренко. Промышленность, инновации, образование, наука в России. М., Наука, 2010.
3. А.П.Ершов. Программирование — вторая грамотность [Электронный ресурс] Архив академика А.П.Ершова, 1981 / Режим доступа: http://ershov.iis.nsk.su/ru/node/771564 свободный.
4. А.А.Ефремов, А.Г.Кушниренко, А.Г.Леонов. Космический робот робоТор. «Информатика», (2010), № 21, 4 – 9.
5. Г.А.Звенигородский. Сравнительный анализ языков программирования, используемых в школьном учебном процессе. «Проблемы школьной информатики», (1986), 24 – 38.
6. А.Д.Кисловская, А.Г.Кушниренко. Методика обучения алгоритмической грамоте дошкольников и младших школьников. «Информационные технологии в обеспечении федеральных государственных образовательных стандартов: Международная научно-практическая конференция. – Елец, 16-17 июня 2014 года. (тезисы)», Елец, Изд-во ЕГУ им. И. А. Бунина, т.2, 2014, 3 – 7.
7. А.Г.Кушниренко, И.Б.Рогожкина, А.Г.Леонов. ПиктоМир: пропедевтика алгоритмического языка (опыт обучения программированию старших дошкольников) [Электронный ресурс] ИТОРОИ-2012 / Режим доступа http://ito.edu.ru/sp/SP/SP-0-2012_09_25.html свободный.

8. А.Г.Кушниренко, А.Г.Леонов, М.А.Ройтберг. Знакомим дошкольников и младших школьников с азами алгоритмики с помощью систем ПиктоМир и Кумир. «Труды НИИСИ РАН», т. 5 (2017), № 1, 134 – 137.

9. Е.Я.Коган, Ю.А.Первин. Курс «Информационная культура» – региональный компонент школьного образования. «Информатика и образование», (1995), № 1, 8.

10. А.Г.Леонов, Ю.А.Первин. Учебные и тестовые логические задачи в пропедевтическом курсе информатики. «Информатика и образование», (2015), № 9, 32 – 36.

11. А. Г.Леонов, Ю.А.Первин. Переход от непосредственного управления исполнителями к составлению программ в пропедевтическом курсе информатики. «Ярославский педагогический вестник», т. 3 (2013), №3, 17 – 30.

12. А. Г.Леонов, Ю.А.Первин. К измерению качественных оценок эффективности методики обучения элементам информатики в пропедевтическом школьном курсе. «Ярославский педагогический Вестник», (2015), 23 – 34.

13. Олимпиадные задачи по программированию [Электронный ресурс] Портал школы 179 / Режим доступа:https://server.179.ru/tasks/olymp/ свободный.

14. Ю.А.Первин, Ю.М.Горвиц, З.А.Зарецкая, Д.В.Зарецкий. Модуль первого (пособие для учителя). Самара, СИПКПРО, 1994.

15. Первин Ю.А., Первин Т.Ю. Множества, элементы, признаки (книга для школьников 2-го класса). Самара, СИПКПРО, 1994.

16. A is for Algorithm [Электронный ресурс] The Economist, April 23, 2014 / Режим доступа:https://www.economist.com/news/international/21601250-global-push-more-computer-science-classrooms-starting-bear-fruit/ свободный.

17. Piaget Jean, Le langage et la pensée chez l'enfant : études sur la logique de l'enfant. Neuchâtel : Delachaux & Nestlé, 2002, P. 210.

18. Larry Hardesty: Teaching programming to preschoolers [Электронный ресурс] MIT News March 11, 2015 / Режим доступа: http://news.mit.edu/2015/teaching-preschoolers-programming-0312 свободный.

19. Matt Richtel: Reading, Writing, Arithmetic, and Lately, Coding [Электронный ресурс] The New York Times, May 10, 2014 / Режим доступа: http://www.nytimes.com/2014/05/11/us/reading-writing-arithmetic-and-lately-coding.html/ свободный.

20. Nikita Besshaposhnikov, Anatoli Kushnirenko, Alexander Leonov, Pictomir: how and why do we teach textless programming for preschoolers, first graders and students of pedagogical universities [Электронный ресурс] Proceedings of the 13th Central & Eastern European Software Engineering Conference in Russia, CEE-SECR '17, ACM Press (New York, N.Y., United States) / Режим доступа: https://dl.acm.org/author_page.cfm?id=99659235857 свободный.

21. The NMC Horizon Report: 2017 Higher Education Edition [Электронный ресурс] New Media Consortium, 2017/ Режим доступа: https://www.nmc.org/publication/nmc-horizon-report-2017-higher-education-edition/ свободный.

22. I.B.Rogozhkina, A.G.Kushnirenko. PiktoMir: Teaching Programming Concepts to Preschoolers with a New Tutorial Environment. “World Conference of Educational Technology and Researches, July, 2011», 2006.

23. Seymour Papert, Educational Computing: How Are We Doing? «T.H.E. (Technological Horizons in Education)», (1997), P. 78 – 80.

Современное состояние электронной библиотеки «Научное наследие России»

Н.Е. Каленов[1], С.А. Кириллов[2], И.Н. Соболевская[3], А.Н. Сотников[4]

Межведомственный суперкомпьютерный центр РАН – филиал ФГУ ФНЦ НИИСИ РАН, Москва, Россия,

E-mail's: [1] nek@benran.ru, [2] skirillov@jscc.ru, [3] nikfirst@jscc.ru, [4] ASotnikov@jscc.ru

Аннотация: В статье описано состояние электронной библиотеки «Научное наследие России» (ЭБ ННР) на конец ноября 2018 года. Средствами электронной библиотеки обеспечивается интеграция разнородных материалов, в том числе электронных копий объектов библиотечного, архивного и музейного хранения, представленных в виде текстов, графических образов, аудио-видео объектов. В работе приведен принцип формирования фондов ЭБ ННР. Показана структура метаданных по каждому виду отражаемых в Библиотеке объектов. Описана работа по развитию ЭБ ННР. Предложена статическая информация о ресурсах ЭБ ННР. Описаны также размещенные на сайте ЭБ ННР мультимедийные ресурсы – оцифрованные фрагменты научных фильмов и 3D-модели музейных экспонатов. Проведен анализ востребованности ресурсов ЭБ ННР.

Ключевые слова: электронная библиотека, Научное наследие России, 3D-модель, музейный объект, виртуальная выставка.

Электронная библиотека «Научное наследие России» (ЭБ ННР) создавалась по решению Президиума Российской академии наук с 2007 года в рамках специальной целевой научной программы. В 2010 году она была введена в промышленную эксплуатацию и поддерживается в открытом доступе в интернет по адресу http://e-heritage.ru/.

Основной целью создания ЭБ является предоставление через Интернет в свободном режиме информации о выдающихся российских ученых, внесших вклад в развитие фундаментальных естественных и гуманитарных наук, с возможностью ознакомления с полными текстами опубликованных ими наиболее значительных работ, архивными и музейными материалами, связанными с ними [1-7].

Разработчиками технологии и программного обеспечения создания и поддержки ЭБ ННР являются Межведомственный суперкомпьютерный центр (МСЦ) РАН (филиал ФГБУ НИИСИ РАН), Вычислительный центр (ВЦ) РАН (подразделение ФИЦ «Информатика и управление»), Библиотека по естественным наукам (БЕН) РАН.

В основу наполнения ЭБ ННР положен принцип распределенной подготовки информации в сочетании с централизованной поддержкой хранилища данных и единой точкой входа для пользователей. Участники проекта дают предложения по включению в нее определенных ресурсов. Предложения рассматриваются редакционной группой в МСЦ РАН; сообщения о принятии или отклонении предложений автоматически поступают их «авторам» через технологический блок системы. Получив подтверждение своих предложений, участники проекта подготавливают и вводят в технологический блок (http://meta.e- heritage. ru/) метаданные согласованного ресурса, осуществляют оцифровку документов и направляют цифровую копию в редакторскую группу, которая осуществляет необходимую обработку материалов и публикует информацию на сайте ЭБ.

В процессе разработки ЭБ была определена структура метаданных по каждому виду отражаемых в ней объектов (персоны, публикации, архивные материалы, музейные предметы, изображения и мультимедийные файлы), разработаны необходимые для формирования, модификации, обеспечения сохранности и поиска данных программные средства; определена организационная структура, обеспечивающая развитие и поддержку ЭБ ННР. В [3] приводятся организационные, технологические и технические решения, положенные в основу функционирования ЭБ ННР.

Научная работа по развитию ЭБ ННР ведется в направлении отражения в Библиотеке новых видов информационных ресурсов, таких как 3D-модели, оцифрованные фильмы и специальные тематические коллекции.

За последнее время в ЭБ ННР включены, наряду с информацией о ряде ученых с их фотографиями и полными текстами опубликованных работ, новые виды ресурсов, в частности, электронный архив Института этнологии и антропологии (ИЭА) РАН, включающий коллекцию фотографий, сделанных во время этнологических экспедиций сотрудников Института в различные регионы страны. Страница ЭБ ННР с результатом поиска фрагментов коллекции, связанных с «русскими переселенческими группами» в Закавказье, приведена на рис. 1.

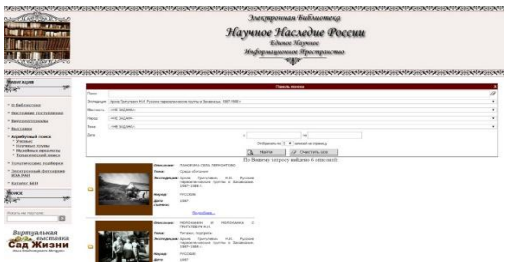


Рис. 1. Фрагмент страницы ЭБ ННР, демонстрирующий элементы коллекции ИЭА РАН

Наряду со статической информацией на сайте ЭБ ННР также размещаются мультимедийные ресурсы – оцифрованные фрагменты научных фильмов и 3D-модели музейных объектов. Перейдя по ссылке «Виртуальная выставка «Сад Жизни», пользователь попадает на ресурс, подготовленный совместно с Государственным биологическим музеем им. К.А. Тимирязева и Российским государственным архивом кинофотодокументов.

Рис. 2. Фрагмент страницы со ссылками на оцифрованные киноматериалы

Эта выставка отражает разнообразную информацию о жизни и деятельности И.В. Мичурина. На рис. 2 показана страница ЭБ ННР, содержащая ссылки на оцифрованные фрагменты фильмов об этом ученом, предоставленные Российским государственным архивом кинофотодокументов.

Рис. 3. Макеты плодов, выведенных И.В.Мичуриным

Эти видеодокументы доступны для просмотра любому пользователю.

Перейдя по ссылке «Коллекция 360º, можно просмотреть изображения макетов плодов, которые вывел И. В. Мичурин (Рис. 3). Каждое изображения, представленное на этой странице, является активной ссылкой, при переходе по которой пользователь увидит 3D-модель соответствующего плода.

В настоящее время ведется работа по формированию виртуальной выставки «Портреты по скелетам», посвященной научному наследию М.М. Герасимова. В этом проекте также представлены цифровые 3D-модели музейных скульптурных композиций, созданных М.М. Герасимовым и его школой. Для формирования виртуальной коллекции 3D-моделей с целью предоставления ее широкому кругу пользователей через интернет используется, так называемый, метод интерактивной мультипликации. Этот метод основан на смене фиксированного набора кадров с изображением объекта. Такая смена кадров осуществляется с помощью специализированных интерактивных программ визуализации, имитирующих смену точки взгляда на исходный объект. На рисунке 4 показана последовательность кадров, позволяющая создать «интерактивную анимацию» для представления музейного объекта на сайте виртуальной выставки.

Рис. 4. Пример последовательности кадров 3D-модели

В настоящее время (по состоянию на конец ноября 2018 года) в ЭБ ННР опубликованы данные о более чем 4800 ученых. Для более полного отражения в ЭБ ННР информации о российской науке сотрудниками МСЦ РАН ведется мониторинг сети Интернет на предмет выявления «внешних» общедоступных ресурсов, относящихся к российским ученым. Найденные ссылки на соответствующие сайты музеев, архивов, а также на отдельные страницы, посвященные ученым, включаются в ЭБ ННР.

Одновременно с вводом в электронный каталог биографических данных ведется рабо-

та по поиску внешних источников знаний по авторам.

Электронная библиотека содержит ссылки на сайты музеев, архивов, а также различные электронные страницы в интернете, которые содержат значимую информацию о жизни выдающихся ученых, представленных в ЭБ ННР.

Фонды электронной библиотеки «Научное наследие России» состоят из двух частей: опубликованных на сайте ЭБ ННР цифровых объектов и объектов, находящихся в работе.

На текущий момент в технологический электронный каталог введены метаданные на более чем 25000 изданий, из них прошли полную проверку и загружены на сайт 22089 изданий (4180824 страниц). Из них 10 опубликовано в 17 веке, 5159 – в 18-м, 6345 – в 19-м, остальные работы 20-го века.

По состоянию на конец ноября 2018 года в ЭБ ННР введены развернутые сведения о 5993 ученых, работавших в России с 17-го по начало 20-го веков. Из них 81 родился в 17 веке, 489 – в 18-м, 3388 – в 19-м, остальные в 20-м веке.

Большая часть публикаций, представленных в ЭБ ННР, издана на русском языке (таблица 1).

Кроме русского языка в библиотеке широко представлены книги на латинском языке, как правило, это российские академические издания XVIII века, и на французском языке, который преимущественно являлся языком научного общения в XIX веке.

Таблица 1

Распределение публикаций по языку издания

Язык	% на сайте	% в техн. базе
Русский	88,67	87,14
Французский	3,55	3,74
Латинский	3,22	3,09
Английский	0,97	1,41
Немецкий	3,18	3,22
Прочие	0,41	1,40

Кроме того, в электронной библиотеке «Научное наследие России» широко представлены издания, отражающие все периоды развития научной мысли со дня основания Академии наук. Данные, представленные в таблице 2, характеризуют, в частности, развитие научной активности в России в разные годы. Следует отметить, что резкое сокращение размещенных в фондах электронной библиотеки работ, со второй половины XX века по настоящее время, обусловлено нормами закона об авторском праве, которым руководствуются крупные библиотеки при поставке материалов для электронной библиотеки.

Таблица 2

Распределение публикаций по времени издания

Период	% на сайте	% в тб
1700-1799	15,18	14,24
1800-1899	24,09	24,93
1900-1999	59,68	59,5
2000-2018	1,05	1,33

Анализ востребованности ЭБ ННР показывает, что ее контент представляет интерес не только для российских, но и для зарубежных пользователей. В таблице 3 приведен фрагмент рейтингового списка стран, из которых запрашивались книги, отраженные в ЭБ ННР.

Таблица 3

Страны, наиболее интенсивно пользующиеся ЭБ ННР

Страна	Книговыдача	Страниц
Россия	144573	5060276
Украина	28614	1228989
Беларусь	6114	208723
Казахстан	3181	114877
США	2857	101974
Германия	2701	107504
Болгария	2004	126883
Польша	1707	65480
Италия	1219	51838
Израиль	1008	36345
Азербайджан	901	39964
Армения	892	31501
Молдавия	817	27751
Литва	778	50378
Франция	756	28053

Заключение

Сегодня Электронная библиотека «Научное наследие России» представляет собой интегрированный информационный ресурс, включающий не только биографические сведения о персоналии, печатные и архивные документы (как самих ученых, так и документы, посвященные научным проектам, научным событиям, научным школам и т.д.), но и связанные с ними цифровые коллекции музейных предметов, а также других внешних источников. Средствами электронной библиотеки обеспечена интеграция разнородных материалов, в том числе электронных копий объектов библиотечного, архивного и музейного хранения, представленных в виде текстов, графических образов, аудио-видео объектов.

Пользователи имеют возможность:

– найти электронную копию печатного издания, архивного документа, объекта музей-

ного хранения, связанного с той или иной персоной;
- прочитать полный текст найденной работы;
- познакомиться с цифровыми копиями некторых музейных коллекций;
- посмотреть фото- видео- материалы, представленные на виртуальных выставках, созданных в рамках совместных проектов с государственными естественнонаучными музеями и архивами;
- использовать сервисы, предоставляемые ЭБ НИР на платформах Android и iOS.

Как уже отмечалось, важным направлением развития ЭБ НИР является сотрудничество с Национальной электронной библиотекой в качестве интегратора научно-образовательных информационных ресурсов организаций информационной сферы, подведомственных ФАНО России.

Работа выполнена в рамках Государственного задания № 0065-2018-0405.

The current state of the digital library "Scientific Heritage of Russia"

N.E. Kalenov, S.A. Kirillov, I.N. Sobolevskaya, A.N. Sotnikov

Abstract: The article describes a current status of the "Scientific Heritage of Russia" digital library (DL SHR) at the end of November 2018. The means of the digital library ensures the integration of heterogeneous materials, including digital copies of the library collections, archival and museum storage, presented in the form of texts, graphic images, and audio-video objects. The paper presents basic formation principles of the DL SHR collections. The structure of metadata for each type of objects reflected in the DL SHR is described. The main directions of the digital library development are discussed. Also described are the multimedia resources placed on the EL SHR website - digitized fragments of scientific films and 3D models of museum exhibits. The analysis of the demand for resources of the EL SHR is carried out.

Keywords: electronic library, Scientific heritage of Russia, 3D-model, museum object, virtual exhibition

Литература

1. Н.Е.Каленов, Г.И.Савин, А.Н.Сотников. Электронная библиотека "Научное наследие России". «Информационные ресурсы России», 2009, № 2, 19-20.

2. Н.Е.Каленов, Г.И.Савин, А.Н.Сотников, А.В.Глушановский, С.А.Кириллов, А.И.Малини. Электронная библиотека "Научное наследие России". «Бібліотечний вісник», т. 6 (2009), 40-42.

3. Н.Е.Каленов, Г.И.Савин, А.Н.Сотников. Электронная библиотека "Научное наследие России" как составляющая интеграционных процессов. «Вестник Библиотечной Ассамблеи Евразии», т. 3 (2011), 52-55

4. John Cooper, Andrew Wetherelt, Chiara Zazzaro. From Boatyard to Museum: 3D laser scanning and digital modelling of the Qatar Museums watercraft collection. «International journal of nautical archaeology», v. 47 (2018), № 2, 419-442

5. Н.Е.Каленов, И.Н.Соболевская, А.Н.Сотников. Цифровые музейные коллекции и представление объектов естественно-научного музейного хранения в электронной библиотеке "Научное наследие России". «Научно-техническая информация», Сер. 1 (2016), № 10, 33-38.

6. К.П. Погорелко. Анализ востребованности электронной библиотеки «Научное наследие России». «Информационное обеспечение науки: новые технологии: Сб-к научных трудов», 2015, 191-199.

7. Н.Е.Каленов, И.Н.Соболевская, А.Н.Сотников. О взаимодействии электронной библиотеки "Научное наследие России" с естественно-научными музеями. «Информационные ресурсы России», 2015, № 6, 2-5.

8. Н.Е.Каленов. Интегрированная электронная библиотека "Научное наследие России". «Информационный бюллетень РБА», 2014, № 71, 92-94.

9. М.М.Якшин М.М. WEB-интерфейс системы "Наука России". «Современные технологии в информационном обеспечении науки: Сб. науч. тр. под ред. Н.Е.Каленова», 2003, 47-52.

10. К.П. Погорелко. Динамика использования электронной библиотеки "Научное наследие России". «Информационное обеспечение науки: новые технологии: Сборник научных трудов», 2017, № 3, 192-200.