ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2015
Т. 4, № 2

ISSN 2305-9052

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Учредитель — Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Южно-Уральский государственный университет» (национальный исследовательский университет)

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Математическое и программное обеспечение высокопроизводительных вычислительных систем
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY
SERIES "COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2015
Vol. 4, no. 2

ISSN 2305-9052

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- High performance computing
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Информатика, вычислительная техника и управление

ОПТИМИЗАЦИЯ ОТОБРАЖЕНИЯ НЕОДНОРОДНО ВЗАИМОДЕЙСТВУЮЩИХ MPI ПРОЦЕССОВ НА ВЫЧИСЛИТЕЛЬНУЮ АРХИТЕКТУРУ
В.В. Гетманский, В.С. Чалышев, Д.И. Крыжановский, Е.И. Лексиков 5

СРЕДСТВА ПРОГРАММИРОВАНИЯ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ОСНОВЕ ПЛИС VIRTEX-7 С ИСПОЛЬЗОВАНИЕМ СОФТ-АРХИТЕКТУР
И.И. Левин, А.И. Дордопуло, В.Б. Коваленко, В.А. Гудков, А.А. Гуленок 20

ОБЕСПЕЧЕНИЕ ОПЕРАТИВНОГО КОНТРОЛЯ И ЭФФЕКТИВНОЙ АВТОНОМНОЙ РАБОТЫ СУПЕРКОМПЬЮТЕРНОГО КОМПЛЕКСА МГУ
А.С. Антонов, Вад.В. Воеводин, А.А. Даугель-Дауге, С.А. Жуматий, Д.А. Никитенко, С.И. Соболев, К.С. Стефанов, П.А. Швец 33

ФОРМИРОВАНИЕ И ПЛАНИРОВАНИЕ ПАКЕТОВ ЗАДАНИЙ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДАХ
В.В. Топорков, Д.М. Емельянов, П.А. Потехин 44

ИНСТРУМЕНТАЛЬНАЯ ПОДДЕРЖКА ФОРМАЛЬНОЙ ВЕРИФИКАЦИИ ПРОГРАММ, НАПИСАННЫХ НА ЯЗЫКЕ ФУНКЦИОНАЛЬНО-ПОТОКОВОГО ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ
М.С. Ушакова, А.И. Легалов 58

Вычислительная математика

МАСШТАБИРУЕМЫЕ АЛГОРИТМЫ ЦЕЛОЧИСЛЕННОЙ АРИФМЕТИКИ И ОРГАНИЗАЦИЯ ПОДДЕРЖКИ РАЦИОНАЛЬНЫХ ВЫЧИСЛЕНИЙ В ГЕТЕРОГЕННЫХ СРЕДАХ
В.А. Голодов, А.В. Панюков 71

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССА ФИЛЬТРАЦИИ ФЛЮИДОВ В ПОРИСТЫХ СРЕДАХ
Н. Равшанов, Н.М. Курбонов 89

Contents

Computer Science, Engineering and Control

OPTIMIZING PROCESSES MAPPING FOR TASKS WITH NON-UNIFORM DATA EXCHANGE RUN ON CLUSTER WITH DIFFERENT INTERCONNECTS
V.V. Getmanskiy, V.S. Chalyshev, D.I. Kryzhanovskiy, E.I. Leksikov 5

PROGRAMMING TOOLS FOR RECONFIGURABLE COMPUTER SYSTEMS BASED ON VIRTEX-7 FPGAS WITH USING SOFT-ARCHITECTURES
I.I. Levin, A.I. Dordopulo, V.B. Kovalenko, V.A. Gudkov, A.A. Gulenok 20

SECURING OF ACTIVE CONTROL AND EFFICIENT AUTONOMOUS OPERATING OF MSU SUPERCOMPUTING CENTER
A.S. Antonov, Vad.V. Voevodin, A.A. Daugel-Dauge, S.A. Zhumatiy, D.A. Nikitenko, S.I. Sobolev, K.S. Stefanov, P.A. Shvets 33

JOB BATCH GENERATION AND SCHEDULING IN DISTRIBUTED COMPUTING ENVIRONMENT
V.V. Toporkov, D.M. Yemelyanov, P.A. Potekhin 44

A TOOLKIT FOR SUPPORTING FORMAL VERIFICATION OF PROGRAMS IN THE FUNCTIONAL DATA-FLOW PARALLEL PROGRAMMING LANGUAGE
M.S. Ushakova, A.I. Legalov 58

Computational Mathematics

SCALABLE ALGORITHMS FOR THE INTEGER ARITHMETICS AND RATIONAL CALULATIONS IN HETEROGENEOUS COMPUTATION ENVIRONMENT
V.A. Golodov, A.V. Panyukov 71

COMPUTER MODELING OF PROCESSES OF FLUID FILTRATION IN POROUS MEDIA
N. Ravshanov, N. Kurbonov 89

УДК 004.272.26 DOI: 10.14529/cmse150201

ОПТИМИЗАЦИЯ ОТОБРАЖЕНИЯ НЕОДНОРОДНО ВЗАИМОДЕЙСТВУЮЩИХ MPI ПРОЦЕССОВ НА ВЫЧИСЛИТЕЛЬНУЮ АРХИТЕКТУРУ[1]

В.В. Гетманский, В.С. Чалышев, Д.И. Крыжановский, Е.И. Лексиков

Разработан метод отображения на кластерную архитектуру неоднородно взаимодействующих параллельных процессов в вычислительном приложении, использующем MPI. Метод предназначен для сокращения задержек при синхронизации за счет назначения наиболее интенсивно взаимодействующих процессов, на вычислительные ядра с наиболее быстрым интерконнектом. Метод использует представление вычислительной задачи и архитектуры кластера в виде взвешенного графа. Разработан эвристический алгоритм, дающий за приемлемое время результат отображения номеров процессов на номера вычислительных ядер кластера. На примере хорошо масштабируемого вычислительного пакета получено ускорение вычислений на 17–20 % в результате оптимизации отображения для тестов от 300 до 4800 процессов.

Ключевые слова: отображение задач, кластер, графы задачи и системы, MPI.

Введение

Проблема отображения параллельной программы на архитектуру вычислительной системы с целью уменьшения времени обмена данными рассмотрена в ряде работ отечественных [1, 2] и зарубежных [3–5] авторов. В настоящей работе рассмотрена разработка и тестирование метода отображения графа задачи на граф вычислительной системы. Вершины графа задачи соответствуют параллельным процессам, а ребра графа задачи взвешены объемами пересылаемых между процессами данных. Вершины соединены ребрами, только если соответствующие им процессы обмениваются данными. В научных вычислительных пакетах граф задачи имеет произвольную структуру. Пример такого графа для вычислительного пакета ZeusMP, запущенного на 2 узлах кластера, показан на рис. 1. Ширина линий, обозначающих ребра, для наглядности пропорциональна весу ребра. Данное обозначение будет использовано на остальных рисунках.

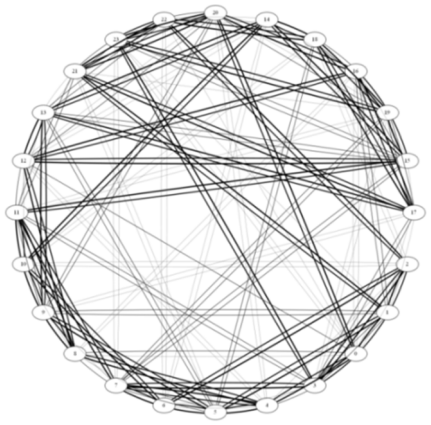


Рис. 1. Граф задачи ZeusMP

[1] Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии-2015».

Граф системы представляет собой топологию кластера с учетом различной производительности каналов связи (интерконнекта). Ребра графа системы взвешены постоянными коэффициентами, представляющими соотношение времени передачи данных внутри узла кластера и времени передачи данных между узлами.

Современные кластеры состоят из многосокетных узлов, как правило, содержащих сопроцессоры GPGPU, FPGA или MIC. Каналы связи также работают с различной скоростью (общая память, InfiniBand, Ethernet). Таким образом, граф кластера тоже может иметь неоднородную структуру из-за сложной конфигурации узлов кластера (рис. 2).

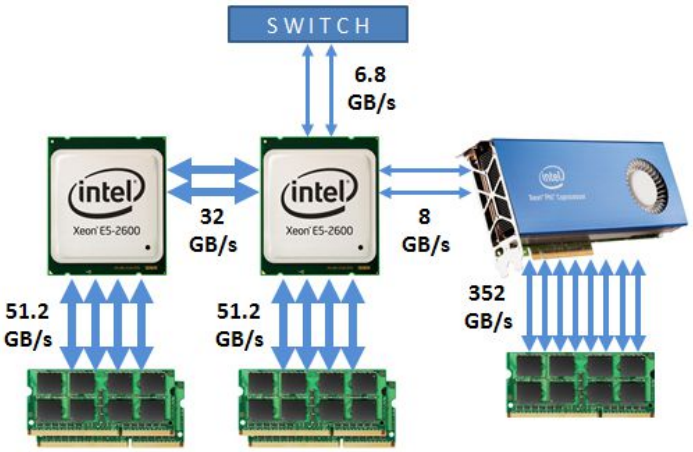


Рис. 2. Конфигурация узлов гетерогенного кластера

Основное проверяемое предположение в настоящей работе состоит в том, что обмен данными можно ускорить, если распределить процессы на кластере так, что наиболее интенсивный обмен (ребра графа задачи с максимальным весом) будет происходить по наиболее быстрому каналу связи (ребра графа системы с максимальным весом).

Статья организована следующим образом. В первом разделе рассмотрена постановка задачи отображения графа параллельной программы на граф вычислительной системы. Во втором разделе описаны два реализованных алгоритма решения этой задачи. В третьем разделе описан синтетический тест параллельной программы с сильно неоднородным обменом данными между процессами. В четвертом разделе описаны вычислительные эксперименты на кластере с использованием синтетического теста и пакета QBox, проведен анализ полученных результатов. В заключении перечислены основные результаты исследования и предложения по усовершенствованию метода.

1. Постановка задачи

Обозначим граф задачи как

$$G_1(P, L),\ L_i = (n_i, d_i),\ i = 1, \dots, K, \tag{1}$$

где P — набор вершин графа (процессы), L — набор связей (взаимодействие процессов), n_i — частота обмена данными, d_i — объем обмениваемых данных, K — число связей.

Обозначим граф системы как

$$G_2(V, D),\ D_j = \left(l_j, b_j\right),\ j = 1, \dots, S, \tag{2}$$

где V — множество вершин (соответствующих вычислительным ядрам центральных процессоров или сопроцессоров), D — множество ребер (каналы передачи данных), l_j — задержка, b_i — пропускная способность, S — общее число каналов передачи данных. В первом приближении в графе системы передача данных возможна между любой парой вершин, т.е. G_2 — полносвязный граф и $K < S$.

Искомое отображение можно записать как

$$G_1 \to G_2\ (P \to V)\colon L_i \to D_j,\ F(D,L) \to \min. \tag{3}$$

Минимизируемая целевая функция задает временную задержку на пересылки и имеет вид

$$F(D,L) = \sum_{k=1}^{N} T_k\ ,\ T_k\left(D_i, L_j\right) = l_i \cdot n_j + \frac{d_j}{b_i},\ F \to \min. \tag{4}$$

Для разработанного прототипа используется упрощенная формулировка. Все ребра графа задачи взвешены объемом обмениваемых данных между процессами, которым соответствуют вершины ребра. Граф системы взвешен коэффициентом задержки канала связи r_i. Для задания того, что сетевое взаимодействие заведомо медленнее взаимодействия через общую память, r_i определено как

$$r_i = \begin{cases} 1, \text{если обмен через общую память} \\ 2, \text{если обмен через } InfiniBand \end{cases} \tag{5}$$

Целевая функция вычисляется как сумма произведений весов ребер:

$$\tilde{F}\left(\widetilde{D}, L\right) = \sum_{k=1}^{N} \tilde{T}_k\ ,\ \tilde{T}_k\left(\widetilde{D}_i, L_j\right) = r_i \cdot d_j,\ \widetilde{D}_i = (r_i),\ \tilde{F} \to \min. \tag{6}$$

Таким образом, для решения задачи отображения необходимо построить граф задачи и граф системы, разработать алгоритм поиска отображения графа задачи на граф системы с целью минимизации суммы произведений весов ребер.

2. Алгоритм отображения

Разработаны два алгоритма отображения: первый использует переборную стратегию, второй — «жадную» стратегию. Для графов, содержащих более 10 вершин, полный перебор выполняется неоправданно долго, поэтому практически применим только второй алгоритм, который дает либо оптимальное решение, либо близкое к оптимальному. Экспериментально установлено, что значение целевой функции становится не хуже исходного после работы алгоритма для проведенных тестов.

2.1. Алгоритм полного перебора

Алгоритм ищет решение с помощью полного перебора с возвратом. Вычислительная сложность алгоритма $O\left(\binom{N}{M} \cdot M!\right)$, где N — число вершин в графе системы, M — число вершин в графе задачи. Таким образом, на практике перебор можно использовать только для графов системы, содержащих до 15 вершин, и для графов задачи, содержащих до 10 вершин.

2.2. Алгоритм с «жадной» стратегией

Реализация основана на похожем подходе, описанном в работе [5]. Основное отличие предлагаемой реализации в том, что отсутствует жесткое требование равенства вершин в графе задачи и графе системы.

Общий алгоритм:

1. Поиск первого приближения. В разработанном прототипе используется отображение -й вершины P_i графа кластера на j-ю вершину V_j графа системы.
2. Итеративная процедура улучшения решения:
   2.1. Перестановка отображения двух вершин. Смена отображения $P_i \to V_j,\ P_k \to V_l$, на отображение $P_i \to V_l,\ P_k \to V_j$.
   2.2. Отображение на новую вершину. Смена отображения $P_i \to P_j$ на $P_i \to P_k$.

Вычислительная сложность алгоритма $O\big(IE(M+N)\big)$, где I — число итераций, E — число ребер в графе задачи M — число вершин в графе задачи, N — число вершин в графе системы.

В общем случае оценить сходимость алгоритма к точному решению сложно. Предположим, что все ребра в графе задачи 2 типов и имеют кратные веса a и $a \cdot k$, все ребра в графе системы тоже 2 типов с весами b и $b \cdot r$. В этом случае верхняя граница сходимости $Mkr/(r-1)$, т.е. сходимость алгоритма для большинства случаев быстрее этого значения.

3. Описание синтетического теста

Синтетический тест моделирует обмен данными различного объема и с различной интенсивностью в итеративной процедуре, выполняющейся в MPI программе. Тест представляет собой наиболее оптимизируемый частный случай входных данных для алгоритма оптимизации. В тесте используется обмен данными двух типов: обмен внутри группы тесно взаимодействующих процессов с интенсивным взаимодействием и обмен между группами процессов.

Параметры синтетического теста: N_{bl} — число групп процессов, N_{perbl} — число процессов в группе, D_{bl} — объем данных, пересылаемых между группами, D — объем данных, пересылаемых внутри группы, N_{skip} — число пропускаемых итераций.

В каждой группе каждый процесс взаимодействует с каждым процессом группы (полно связанный подграф). Взаимодействие между группами включает только обмен данными между последним процессом группы i и первым процессом группы с номером $i+1$. Например, для значений параметров $N_{bl} = 2$, $N_{perbl} = 6$ взаимодействие между 11-м процессом, входящим в первую группу, и первым процессом, входящим во вторую группу, показано тонкими ребрами на графе задачи (рис. 3).

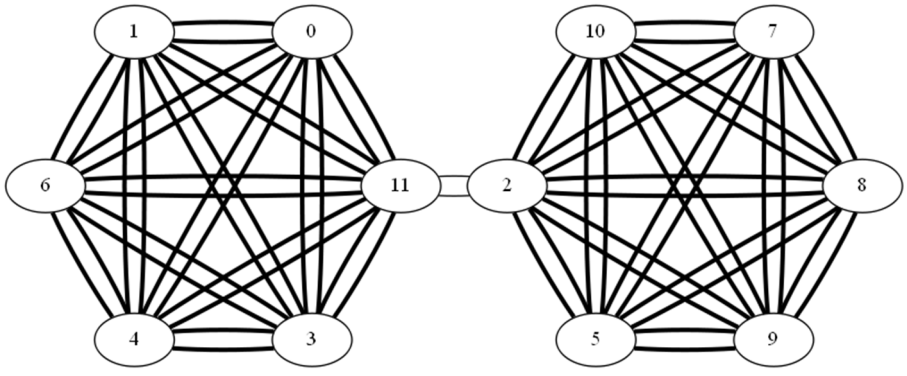


Рис. 3. Пример графа для 2 блоков с 6 процессами в каждом

Взаимодействие между группами происходит не на каждой итерации, поэтому задается пропуск итераций. Номера процессов перемешаны случайным образом (с постоянным параметром генератора случайных чисел для воспроизводимости коммуникационного взаимодействия).

Смещения и длины в линейном массиве с данными строятся на каждой итерации для операций MPI_Alltoallv коллективного обмена MPI-библиотеки. Эти массивы дополняются на каждой $1 + N_{skip}$ итерации данными для обмена между группами. Например, в случае $N_{skip} = 2$ массивы для графа задачи, показанном на рис. 3, имеют вид, представленный в табл. 1.

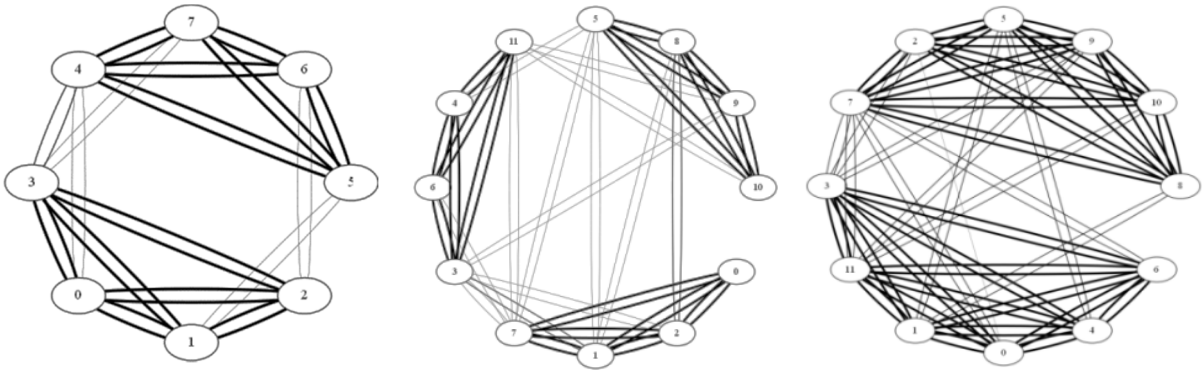
Таблица 1

Смещения и длины для блоков обмениваемых данных MPI-процессов

Процесс назначения		**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
итерации $3k$, $1+3k$, процесс 2	смещение	0	0	0	0	0	0	D	D	$2D$	$3D$	$4D$	$5D$
	длина	0	0	0	0	0	D	0	D	D	D	D	0
итерации $3k$, $1+3k$, процесс 11	смещение	0	D	$2D$	$2D$	$3D$	$4D$	$4D$	$5D$	$5D$	$5D$	$5D$	$5D$
	длина	D	D	0	D	D	0	D	0	0	0	0	0
итерация $2+3k$, процесс 2	смещение	0	0	0	0	0	0	D	D	$2D$	$3D$	$4D$	$5D$
	длина	0	0	0	0	0	D	0	D	D	D	D	D_{bl}
итерация $2+3k$, процесс 11	смещение	0	D	$2D$	$2D + D_{bl}$	$3D + D_{bl}$	$4D + D_{bl}$	$4D + D_{bl}$	$5D + D_{bl}$	$5D + D_{bl}$	$5D + D_{bl}$	$5D + D_{bl}$	$5D + D_{bl}$
	длина	D	D	D_{bl}	D	D	0	D	0	0	0	0	0

В табл. 1 показано, что массив смещений и длин обмениваемых блоков для процессов, участвующих в обмене между группами (процессы 2 и 11), на каждой итерации с номером $0 + 3k$ и $1 + 3k$ содержит 5 обмениваемых блоков длиной D, а на каждой итерации с номером $2 + 3k$ содержит дополнительный блок обмена между группами длиной D_{bl}. У остальных процессов массив длин и смещений не меняется.

Пример графов задач, полученных с помощью запусков синтетического теста с различными параметрами, показан на рис. 4. Тонкими линиями показаны системные взаимодействия процессов при синхронизации, полученные при сборе статистики.



а б в

Рис. 4. Пример графов задачи, построенных по синтетическим тестам: а) 2 блока с 4 процессами на блок, б) 3 блока с 4 процессами на блок, в) 2 блока с 6 процессами на блок

Таким образом, с помощью синтетического теста моделируется неоднородное взаимодействие заданного количества групп параллельных процессов с возможностью менять объем данных для обмена и интенсивность обмена данными между группами. При задании числа процессов, соответствующего числу ядер на один узел кластера и перенумерации процессов случайным образом, собирается статистика обмена данными, которая подается на вход алгоритму оптимизации.

4. Результаты тестирования

4.1. Запуск тестов

Запуск тестов проводился с использованием программы запуска Intel® MPI Library [6]. Передача набора параметров, необходимых для назначения процессов на вычислительные ядра, реализована с помощью конфигурационного файла, передаваемого по ключу командной строки «--configfile». Строка запуска имеет вид: «mpirun --configfile config.txt».

Разработан инструментарий для генерации файла конфигурации. В качестве входного файла используется список хостов кластера. Первый шаг — запуск непродолжительного теста для сбора статистики коммуникаций MPI-процессов. Входные данные для генерации конфигурационного файла (config.txt) — это список хостов с именами узлов машин на кластере и числом ядер на каждом узле (hostlist.txt).

Например, для кластера из 4 узлов с 4-ядерными процессорами содержимое файла hostlist.txt имеет вид в листинге на рис. 5:

```
line 01: node1 4
line 02: node2 4
line 03: node3 4
line 04: node4 4
```

Рис. 5. Пример файла hostlist.txt для 4 узлов

Здесь и в остальных листингах начало строки обозначено как line xx, xx — номер строки. Генерируемый файл конфигурации config.txt для сбора статистики имеет вид, приведенный в листинге на рис. 6.

```
line 01: -env I_MPI_STATS 4 -env I_MPI_STATS_FILE
/path/to/stat.txt -n 4 -host node1 /path/to/application
line 02: -env I_MPI_STATS 4 -env I_MPI_STATS_FILE
/path/to/stat.txt -n 4 -host node2 /path/to/application
line 03: -env I_MPI_STATS 4 -env I_MPI_STATS_FILE
/path/to/stat.txt -n 4 -host node3 /path/to/application
line 04: -env I_MPI_STATS 4 -env I_MPI_STATS_FILE
/path/to/stat.txt -n 4 -host node 4 /path/to/application
```

Рис. 6. Пример файла config.txt для сбора статистики на четырех узлах

Запуск MPI программы с использованием файла конфигурации config.txt указанного формата приводит к формированию средствами библиотеки Intel® MPI Library файла статистики обменов stats.txt. Программе оптимизации подаются на вход файлы stats.txt и hostlist.txt, на выходе генерируется файл config.txt с оптимизированным отображением процессов на узлы кластера. Для рассматриваемого примера содержимое файла config.txt приведено в листинге на рис. 7.

```
line 01:-n 2 -host node1 /path/to/application
line 02:-n 2 -host node4 /path/to/application
line 03:-n 1 -host node2 /path/to/application
line 04:-n 2 -host node1 /path/to/application
line 05:-n 3 -host node2 /path/to/application
line 06:-n 4 -host node3 /path/to/application
line 07:-n 2 -host node4 /path/to/application
```

Рис. 7. Пример файла config.txt для четырех узлов после оптимизации

Оптимизированный файл конфигурации используется для запуска длительного теста. Методология тестирования состоит в замере времени выполнения длительного теста до и после оптимизации размещения процессов на кластере. Показатель эффективности — ускорение, вычисляемое как $S = T/T_{opt}$, где T — время работы теста до оптимизации, T_{opt} — после оптимизации.

Для запуска тестов использован суперкомпьютер «Торнадо ЮУрГУ» [7]. Конфигурация выделенных узлов: двухсокетные узлы с процессорами Intel® Xeon® X5680 (2 процессора по 6 ядер), Intel® Xeon Phi™ SE10X (61 ядро по 1,1 ГГц), 2 Гб RAM, InfiniBand QDR, топология сети — толстое дерево. При тестировании были поставлены задачи исследования синтетического теста до и после оптимизации отображения процессов на ядра двух сопроцессоров и центральных процессоров кластера при различной размерности блоков. На центральных процессорах кластера также было необходимо провести запуск масштабируемого на большое число узлов вычислительного приложения с неоднородным взаимодействием процессов, в качестве которого был выбран CORAL QBox [8].

4.2. Синтетический тест для двух узлов кластера с сопроцессорами

Конфигурация для тестирования состоит из двух узлов с установленными сопроцессорами Intel® Xeon Phi™, содержащими по 60 ядер, доступных для вычислений. Такая конфигурация была выбрана для оценки влияния работы MPI тестов, характеризующихся слабой связанностью по данным групп параллельных процессов с интенсивным

обменом, что имеет место, например, при моделировании связанных физических задач. Данный тест был необходим для оценки влияния сетевого взаимодействия в случае интенсивного обмена групп с большим числом параллельных процессов, поэтому в графе системы было только 2 типа весов ребер: связи между ядрами и сетевой интерконнект, взаимодействие ядер сопроцессора принималось приближенно равнозначным. Результаты тестирования приведены в табл. 2. Параметры теста: $D = 60$ Мб, $D_{bl} = 60$ Кб, $N_{skip} = 0$.

Таблица 2

Результаты синтетического теста для 2 узлов с сопроцессорами

N_{bl}	N_{perbl}	T, с	T_{opt}, с	S
2	60	660,847	84,125	7,855
3	40	440,512	194,514	2,264
4	30	320,938	153,668	2,088
5	24	261,291	86,774	3,011
6	20	221,371	35,210	6,287
8	15	165,581	34,700	4,771
15	8	93,015	39,100	2,378
20	6	68,347	30,007	2,277

Каждый тестовый запуск проводился в Native режиме (код выполняется только на сопроцессорах) и использовал все вычислительные ядра сопроцессора: $N_{bl} \cdot N_{perbl} = 120$. Содержимое сгенерированного конфигурационного файла для сбора статистики приведено в листинге на рис. 8.

```
line 01:-env I_MPI_STATS 4 -env I_MPI_STATS_FILE
/path/to/stat.txt -n 60 -host mic0 /path/to/application
line 02:-env I_MPI_STATS 4 -env I_MPI_STATS_FILE
/path/to/stat.txt -n 60 -host mic1 /path/to/application
```

Рис. 8. Пример файла config.txt для сбора статистики на 2 сопроцессорах

Фрагмент файла конфигурации config.txt после оптимизации на основе собранной статистики приведен в листинге на рис. 9.

```
line 01:-n 5 -host mic0 /home/test/MPICommunication 20 6
line 02:-n 1 -host mic1 /home/test/MPICommunication 20 6
line 03:-n 4 -host mic0 /home/test/MPICommunication 20 6
line 04:-n 2 -host mic1 /home/test/MPICommunication 20 6
line 05:-n 1 -host mic0 /home/test/MPICommunication 20 6
line 06:-n 3 -host mic1 /home/test/MPICommunication 20 6
. . .
```

Рис. 9. Фрагмент файла config.txt для 2 сопроцессоров после оптимизации

Файл для сбора статистики состоит из 2 строк (по числу узлов), а сгенерированный после оптимизации конфигурационный файл для запуска содержит 55 строк, так как номера процессов изначально перемешаны случайным образом.

В результате отображения параллельных процессов на ядра получено ускорение выполнения тестов в несколько раз, что свидетельствует о существенной зависимости расположения параллельных процессов на вычислительных ядрах и целесообразности использования алгоритма отображения.

4.3. Синтетический тест для 10 узлов с InfiniBand интерконнектом

В качестве тестового стенда использовался кластер с 10 двухсокетными узлами и 6-ядерными процессорами Intel® Xeon®. Тестирование проводилось для фиксированного объема данных, но с разным числом итераций (N_{iter}), в соответствии с параметрами, приведенными в табл. 3.

Таблица 3

Результаты синтетического теста для 10 узлов кластера

N_{iter}	D_{bl}, Кб	D, Мб	T, с	T_{opt}, с	S
100	8	8	1755	880	1,99
1000	0,8	0,8	23,21	9,56	2,42
10000	0,08	0,08	5,96	1,01	5,9

Для обмена внутри групп общий объем передаваемых данных между парой процессов составляет примерно 800 Мб, для обмена между группами — 800 Кб. В тесте сгенерированы 10 групп по 12 процессов в каждой (по числу ядер каждого узла кластера). Фрагмент графа задачи приведен на рис. 10.

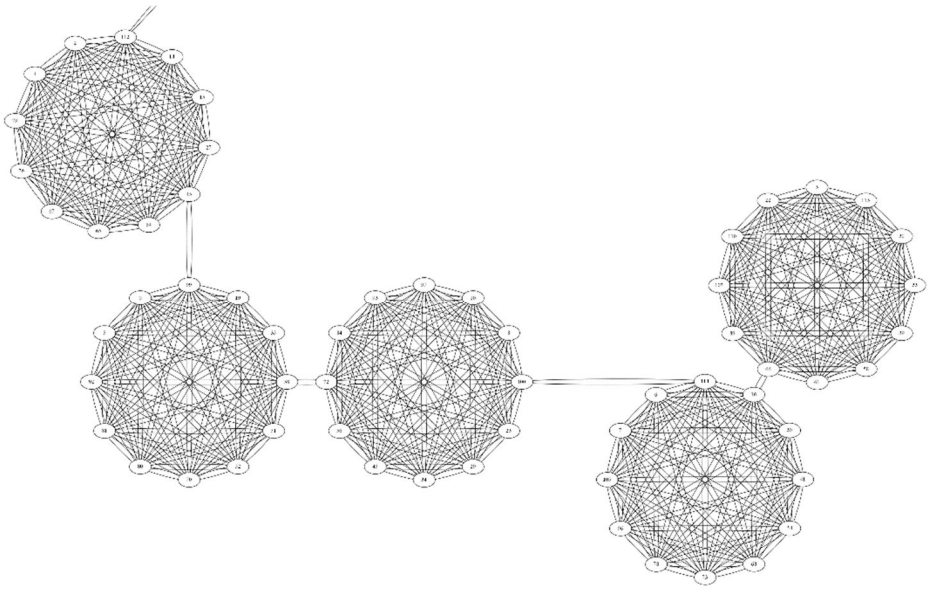


Рис. 10. Фрагмент графа задачи для синтетического теста (5 групп по 12 процессов)

Результаты свидетельствуют о том, что более частые обмены небольших порций данных с помощью наиболее эффективных в данном случае операций коллективного обмена библиотеки MPI оптимизируются лучше, чем менее интенсивные обмены больших порций данных. Частые обмены небольших порций характерны для вычислительных приложений, использующих, например, декомпозицию расчетной области или параллельную реализацию явных методов интегрирования систем дифференциальных уравнений. Для некоторых задач при неполной загрузке узлов (тест с менее чем 12 процессами на узел) выполнение параллельной программы может оказаться быстрее при использовании InfiniBand, чем при использовании общей памяти, например, для описанного в [9] случая. Для рассматриваемой задачи эксперименты показали проявление такого же эффекта: в случае неполной загрузки узлов (по 4–8 ядер из 12) тест срабатывал

медленнее при переназначении наиболее интенсивного обмена на общую память, чем при переназначении интенсивного обмена на InfiniBand. Разница времени выполнении составляла не более 5 %.

4.4. Результаты тестирования на удаленном кластере пакета для моделирования молекулярной динамики CORAL QBox

CORAL QBox — это пакет моделирования молекулярной динамики для определения свойств материала, использующий теорию функционала плотности. Метод имеет вычислительную сложность $O(N^3)$, где N — общее число валентных электронов в системе.

Тесты собраны и запущены для различного числа узлов удаленного кластера вплоть до 400. В тестах задействовано разное число узлов (N_{nodes}), при максимальной загрузке узлов число параллельных процессов составляет $12N_{nodes}$.

Основной параметр, влияющий на производительность QBOX, — это параметр максимального числа строк в $2D$ сетке параллельных процессов ($nrowmax$). Количество строк и столбцов в этой сетке подбирается так, что $nrowmax$ уменьшается, пока число параллельных процессов (N) не будет делиться на $nrowmax$, после чего число столбцов принимается равным $N/nrowmax$. Эксперименты показали, что максимальная эффективность имеет место при $nrowmax = N_{nodes}$, соответственно число столбцов составляет при этом 12. Число атомов при генерации входных данных было выбрано как максимальное число атомов, при котором задача помещается в оперативную память, значение которой ограничено 2 Гб на узел.

Граф задачи для пакета QBox имеет общий вид из-за неоднородного обмена данными. По статистике, собранной при запуске на двух узлах с $nrowmax = 2$, построен граф задачи, на котором можно выделить 2 группы по 12 процессов с тесным взаимодействием (рис. 11), причем порядок соответствующих вершин графа соответствует номерам ядер процессоров.

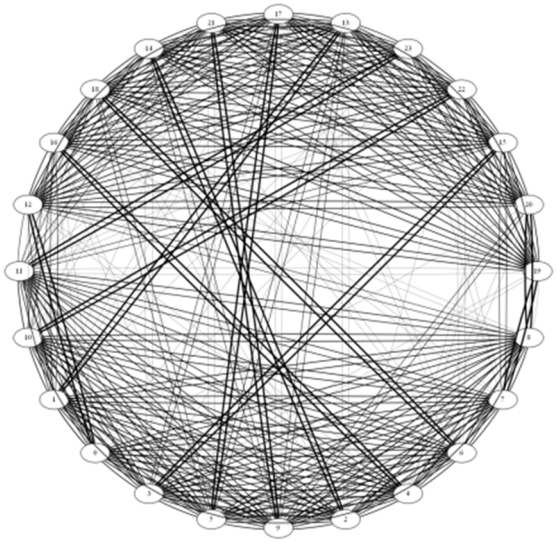


Рис. 11. Две группы из 12 процессов в графе задачи для пакета QBox

Каждой подгруппе соответствует полносвязный подграф графа задачи, в то время как между подграфами набор связей неполный и их суммарный вес, соответствующий объему переданных данных сопоставим по величине с суммарным весом ребер, инци-

дентных ребрам одной вершины в подгруппе (для каждой вершины средний объем переданных данных составил 185 Мб, а общий объем переданных данных между подграфами — 421 Мб).

Тесты в пакете QBox были подобраны для наилучшего времени выполнения при размерностях сеток 25×12, 50×12, 100×12, 150×12, 200×12, 400×12 для 25–400 узлов соответственно (число процессов и ядер 300–4800). Соотношение весов ребер графа системы составило 2:1 (общая память по отношению к InfiniBand). Результаты тестирования показаны в табл. 4.

Таблица 4

Результаты тестирования QBox

Число узлов	**Размерность сетки**	T, с	T_{opt}, с	S
25	25×12	773,5	660,5	1,171083
50	50×12	398,3	336,4	1,184007
100	100×12	216,8	180,2	1,203108
150	150×12	180,5	155,1	1,163765
200	200×12	172,4	148,3	1,162508
400	400×12	170	145,2	1,170799

На основании собранных данных видно, что оптимизация обменов данными дает практически постоянное улучшение производительности на 17–20 %. Максимум приходится на 100 узлов (1200 параллельных процессов). С этого же числа узлов падает масштабируемость задачи, так как размерность задачи ограничена оперативной памятью и задачу большей размерности для сохранения масштабируемости рассчитать при такой конфигурации кластера нельзя. Время выполнения рассматриваемой задачи в пакете QBox после 100 узлов сходится к постоянному значению (рис. 12). Тесты на большем числе процессов (до 4800 процессов на 400 узлах) показывают примерно одинаковый результат как по времени выполнения, так и по ускорению за счет оптимизации отображения процессов.

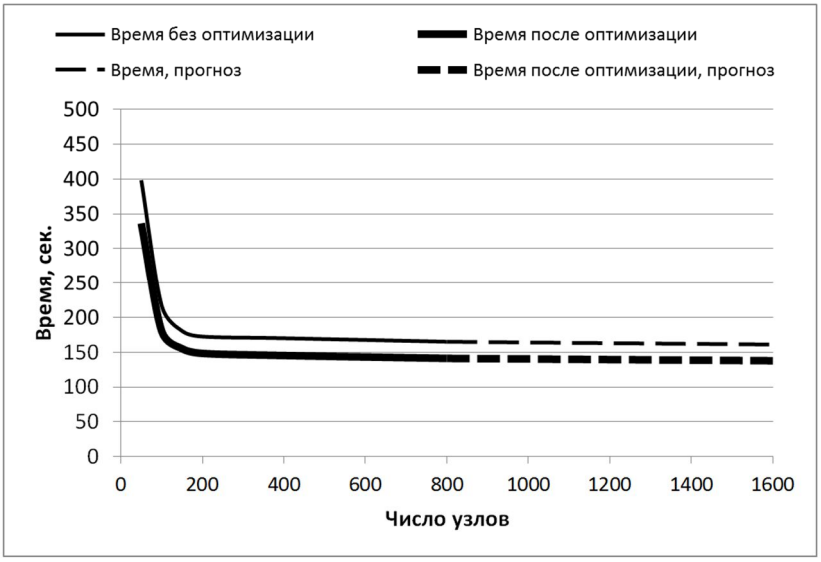


Рис. 12. Время выполнения QBox до и после оптимизации

В рассматриваемом случае сильного масштабирования время пересылок на несколько порядков меньше по сравнению со временем расчета одной итерации, поэтому, экстраполируя данные на большее число процессов, можно сделать предположение о том, что время выполнения MPI-программы будет сохраняться постоянным, так же, как и эффект от оптимизации, для достаточно большого числа процессов.

При существенном увеличении числа процессов время расчета одной итерации будет приближаться ко времени выполнения пересылки данных, что приведет к увеличению влияния пересылок. При этом время выполнения параллельной программы будет расти, а эффект от оптимизации, предположительно, будет усиливаться из-за более существенного влияния времени пересылок на общее время выполнения. Последний сценарий не используется на практике, так как предполагает неэффективное использование ресурсов.

Рассмотренный вычислительный пакет CORAL QBox хорошо масштабируется до 100 узлов. На большем числе узлов эффективность параллельного расчета снижается, так как размерность задачи недостаточно большая и время обмена данными сопоставимо с расчетным временем. Для задач с большей размерностью пакет масштабируется лучше (анализ проведен в [8]), возможно получение более хороших результатов по ускорению за счет оптимизации обмена данными.

Заключение

Предложенный подход обеспечивает улучшение производительности параллельного расчета в случае, если граф задачи MPI-приложения имеет неоднородное распределение весов ребер и выраженные подграфы, вершины в которых сильно связаны между собой, а количество этих вершин близко к количеству ядер на одном узле кластера.

Положительный эффект ускорения в 2–7 раз получен для синтетического теста при тестировании на узлах с сопроцессорами Intel® Xeon Phi™ и процессорами Intel® Xeon®. Основные факторы, влияющие на эффективность метода: неоднородность взаимодействия процессов, частота обмена данными и объемы пересылаемых данных.

Положительный эффект ускорения открытого пакета CORAL QBox на 17–20 % получен предложенным методом для числа процессов до 4800.

Предложенный метод не дает ощутимого эффекта в случае плохо масштабируемого приложения, задач с равномерным обменом между параллельными процессами и задач с малой интенсивностью обменов.

Экспериментально установлено, что использование подхода при неполной загрузке узлов кластера дает отрицательный эффект из-за сопоставимой в этом случае скорости передачи данных через интерконнект между узлами кластера и через общую память на каждом узле.

Метод отображения задачи можно усовершенствовать, используя физические параметры интерконнекта (пропускную способность и задержки канала связи), а также учитывая полную топологию кластера.

Литература

1. Копысов, С.П. Методы привязки параллельных процессов и потоков к многоядерным узлам вычислительных систем / С.П. Копысов, А.К. Новиков, Л.Е. Тонков

и др. // Вестн. Удмуртск. ун-та. Матем. Мех. Компьют. науки. — 2010. — Вып. 1. — С. 123–132.

2. Курносов, М.Г. Назначение ветвей параллельной программы на процессорные ядра распределенной вычислительной системы / М.Г. Курносов // Материалы Межд. научно-технической конференции «Многопроцессорные вычислительные и управляющие системы» (пос. Дивноморское, Геленджик, 2007). — Таганрог: ТТИ ЮФУ, 2007. — Т. 1. — С. 227–231.
3. Karlsson, C. Optimizing Process-to-Core Mappings for Application Level Multi-dimensional MPI Communications / C. Karlsson, T. Davies, Z. Chen // Cluster Computing (CLUSTER), 2012 IEEE International Conf. Proceedings (Beijing, China, September, 24–28, 2012). — Beijing, 2012. — P. 486–494. DOI: 10.1109/cluster.2012.47
4. Zhang, J. Process Mapping for MPI Collective Communications / J. Zhang, J. Zhai, W. Chen, et al. // Lecture Notes in Computer Science. — 2009. — Vol. 5704. — P. 81–92. DOI: 10.1007/978-3-642-03869-3_11
5. Chen, H. MPIPP: an Automatic Profile-Guided Parallel Process Placement Toolset for SMP Clusters and Multiclusters / H. Chen, W. Chen, J. Huang, et al. // ICS'06 Proceedings of the 20th annual international conference on Supercomputing (Queensland, Australia, June, 28 – July, 01, 2006). — Queensland, 2006. — P. 353–360. DOI: 10.1145/1183401.1183451
6. Intel® MPI Library Reference Manual. URL: http://software.intel.com/sites/products/documentation/hpc/ics/impi/41/lin/Reference_Manual/index.htm (дата обращения: 20.12.2014).
7. Larsson, P. Shared Memory Communication vs. Infiniband / P. Larsson. URL: http://www.nsc.liu.se/~pla/blog/2013/09/12/smp-vs-infiniband (дата обращения: 20.12.2014).
8. Gygi, F. Large-Scale First-Principles Molecular Dynamics Simulations on the BlueGene/L Platform using the Qbox Code / F. Gygi, R.K. Yates, J. Lorenz, et al. // Proceedings of the ACM/IEEE SC 2005 Conference (Seattle, WA, USA, November, 12–18, 2005). — Seattle, 2005. — 24 p. DOI: 10.1109/sc.2005.40
9. Суперкомпьютер «Торнадо ЮУрГУ». URL: http://supercomputer.susu.ac.ru/computers/tornado (дата обращения: 20.12.2014).

Гетманский Виктор Викторович, к.т.н., младший научный сотрудник кафедры высшей математики, Волгоградский государственный технический университет (Волгоград, Российская Федерация), victor.getmanski@gmail.com.

Чалышев Владимир Сергеевич, ООО «Сингулярис Лаб» (Волгоград, Российская Федерация), cmdsoft@gmail.com.

Крыжановский Дмитрий Иванович, к.т.н., ООО «Сингулярис Лаб» (Волгоград, Российская Федерация), dmitry.kryzhanovsky@singularis-lab.com.

Лексиков Евгений Иванович, Intel (Нижний Новгород, Российская Федерация), evgeny.leksikov@intel.com.

Поступила в редакцию 19 марта 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 2, pp. 5–19

DOI: 10.14529/cmse150201

OPTIMIZING PROCESSES MAPPING FOR TASKS WITH NON-UNIFORM DATA EXCHANGE RUN ON CLUSTER WITH DIFFERENT INTERCONNECTS

V.V. Getmanskiy, Volgograd State Technical University (Volgograd, Russian Federation) victor.getmanski@gmail.com,
V.S. Chalyshev, Singularis Lab Ltd. (Volgograd, Russian Federation) cmdsoft@gmail.com,
D.I. Kryzhanovskiy, Singularis Lab Ltd. (Volgograd, Russian Federation) dmitry.kryzhanovsky@singularis-lab.com,
E.I. Leksikov, Intel (Nizhniy Novgorod, Russian Federation) evgeny.leksikov@intel.com

The problem of mapping the parallel task to the nodes of computing cluster is considered. MPI software with non-uniform communication and heterogeneous interconnect of computing cluster require to appropriate parallel processes mapping for optimization of data exchange. The graph mapping algorithm is developed. It uses parallel program representation as a task graph and cluster topology representation as system graph. The proposed optimization technique is tested on synthetic benchmark and on real QBox software to study its efficiency on large number of computing cores. The positive results of optimization are achieved and the summary is presented in the paper. Speedup of 17–20 % is obtained on scalable benchmarks using 300–4800 parallel processes.

Keywords: task mapping, cluster, communication graph, MPI.

References

1. Kopysov S.P., Novikov A.K., Tonkov L.E., et al. Metody privyazki parallelnyx processov i potokov k mnogoyadernym uzlam vychislitelnyx sistem [Methods of Parallel Processes and Threads Binding to Multicore Cluster Nodes]. Vestnik Udmurtskogo universiteta. Matematika. Mehanika. Kompyuternye nauki [Bulletin of Udmurt University: Mathematics, Mechanics and Computing Science]. 2010. Issue 1. P. 123–132.
2. Kurnosov M.G. Naznachenie vetvej parallelnoj programmy na processornye yadra raspredelennoj vychislitelnoj sistemy [Mapping of parallel program branches to computing cores in distributed system]. Materialy Mezhd. nauchno-texnicheskoj konferencii "Mnogoprocessornye vychislitelnye i upravlyayushhie sistemy" (pos. Divnomorskoe, Gelendzhik, 2007) [Multiprocessor computing and control systems, International Conference Proceedings (Divnomorskoe, Gelendzhik, Russia, 2007)]. P. 227–231.
3. Karlsson C., Davies T., Chen Z. Optimizing Process-to-Core Mappings for Application Level Multi-dimensional MPI Communications. Cluster Computing (CLUSTER), 2012 IEEE International Conf. Proceedings (Beijing, China, September, 24–28, 2012). P. 486–494. DOI: 10.1109/cluster.2012.47

4. Zhang J., Zhai J., Chen W., et al. Process Mapping for MPI Collective Communications. Lecture Notes in Computer Science. 2009. Vol. 5704. P. 81–92. DOI: 10.1007/978-3-642-03869-3_11
5. Chen H., Chen W., Huang J., et al. MPIPP: an Automatic Profile-Guided Parallel Process Placement Toolset for SMP Clusters and Multiclusters. ICS'06 Proceedings of the 20th annual international conference on Supercomputing (Queensland, Australia, June, 28 – July, 01, 2006). P. 353–360. DOI: 10.1145/1183401.1183451
6. Intel® MPI Library Reference Manual. URL: http://software.intel.com/sites/products/documentation/hpc/ics/impi/41/lin/Reference_Manual/index.htm (accessed: 20.12.2014).
7. Larsson P. Shared Memory Communication vs. Infiniband. URL: http://www.nsc.liu.se/~pla/blog/2013/09/12/smp-vs-infiniband (accessed: 20.12.2014).
8. Gygi F., Yates R.K., Lorenz J., et al. Large-Scale First-Principles Molecular Dynamics Simulations on the BlueGene/L Platform using the Qbox Code. Proceedings of the ACM/IEEE SC 2005 Conference (Seattle, WA, USA, November, 12–18, 2005). 24 p. DOI: 10.1109/sc.2005.40
9. Tornado SUSU Supercomputer. URL: http://supercomputer.susu.ac.ru/en/computers/tornado/ (accessed: 20.12.2014).

Received March 19, 2015.

УДК 004.43 DOI: 10.14529/cmse150202

СРЕДСТВА ПРОГРАММИРОВАНИЯ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ НА ОСНОВЕ ПЛИС VIRTEX-7 С ИСПОЛЬЗОВАНИЕМ СОФТ-АРХИТЕКТУР[1]

И.И. Левин, А.И. Дордопуло, В.Б. Коваленко, В.А. Гудков, А.А. Гуленок

В статье рассматриваются существующие средства проектирования цифровых устройств в программируемых логических интегральных схемах (ПЛИС), языки программирования реконфигурируемых вычислительных систем и возможность их использования при программировании многокристальных реконфигурируемых вычислительных систем. Также рассмотрены разработанные в НИИ МВС ЮФУ язык программирования высокого уровня COLAMO и комплекс средств разработки многокристальных решений на реконфигурируемых вычислительных системах. Особое внимание уделено новому подходу в программировании, заключающемуся в разработке и использовании настраиваемых проблемно-ориентированных софт-архитектур, которые позволяют сократить количество трансляций конфигурационных файлов ПЛИС при отладке параллельных программ на реконфигурируемых вычислительных системах. Проблемно-ориентированные софт-архитектуры дают возможность без перезагрузки файлов конфигурации ПЛИС вычислительного поля с помощью программной настройки изменять коммутацию между устройствами и создавать необходимые вычислительные структуры для решения прикладных задач пользователя, что существенно сокращает время отладки параллельных прикладных программ.

Ключевые слова: реконфигурируемые вычислительные системы, параллельное программирование, софт-архитектура, программирование РВС, проблемно-ориентированные софт-архитектуры.

Введение

За последние годы значительно вырос интерес к программируемым логическим интегральным схемам. Различные отечественные и зарубежные компании создают как отдельные ускорители с одним-двумя кристаллами программируемых логических интегральных схем (ПЛИС), так и целые вычислительные комплексы. Реконфигурируемые вычислительные системы (РВС) на базе ПЛИС показали свою эффективность при решении вычислительно трудоемких задач из различных областей науки и техники. Такие фирмы как Nallatech [1] и Pico Computing [2] выпускают ряд ускорителей и несущих плат с небольшим числом (до четырех) кристаллов ПЛИС, которые используются в создании серверов и гибридных кластерных систем фирмами HP и IBM. Компании Convey [3] и Maxeler Technologies [4] создают гибридные суперкомпьютеры на основе собственных гетерогенных кластерных узлов, каждый из которых может содержать от 1 до 4 кристаллов ПЛИС и несколько универсальных процессоров. Похожее решение используется и компанией SRC [5], которая выпускает узлы, названные MAP processor, для стойки (MAPstation) форм-фактором 1U, 2U и 4U. MAPstation 1U содержит один MAP processor. MAPstation 2U содержит до трех MAP processor. MAPstation 4U может содержать до 10 различных модулей — MAP

[1] Статья рекомендована для публикации Программным комитетом конференции «Научный сервис в сети Интернет — 2014»

processor, модуль с универсальным микропроцессором или модуль памяти. При этом наиболее рациональным способом построения реконфигурируемой вычислительной системы является использование в качестве основного вычислительного элемента множества кристаллов ПЛИС, объединенных в единое вычислительное поле высокоскоростными каналами передачи данных, что позволяет получить мощный вычислительный ресурс, достаточный для эффективного решения задачи.

Статья организована следующим образом. В первом разделе рассмотрены существующие в настоящее время системы проектирования, позволяющие создавать структурные решения прикладных задач на ПЛИС, отмечен их общий основной недостаток — невозможность автоматизированного создания схемотехнических решений для РВС, содержащих множество кристаллов ПЛИС, соединенных между собой пространственной коммутационной системой.

Решение этой проблемы возможно при использовании языка программирования высокого уровня с неявным описанием параллелизма COLAMO, краткое описание основных принципов которого представлено во втором разделе.

Третий раздел содержит описание концепции программирования многокристальных РВС с помощью софт-архитектур, которая позволяет без перезагрузки файлов конфигурации ПЛИС РВС с помощью программной настройки изменять коммутацию между устройствами и создавать необходимые вычислительные структуры для решения прикладных задач пользователя, что позволяет существенно сократить время отладки прикладной программы.

В заключении представлены практические результаты, полученные при решении задачи обработки спекл-изображений по методу Лабейри с помощью предложенной концепции софт-архитектур.

1. Существующие средства программирования РВС

Наиболее популярными как в качестве отдельного средства разработки, так и в составе комплексов являются программы-синтезаторы, предоставляемые фирмами-изготовителями ПЛИС: ISE и Vivado фирмы Xilinx [6], Quartus II фирмы Altera [7] и Actel Libero IDE фирмы Actel [8]. Данные средства, помимо непосредственно среды проектирования цифровых устройств, включают в себя ряд вспомогательных утилит: анализаторы временных характеристик, редакторы размещения, модули программирования ПЛИС, системы моделирования цифровых устройств и др. Благодаря широкому инструментарию эти системы проектирования обеспечивают полный цикл разработки цифровых устройств в пределах одного кристалла ПЛИС: создание исходного описания проекта, синтез, моделирование, размещение, трассировку, конфигурирование кристалла.

В связи с постоянным ростом емкости кристаллов ПЛИС проектирование решений прикладных задач в кристаллах ПЛИС с помощью языков описания аппаратуры (VHDL, AHDL, Verilog и др.), а также разработка цифровых устройств в графических редакторах становится все более трудоемкой. Поэтому в настоящее время ведущие производители ПЛИС и реконфигурируемых вычислителей ориентируются на языки высокого уровня. Так, в новой среде разработки Vivado фирмы Xilinx добавлен новый инструмент проектирования Vivado HLS [9], основанный на языке высокого уровня, а ком-

пания Altera для своих ПЛИС предлагает инструментарий разработки Altera SDK [10] для нового стандарта параллельного программирования гетерогенных систем OpenCL. В данных решениях используются трансляторы C-подобных языков, генерирующие из программы на аналоге языка программирования высокого уровня C код на языках описания аппаратуры на уровне регистровых передач (RTL-уровень, трансляторы C-to-RTL).

Несмотря на схожесть синтаксисов данных C-подобных языков с самим языком C, подобный подход вовсе не означает, что исходный код на C, написанный под персональный компьютер или кластерную вычислительную систему, будет понят трансляторами C-to-RTL. Выбор языка C в качестве основы обусловлен широкой распространенностью данного языка, что существенно упрощает освоение новых инструментариев разработки решений прикладных задач для ПЛИС.

Также при использовании трансляторов C-to-RTL весь код программы либо явно указанные процедуры транслируются в RTL-описания отдельных кристаллов ПЛИС. В подобных системах разработки отсутствует инструментарий, обеспечивающий автоматические разбиения параллельной программы на множество связанных кристаллов ПЛИС.

В Vivado HLS проект разрабатывается в рамках одного кристалла ПЛИС, и если программисту недостаточно аппаратного ресурса кристалла, то он вынужден самостоятельно распределять вычисления между несколькими проектами для каждой ПЛИС и синхронизировать управляющие и информационные потоки между ними.

Стандарт OpenCL используется компанией Nallatech (разработчиком реконфигурируемых вычислителей) и подразумевает использование нескольких ПЛИС в одном проекте. Программирование решений в кристаллах ПЛИС в данном случае осуществляется с помощью вызова функций из библиотек инструментария разработки Altera SDK, и в каждом задействованном кристалле ПЛИС выполняются вычисления, описанные отдельным участком кода. Таким образом, программа, написанная в стандарте OpenCL, представляет собой основной код, написанный под традиционные процессоры, и отдельные участки кода, написанные под ПЛИС, задействованные как сопроцессоры. В данном случае задача синхронизации данных также возлагается на самого программиста.

Еще одним известным средством программирования ПЛИС является комплекс, разработанный компанией Mitrionics, состоящий из виртуального процессора Mitrion Virtual Processor (MVP), программирование которого осуществляется на языке высокого уровня Mitrion-C, и библиотеки функций для построения хост-программ MithalAPI, входящих в пакет разработки Mitrion SDK [6].

Mitrion-C является специализированным языком высокого уровня семейства C, но работает с потоком данных, а не с потоком команд, реализуя параллелизм на уровне инструкций и циклов. Параллелизм на уровне инструкций является неявным, поскольку анализ зависимостей данных происходит автоматически на этапе компиляции программы. Порядок выполнения инструкций в языке не определен и зависит от степени готовности данных для каждой конструкции. Таким образом, параллелизм на этом уровне обеспечивается независимостью данных.

Основным средством распараллеливания на языке Mitrion-C является распараллеливание на уровне циклов. Параллелизм на уровне циклов является явным и определяется ключевым словом «Foreach», что дает возможность программисту самому определять, какие циклы он хочет распараллеливать.

Наличие неявного параллелизма на уровне инструкций требуют от пользователя особого внимания при разработке параллельной программы, а явное указание распараллеливаемых циклов может приводить к значительным трудностям при модификации параллельной программы или ее переносимости на другие архитектуры ПЛИС.

В целом язык Mitrion-C обеспечивает пользователя достаточно простыми средствами распараллеливания, но предъявляет серьезные требования к разработке параллельного алгоритма и упрощает процесс разработки параллельных программ незначительно.

Разрабатываемая на языке Mitrion-C программа должна быть полностью реализована на одном виртуальном процессоре MVP, что не позволяет программировать многокристальные РВС, а следовательно, существенно снижает эффективность использования программного комплекса компании Mitrionics. Для программирования многокристальных РВС со связями между ПЛИС программисту необходимо самому реализовать интерфейс (протокол) обмена данными между ПЛИС и решить вопросы, связанные с синхронизацией потоков данных. В этом случае программа для РВС вырождается в программу для кластера (множество процессоров MVP), реализованного на ПЛИС, что существенно снижает эффективность реализации задач на многокристальных РВС.

2. Язык COLAMO и комплекс средств программирования многокристальных РВС

Альтернативный подход к программированию РВС предлагается в Научно-исследовательском институте многопроцессорных вычислительных систем Южного федерального университета (НИИ МВС ЮФУ), занимающемся разработкой многокристальных реконфигурируемых вычислительных систем различной архитектуры и конфигурации уже более 15 лет.

Опыт решения различных классов задач в НИИ МВС ЮФУ показал, что для эффективного решения современных трудоемких задач программисту необходимы средства программирования, обеспечивающие следующие основные возможности:

- программирование на языке высокого уровня;
- поддержка многокристального программирования;
- обеспечение высокой частоты работы ПЛИС;
- высокая плотность заполнения кристалла ПЛИС;
- поддержка конвейерной и макроконвейерной организации вычислений.

В НИИ МВС ЮФУ разработан и широко используется программный комплекс, включающий в себя следующие компоненты:

- *транслятор языка программирования COLAMO*, осуществляющий трансляцию исходного кода на COLAMO в информационный граф параллельной прикладной программы;
- *синтезатор масштабируемых схемотехнических решений* на уровне логических ячеек ПЛИС Fire!Constructor, осуществляющий отображение полученного от транслятора языка программирования COLAMO информационного графа на архитектуру РВС, размещение отображенного решения по кристаллам ПЛИС и автоматическую синхронизацию фрагментов информационного графа в разных кристаллах ПЛИС;
- *библиотеку IP-ядер*, соответствующих операторам языка COLAMO (функционально-законченных структурно-реализованных аппаратных устройств) для раз-

личных предметных областей, и интерфейсов для согласования скорости обработки информации и связи в единую вычислительную структуру;

– *средства отладки* программ, *средства доступа и мониторинга* состояния РВС.

Язык высокого уровня COLAMO предназначен для описания реализации параллельного алгоритма и создания на основе принципов структурно-процедурной организации вычислений [11] в архитектуре РВС специализированной вычислительной структуры, которая предполагает последовательную смену структурно (аппаратно) реализованных фрагментов информационного графа задачи, каждый из которых является вычислительным конвейером потока операндов. Таким образом, приложение (прикладная задача) для РВС состоит из структурной составляющей, представленной в виде аппаратно реализованных фрагментов вычислений, и процедурной составляющей, представляющей собой единую для всех структурных фрагментов управляющую программу последовательной смены вычислительных структур и организации потоков данных. Для представления такой организации вычислений в языке используется понятие «кадр». Кадром является программно-неделимый компонент, представляющий собой совокупность операторов, которые реализуются в виде арифметико-логических команд и команд чтения/записи, выполняемых на различных функциональных устройствах, соединенных между собой в соответствии с информационной структурой алгоритма.

В языке COLAMO отсутствуют явные формы описания параллелизма, а распараллеливание достигается с помощью объявления типов доступа к переменным и индексации элементов массивов, что характерно для языков потока данных. Для обращения к данным используется два основных метода доступа: параллельный доступ (задаваемый типом Vector) и последовательный доступ (задаваемый типом Stream). Степень параллелизма определяется по минимальному значению параметра распараллеливания. Для типа доступа Stream степень параллелизма равна 1, а для типа доступа Vector определяется наименьшим значением векторной составляющей каждого массива, участвующего в вычислениях. Для параллельного типа доступа возможна одновременная обработка всех размерностей массивов, заданных типом Vector, при этом повышается аппаратный ресурс на обработку, но снижается время обработки.

Многомерные массивы данных могут иметь множество измерений, каждое из которых может иметь последовательный или параллельный тип доступа, задаваемый ключевым словом Stream или Vector соответственно. Изменение типа доступа позволяет достаточно просто управлять как степенью распараллеливания вычислений на уровне описания структур данных, так и скоростью обработки и занимаемым ресурсом, что позволяет программисту описывать различные виды параллелизма в достаточно сжатом виде.

Помимо типа доступа, для переменной в языке COLAMO определены также и типы ее хранения: мемориальный (Mem), регистровый (Reg) и коммутационный (Com).

Мемориальной переменной называется величина, хранящаяся в ячейке распределенной памяти и, следовательно, сохраняющая свое значение до очередного переприсваивания. Для мемориальной переменной возможно одновременное выполнение только одного процесса. Поэтому в семантике языка COLAMO для мемориальной переменной в теле кадра действуют правило однократного присваивания и правило единственной подстановки. Правило однократного присваивания указывает на то, что мемориальная переменная в кадре изменяет свое значение только один раз. Правило единственной подста-

новки определяет, что переменная в кадре может использоваться только для одного процесса чтения или записи.

Для описания связей между элементами информационного графа задачи в языке COLAMO предназначена коммутационная переменная. Поскольку коммутационная переменная описывает информационные связи, то она не требует никакого вычислительного аппаратного ресурса для своей реализации. Доступ к значению коммутационной переменной после выполнения кадра невозможен. Коммутационная переменная необходима транслятору для указания информационных зависимостей при построении вычислительной структуры задачи. На коммутационную переменную, как и на мемориальную переменную, действует правило однократного присваивания, но для нее не выполняется правило единственной подстановки. Использование коммутационных переменных позволяет легко разветвлять и дублировать потоки данных, но не позволяет реализовать рекурсию.

Для организации рекурсии в языке COLAMO используется регистровая переменная, которая представляет собой регистр на аппаратном уровне и используется для хранения промежуточных данных, полученных в процессе вычислений. Единственным ограничением для регистровой переменной в теле кадра является правило однократного присваивания.

Трансляция программы на языке высокого уровня COLAMO состоит в создании схемотехнической конфигурации вычислительной системы (структурной составляющей) и параллельной программы, управляющей потоками данных (потоковой и процедурной составляющих) [12]. Создание структурной составляющей заключается в построении вычислительного графа, соответствующего описанным на COLAMO информационным зависимостям между результатами вычислений. При этом для каждой используемой в тексте программы операции подставляется специализированный вычислительный блок в зависимости от типа доступа к переменным, типа данных, их разрядности и т.д. Синтезированный информационный граф задачи передается в синтезатор Fire!Constructor для отображения на аппаратный ресурс многокристальной РВС [13].

Задача автоматического отображения параллельной программы на аппаратный ресурс многокристальной РВС состоит из трех подзадач: разбиения информационного графа на непересекающиеся подграфы, размещения сформированных подграфов в ПЛИС РВС и трассировки внешних связей размещенных подграфов в системе коммутаций РВС.

Результатом работы синтезатора Fire!Constructor являются файлы VHDL-описаний и файлы временных и топографических ограничений (User Constraints Files). Файлы VHDL описывают структурные реализации фрагментов параллельной программы. На основании этих файлов и библиотеки схемотехнических элементов формируются проекты для синтезатора ISE под каждую отдельную ПЛИС. Далее с помощью синтезатора ISE формируются конфигурационные файлы ПЛИС. Полученные конфигурационные файлы загружаются в РВС.

Программа на языке COLAMO разрабатывается в едином проекте и может быть транслирована на любую РВС, описание которой и соответствующие библиотеки присутствуют в комплексе программирования РВС. В отличие от других существующих систем разработки решений прикладных задач на ПЛИС пользователю не требуется в тексте программы указывать, какие фрагменты программы в каких ПЛИС будут вы-

полняться. Синтезатор Fire!Constructor обеспечивает автоматическое разбиение вычислительной структуры программы на языке COLAMO на несколько проектов в синтезаторе Xilinx ISE, при этом обеспечивается синхронизация информационных потоков как внутри кристалла ПЛИС, так и между ними.

3. Использование софт-архитектур для программирования РВС на базе ПЛИС VIRTEX-7

С ростом емкости кристаллов ПЛИС растет и время синтеза конфигурационных файлов для загрузки в ПЛИС. Современные ПЛИС фирмы Xilinx семейства Virtex-7 имеют емкость от 33 до 200 млн. эквивалентных вентилей. Время синтеза конфигурационного файла с большим заполнением (от 80%) для одного такого кристалла может составлять от нескольких часов до нескольких суток, что существенно увеличивает время отладки параллельной программы для многокристальной РВС. Рассмотренные ранее средства программирования РВС лишь частично сокращают процесс отладки, позволяя моделировать формируемые виртуальные вычислительные структуры для отдельных кристаллов ПЛИС, и не имеют средств для отладки для многокристальных решений.

Для сокращения времени отладки многокристальных решений задач предметной области разработана [14] и реализована [15] концепция софт-архитектур, которая позволяет без перезагрузки файлов конфигурации ПЛИС вычислительного поля с помощью программной настройки изменять коммутацию между устройствами и создавать необходимые вычислительные структуры для решения прикладных задач пользователя.

Софт-архитектура — это созданная вычислительная структура, содержащая макрообъекты. Макрообъект — это совокупность вычислительных устройств, выполняющих определенную группу команд и соединенных между собой коммутационной системой. Для каждого класса задач, решаемых на РВС, можно подобрать определенный набор оптимальных вычислительных структур (макрообъектов), наиболее эффективно решающие задачи данного класса. Для макрообъекта допустимо изменение количества функциональных вычислительных устройств и их параметров (разрядности операндов, числа информационных каналов, системы команд и др.), но недопустимо изменение их назначения. Таким образом, макрообъект с точки зрения прикладного программиста является «заготовкой», которая может доопределяться им при создании конкретного технического решения, а затем тиражироваться в нужном количестве в ПЛИС вычислительных модулей и соединяться с подобными или другими макрообъектами в вычислительные структуры, которые оптимально соответствуют структуре решаемой задачи. На основе макрообъектов возможно создание «софт-архитектуры РВС», под которой понимается созданная схемотехником вычислительная структура, содержащая макрообъекты, в которой можно без перезагрузки файлов конфигурации ПЛИС вычислительного поля с помощью программной настройки изменять коммутацию между устройствами и создавать необходимые вычислительные структуры для решения прикладных задач пользователя.

Использование софт-архитектур позволяет сократить время разработки прикладной программы РВС за счет уменьшения времени трансляции. При этом производительность вычислительной системы остается на уровне, сопоставимом с производительностью специализированных вычислительных структур. Это достигается путем создания и предварительной загрузки в РВС настраиваемых проблемно-ориентированных софт-

архитектур, способных решать задачи определенного класса. Опыт создания вычислительных структур на РВС показывает, что для каждого класса задач можно выделить конечный набор вычислительных узлов, при помощи которых может быть построена вычислительная архитектура, эффективно решающая задачи данного класса. Вычислительные узлы, управляемые посредством системы команд и связанные в единую вычислительную структуру, составляют софт-архитектуру.

Это обеспечивает при сохранении преемственности принципов программирования и использования языка высокого уровня для программирования РВС возможность простой адаптации программных компонентов средств разработки для РВС при переходе на новые топологии ПВМ без внесения существенных изменений в код программных компонентов комплекса, а также позволяет сократить время решения прикладных задач.

Для создания софт-архитектур разработан язык SADL (Soft-Architecture Development Language) [14], который транслируется в виртуальную архитектуру вычислительной системы для синтезатора Fire!Constructor, на которую отображается информационный граф прикладной задачи. Для создания софт-архитектуры выполняются:

- разработка описания софт-архитектуры на языке SADL;
- трансляция описания софт-архитектуры в промежуточное представление при помощи синтезатора конфигураций параллельно-конвейерных вычислительных структур Fire!Constructor;
- размещение элементов софт-архитектуры на аппаратной платформе при помощи синтезатора масштабируемых параллельно-конвейерных процедур Steam!Constructor.

Транслятор языка SADL преобразует текст программы в промежуточное представление, используемое синтезатором FireConstructor для размещения на аппаратной платформе РВС. Результатом размещения софт-архитектуры на аппаратную платформу являются модифицированный файл промежуточного представления и конфигурационные файлы для ПЛИС, участвующих в размещении софт-архитектуры на аппаратной платформе РВС. После того как софт-архитектура была размещена на аппаратной платформе РВС, она может быть использована для решения различных прикладных задач заданной проблемной области.

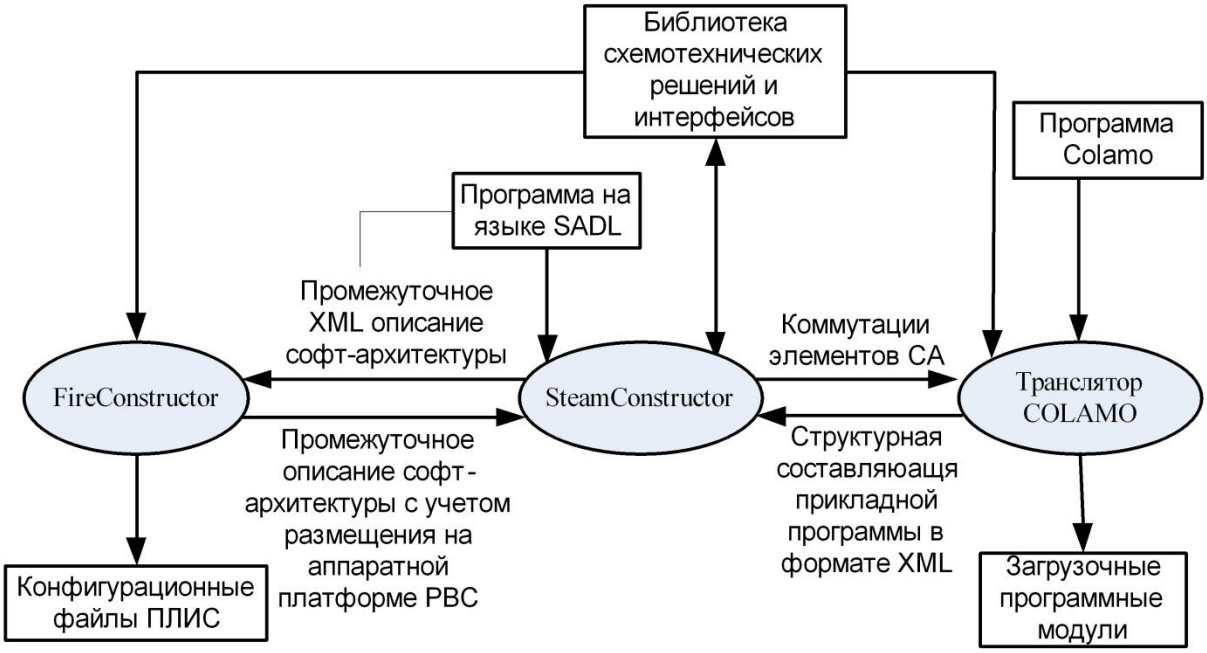
Разработка прикладной программы для софт-архитектуры РВС состоит из следующих основных этапов:

1) разработка параллельной программы на языке высокого уровня COLAMO;
2) трансляция параллельной программы, преобразование информационного графа в структурный и процедурный компоненты;
3) отображение структурного компонента параллельной программы на софт-архитектуру при помощи синтезатора конфигураций параллельно-конвейерных вычислительных структур SteamConstructor;
4) трансляция процедурного компонента параллельной программы на уровень команд устройств софт-архитектуры;
5) формирование загрузочного out-файла, содержащего команды элементов софт-архитектуры;

6) загрузка конфигурационных файлов ПЛИС, полученных в результате размещения элементов софт-архитектуры на аппаратной платформе реконфигурируемой системы;
7) загрузка out-файла;
8) загрузка в софт-архитектуру исходных данных решаемой задачи;
9) запуск программы на исполнение и выгрузка результатов.

Взаимодействие средств разработки прикладных программ при разработке софт-архитектуры показано на рисунке.

Благодаря разработанным программным средствам разработка и модификация софт-архитектур не требуют привлечения высококвалифицированного специалиста-схемотехника для изменения вычислительной структуры. При этом время, затрачиваемое на создание или модификацию софт-архитектуры, значительно сокращается, а получаемые многокристальные архитектурные решения сравнимы по эффективности с решениями, выполненными специалистами-схемотехниками вручную. В соответствии с основными принципами языка COLAMO параллельные прикладные программы могут быть легко модифицированы для адаптации к доступному вычислительному ресурсу.



Структура взаимодействия средств разработки

Автоматизация отображения графов на ресурс РВС позволяет разработчикам прикладных задач мыслить не несколькими ПЛИС, а одной виртуальной ПЛИС с большим логическим объемом.

Комплекс разработанных программных средств позволяет программисту РВС разрабатывать и выполнять отладку прикладных параллельных программ для РВС, не вникая в особенности архитектуры РВС, а также самостоятельно создавать и модифицировать различные софт-архитектуры, ориентируясь на предметную область, к которой принадлежит решаемая задача.

4. Заключение

Благодаря разработанной концепции программирования РВС с использованием софт-архитектур программисту не требуется при каждом изменении программы синтезировать новые конфигурационные файлы ПЛИС. Текст программы транслируется в команды настройки, заранее загруженной в РВС софт-архитектуры, на выполнение тех или иных вычислений. Если разработанная софт-архитектура не имеет достаточного количества вычислительных блоков определенного типа, или блоки данного типа совсем отсутствуют в ней, то программист с помощью языка SADL может модернизировать софт-архитектуру и лишь в данном случае потребуется синтезировать новые конфигурационные файлы ПЛИС.

Современные ПЛИС, такие как кристаллы семейства Virtex-7 фирмы Xilinx, имеют большую емкость и время синтеза конфигурационных файлов занимает существенную часть на этапе отладки параллельной программы. Таким образом, снижение количества выполнения этапа синтеза конфигурационных файлов ПЛИС приводит к снижению времени отладки параллельной программы и времени разработки решения прикладной задачи в целом. При этом производительность вычислительной системы при использовании софт-архитектур остается на уровне, сопоставимом с производительностью специализированных вычислительных структур.

Предложенная концепция программирования РВС была апробирована на РВС на основе ПЛИС Xilinx Virtex-7, при решении задач из области цифровой обработки сигнала. В частности были решены задача многоканальной цифровой фильтрации для мониторинга систем цифровой связи третьего и четвертого поколений (КИХ-фильтр высокого порядка), задача нахождения спектра комплексного сигнала с использованием алгоритма быстрого преобразования Фурье (БПФ) и задача обработки спекл-изображений по методу Лабейри. Использование софт-архитектур позволило сократить время отладки в 2–3 раза, при этом снижение производительности по сравнению с производительностью специализированных вычислительных структур решающих данные задачи, не превышает 15–20%.

Работа выполнена при финансовой поддержке Министерства образования и науки РФ по Соглашению о предоставлении субсидии №14.578.21.0006 от 05.06.2014, уникальный идентификатор RFMEFI57814X0006, НИР регистрационный №114061040060 и НИР регистрационный №01201460286.

Литература

1. http://www.nallatech.com/(дата обращения 25.12.2014)
2. http://picocomputing.com/(дата обращения 25.12.2014)
3. http://www.conveycomputer.com/(дата обращения 25.12.2014)
4. http://www.maxeler.com/(дата обращения 25.12.2014)
5. http://www.srccomp.com/(дата обращения 25.12.2014)
6. Зотов, В.Ю. Проектирование цифровых устройств на основе ПЛИС фирмы XILINX в САПР WebPACK ISE / В.Ю. Зотов. — М.: Горячая линия-Телеком. 2003. — 624 с.
7. Quartus II Handbook Version 10.1 Volume 1: Design and Synthesis. Altera Corporation 2010 — 130 p.

8. Libero IDE v9.1 User's Guide. Actel Corporation 2010. — 633 p.
9. Проектирование для ПЛИС Xilinx с применением языков высокого уровня в среде Vivado HLS / Компоненты и технологии, 2013. — №12. — С. 10—17.
10. http://www.altera.com/literature/lit-opencl-sdk.jsp/(дата обращения 25.12.2014)
11. Каляев, И.А. Реконфигурируемые мультиконвейерные вычислительные структуры / И.А. Каляев, И.И. Левин, Е.А. Семерников, В.И. Шмойлов / Изд. 2-е, перераб. и доп. / Под общ. ред. И.А. Каляева. — Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. — 344 с.
12. Гудков, В.А. Расширение языка высокого уровня COLAMO для программирования реконфигурируемых вычислительных систем на уровне логических ячеек ПЛИС / В.А. Гудков, И.И. Левин // Вестник компьютерных и информационных технологий. — М.: Машиностроение, 2010. — № 12. — С. 10—17.
13. Гудков, В.А. Средства программирования реконфигурируемых многопроцессорных вычислительных систем / В.А. Гудков, А.А Гуленок, А.И. Дордопуло, Л.М. Сластен // Известия ТРТУ. Тематический выпуск «Интеллектуальные и многопроцессорные системы». — Таганрог: Изд-во ТРТУ, 2006. — № 16 (71). Специальный выпуск. — С. 16—20.
14. Семерников, Е.А. Организация многоуровневого программирования реконфигурируемых вычислительных систем / Е.А. Семерников, В.Б. Коваленко // Вестник компьютерных и информационных технологий. — М.: Машиностроение, 2011. — № 9. — С. 3—10.
15. Gudkov, V.A. / V.A. Gudkov, A.A., Gulenok, V.B. Kovalenko, L.M. Slasten Multi-level Programming of FPGA-based Computer Systems with Reconfigurable Macroobject Architecture / Preprints of the 12th IFAC Conference on Programmable Devices and Embedded Systems PDES, 2013.— Technical University of Ostrava, Czech Republic. P. 65—70.

Левин Илья Израилевич, доктор технических наук, профессор, зам. директора по науке, НИИ многопроцессорных вычислительных систем имени академика А.В. Каляева, Южный федеральный университет (Таганрог, Российская Федерация), levin@mvs.tsure.ru.

Дордопуло Алексей Игоревич, кандидат технических наук, старший научный сотрудник, Южный научный центр РАН (Ростов-на-Дону, Российская Федерация), scorpio@mvs.tsure.ru.

Коваленко Василий Борисович, кандидат технических наук, младший научный сотрудник, Южный научный центр РАН, (Ростов-на-Дону, Российская Федерация), vereten@hotbox.ru.

Гудков Вячеслав Александрович, кандидат технических наук, старший научный сотрудник, НИИ многопроцессорных вычислительных систем имени академика А.В. Каляева, Южный федеральный университет (Таганрог, Российская Федерация), Slava_Gudkov@mail.ru.

Гуленок Андрей Александрович, кандидат технических наук, старший научный сотрудник, НИИ многопроцессорных вычислительных систем имени академика А.В. Каляева, Южный федеральный университет (Таганрог, Российская Федерация), andrei_gulenok@mail.ru.

Поступила в редакцию 29 декабря 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 2, pp. 20–32

DOI: 10.14529/cmse150202

PROGRAMMING TOOLS FOR RECONFIGURABLE COMPUTER SYSTEMS BASED ON VIRTEX-7 FPGAS WITH USING SOFT-ARCHITECTURES

I.I. Levin, Academician A.V. Kalyaev SRI multiprocessor computer system at Southern Federal University (Taganrog, Russian Federation) levin@mvs.tsure.ru,

A.I. Dordopulo, Southern Scientific Centre of the RAS (Rostov-on-Don, Russian Federation) scorpio@mvs.tsure.ru,

V.B. Kovalenko, Southern Scientific Centre of the RAS (Rostov-on-Don, Russian Federation) vereten@hotbox.ru,

V.A. Gudkov, Academician A.V. Kalyaev SRI multiprocessor computer system at Southern Federal University (Taganrog, Russian Federation) Slava_Gudkov@mail.ru,

A.A. Gulenok, Academician A.V. Kalyaev SRI multiprocessor computer system at Southern Federal University (Taganrog, Russian Federation) andrei_gulenok@mail.ru

The paper covers the existing design tools of digital devices in FPGAs, programming languages of reconfigurable computer systems and ability of their use for programming of multichip reconfigurable computer systems. Besides it deals with the high-level programming language COLAMO and the multichip solution development suite for reconfigurable computer systems that are developed in SRI MCS SFU. A particular attention is paid to a new programming approach, which consists in development and use of adjustable special-purpose soft-architectures which help to reduce the number of translations of FPGA bitstream files during debugging of parallel programs on reconfigurable computer systems. Owing to special-purpose soft-architectures it is possible to change communication links between devices and create required computing structures for user applications only by means of program adjustment and without re-loading FPGA bitstream files of the computational field. It considerably reduces the debugging time of parallel applications.

Keywords: reconfigurable computer systems, parallel programming, soft-architecture, RCS programming, special-purpose soft-architectures.

References

1. http://www.nallatech.com/(accessed:25.12.2014)
2. http://picocomputing.com/(accessed: 25.12.2014)
3. http://www.conveycomputer.com/(accessed: 25.12.2014)
4. http://www.maxeler.com/(accessed: 25.12.2014)
5. http://www.srccomp.com/(accessed: 25.12.2014)
6. Zotov V.Y. Proijektirovanije tsifrovykh ustroistv na osnove PLIS firmy XILINX v SAPR WebPACK ISE [The Design (большая буква) of Digital Devices Based on XILINX FPGAs Using CAD WebPACK ISE]. M.: Goryachaya liniya-Telekom. [Moscow: Hot-Line-Telecom]. 2003. 624 P.

7. Quartus II Handbook Version 10.1 Volume 1: Design and Synthesis. Altera Corporation, 2010. 130 P.
8. Libero IDE v9.1 User's Guide. Actel Corporation. 2010. 633 P.
9. Proijektirovanije dlya PLIS Xilinx s primenenijem yazikov visokogo urovnya v srede Vivado HLS [Design for a Xilinx FPGAs Using High-level Languages in Vivado HLS IDE] / Komponenty i tekhnologii. [Components and technologies]. 2013. Vol. 12. P. 10—17.
10. http://www.altera.com/literature/lit-opencl-sdk.jsp/(accessed: 25.12.2014)
11. Kaliaev I.A., Levin I.I., Semernikov E.A., Shmoilov V.I. Rekonfiguriruijemyie multikonveyijernyie vichislitelniye struktury [Reconfigurable Multipipeline Computational Structures] / Izd. 2nd, pererabotannoye i dopolnennoye / Pod obshei redaktsiyei I.A. Kaliaeva. [Second Edition, Rev. and Suppl./ Editor I.A. Kaliaev]. Rostov-on-Don, Publishing of SSC RAS, 2009. 344 P.
12. Gudkov V.A.,(шр) Levin I.I. Rasshireniye yazika visokogo urovnya COLAMO dlya programmirovaniya rekonfiguriruijemykh vichislitelnykh sistem na urovnie logicheskikh yacheek PLIS [High-level Language COLAMO Extension for Reconfigurable Computing Systems Programming at the Level of the FPGA Logic Cells]. Vestnik komputernykh i informatsionnykh tekhnologii. M.: Mashinostroyeniye. [Herald of Computer and Information Technologies]. Moscow, Engineering , 2010. Vol. 12. P. 10—17.
13. Gudkov V.A., Gulenok A.A., Dordopulo A.I., Slasten L.M. Sredstva programmirovaniya rekonfiguriruijemykh mnogoprotsessornykh vichislitelnykh sistem [Software Tools for Reconfigurable Multiprocessor Computing Systems]. Izvestia TRTU. Tematicheskiy vipusk "Intellectualnie i mnogoprocessornie systemi". [Proceedings of TSURE. Subject issue "Intelligent and multiprocessor systems"]. Taganrog, TSURE Publishing, 2006. Vol. 16 (71). Special issue. P. 16—20.
14. Semernikov E.A., Kovalenko V.B. Organizatsiya mnogourovnevogo programmirovaniya rekonfiguriruijemykh vichislitelnykh sistem [Organization Multi-level Programming of Reconfigurable Computing Systems]. Vestnik komputernykh i informatsionnykh tekhnologii. M.: Mashinostroyeniye. [Herald of Computer and Information Technologies.] Moscow, Engineering, 2011. Vol. 9. P. 3—10.
15. Gudkov V.A., Gulenok A.A., Kovalenko V.B., Slasten L.M. Multi-level Programming of FPGA-based Computer Systems with Reconfigurable Macroobject Architecture. Preprints of the 12th IFAC Conference on Programmable Devices and Embedded Systems PDES-2013, Technical University of Ostrava, Czech Republic (24—29 September 2013). P. 65—70.

Received December 29, 2014.

УДК 004.457, 004.382.2 DOI: 10.14529/cmse150203

ОБЕСПЕЧЕНИЕ ОПЕРАТИВНОГО КОНТРОЛЯ И ЭФФЕКТИВНОЙ АВТОНОМНОЙ РАБОТЫ СУПЕРКОМПЬЮТЕРНОГО КОМПЛЕКСА МГУ[1]

А.С. Антонов, Вад.В. Воеводин, А.А. Даугель-Дауге, С.А. Жуматий, Д.А. Никитенко, С.И. Соболев, К.С. Стефанов, П.А. Швец

В НИВЦ МГУ разрабатывается система для обеспечения оперативного контроля и поддержки эффективного автономного функционирования суперкомпьютерных комплексов. Данная система внедряется в Суперкомпьютерном центре МГУ. В работе описывается опыт установки, настройки и эксплуатации системы для контроля работы суперкомпьютера «Чебышёв».

Ключевые слова: суперкомпьютер, граф, графовая модель, мониторинг, оперативный контроль, автономная работа, Octotron.

Введение

Система Octotron, разрабатываемая в НИВЦ МГУ имени М.В. Ломоносова, предназначена для обеспечения оперативного контроля функционирования суперкомпьютерных комплексов [1, 2]. Система отслеживает наступление нештатных ситуаций в работе всех компонентов комплекса. При возникновении таких ситуаций система выполняет определенный набор действий. Оперативное реагирование на сбои различного рода позволяет минимизировать их негативные последствия, тем самым обеспечивая эффективную автономную работу комплекса.

Кратко напомним основные понятия и идеи, заложенные в систему Octotron. Система работает на основе модели суперкомпьютерного комплекса, представленной в виде мультиграфа [3]. Вершины графа в модели соответствуют физическим или логическим компонентам суперкомпьютера, работу которых необходимо контролировать (вычислительные узлы, источники бесперебойного питания, очереди, лицензии на ПО и т.д.), а дуги — связям между компонентами («состоит из», «обеспечивает электропитанием», «соединены сетью Infiniband» и т.д.). С каждой из вершин связан набор атрибутов — характеристик состояния компонентов (температура процессора, объем памяти, число заданий в очереди и т.д.). Значения атрибутов поставляются штатными системами мониторинга суперкомпьютера либо получаются обращением к внешним интерфейсам компонентов. При изменении значений атрибутов срабатывают зависимые от них правила, определяющие факт наличия нештатной ситуации. В случае определения такой ситуации вызывается определенный набор действий, т.е. выполняется реакция.

Новизна подхода, реализованного в системе Octotron, заключается в использовании модели суперкомпьютерного комплекса в качестве входных данных контролирующей системы: фактически, суперкомпьютер начинает самостоятельно контролировать собственную работу. Среди функциональных аналогов можно назвать решения известных вендоров (HP OpenView [4], IBM xCAT [5]), китайскую систему Iaso [6], созданную спе-

[1] Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии – 2015».

циально для суперкомпьютера Tianhe-2, и российскую разработку «Система Автоматического Отключения Оборудования» (САОО) [7] компании Т-Платформы. Упомянутые разработки являются закрытыми и не являются универсальными: возможности первых двух систем во многом завязаны на аппаратуру собственного производства, для Iaso необходима модификация ядра ОС и установка дополнительных компонентов ПО, система САОО контролирует только инфраструктуру вычислительного комплекса.

В настоящее время система Octotron проходит апробацию в Суперкомпьютерном комплексе МГУ. Система уже находится в штатной эксплуатации на суперкомпьютере «Чебышёв». С начала 2015 года система запущена в опытную эксплуатацию на суперкомпьютере «Ломоносов», где она работает параллельно штатной системой автоматического отключения оборудования. Данная статья посвящена особенностям настройки и эксплуатации системы Octotron для обеспечения оперативного контроля работы суперкомпьютера «Чебышёв».

Статья организована следующим образом. Раздел 1 посвящен описанию модели суперкомпьютера «Чебышёв», построенной в системе Octotron. В разделе 2 приведен список источников данных о состоянии суперкомпьютера, используемых системой Octotron для поддержания автономной работы суперкомпьютера. В разделе 3 описываются правила системы Octotron для определения сбоев в работе суперкомпьютера, а в разделе 4 — методы реагирования системы на сбои. В заключении приводится краткое описание текущего состояния системы Octotron и планы ее развития.

1. Модель суперкомпьютера «Чебышёв»

Модель суперкомпьютера «Чебышёв» (625 вычислительных узлов, 5 000 процессорных ядер) содержит 10 228 вершин, 25 698 дуг и 205 044 атрибута. В модели отражены следующие компоненты суперкомпьютера [3]:

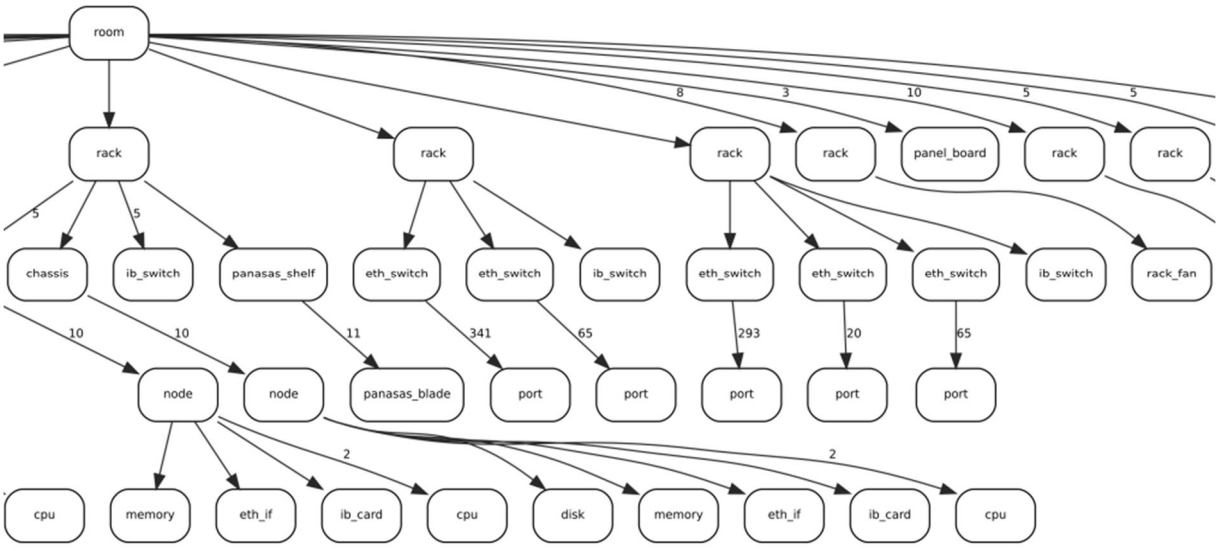


Рис. 1. Фрагмент модели суперкомпьютера «Чебышёв»: физические связи между компонентами

- система электропитания (источники бесперебойного питания, модули с батареями);
- система охлаждения (холодильные установки, воздушные кондиционеры, мониторинг среды);
- управляющая часть (узлы доступа, очереди задач);

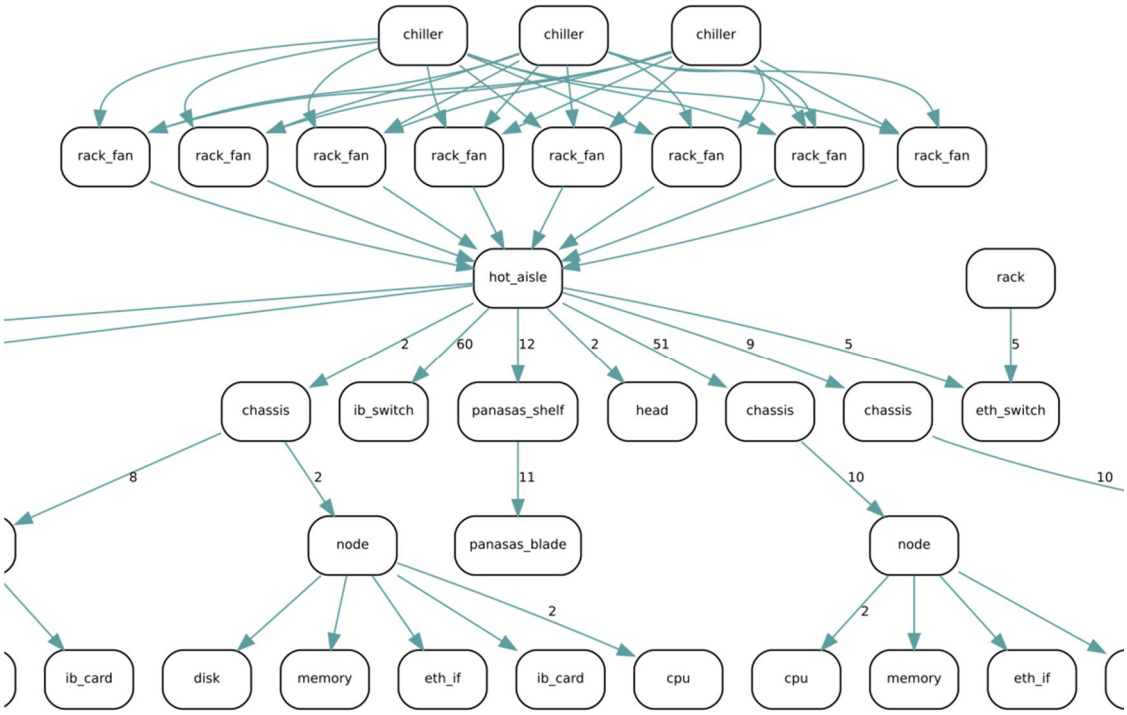


Рис. 2. Фрагмент модели суперкомпьютера «Чебышёв»: охлаждение

– вычислительная часть (шасси, узлы, диски, память);
– файловая система (разные компоненты для разных типов ФС);
– сеть Ethernet (коммутаторы, порты);
– сеть Infiniband (коммутаторы, менеджер сети).

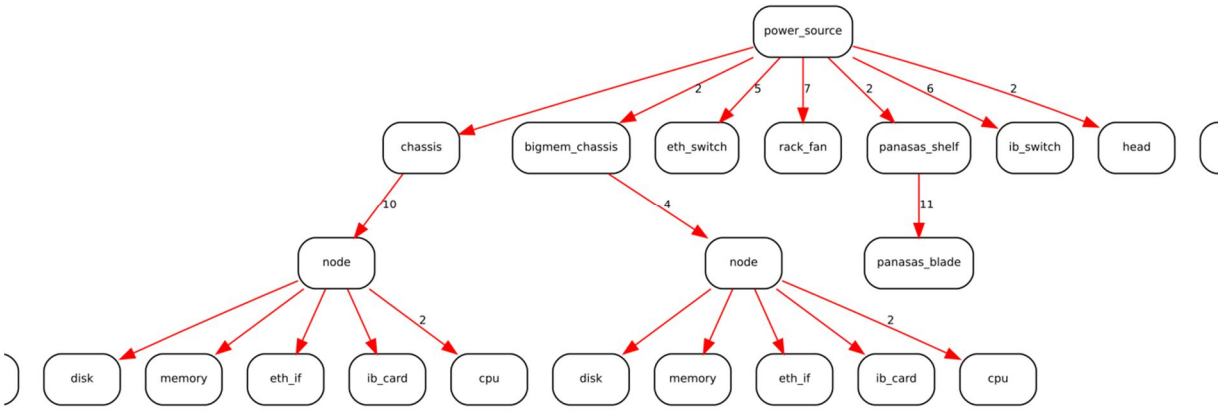


Рис. 3. Фрагмент модели суперкомпьютера «Чебышёв»: электропитание

Следующие типы связей реализованы в модели суперкомпьютера «Чебышёв»:

– содержит;
– охлаждает;
– соединены сетью Ethernet;
– соединены сервисной сетью;
– соединены сетью Infiniband;
– включает в себя;
– обеспечивает электропитанием.

На рис. 1 приведен фрагмент модели суперкомпьютера «Чебышёв», содержащий компоненты, связанные дугами типа «содержит». Количество вершин графа и дуг между ними велико, однако суперкомпьютер имеет в целом довольно регулярную структуру, поэтому модель содержит множество изоморфных подграфов. На рисунках подобные подграфы объединяются в один подграф, при этом дуге, ведущей к этому подграфу, приписывается число — количество объединенных подграфов.

Аналогичным образом отображаются фрагменты модели, описывающие охлаждение вычислительного комплекса (рис. 2) и электропитание его компонентов (рис. 3).

Разработка метода визуализации графа модели потребовала выполнения отдельного исследования. Подсистема визуализации графа на текущий момент позволяет получить компактные изображения, удобные для визуальной проверки корректности модели, для большинства типов связей, за исключением тех, которые образуют циклы в графе (как, например, связь «соединены сетью Infiniband»). Созданная подсистема стала существенным подспорьем при создании модели суперкомпьютера «Чебышёв».

В настоящий момент ведется активная разработка новой интерактивной подсистемы визуализации. Она позволяет развертывать и свертывать вершины графа, обеспечивая при необходимости доступ и просмотр списка атрибутов любой вершины, а также поддерживает различные варианты группировки вершин. Пример интерфейса новой подсистемы и отображения фрагмента модели приведен на рис. 4.

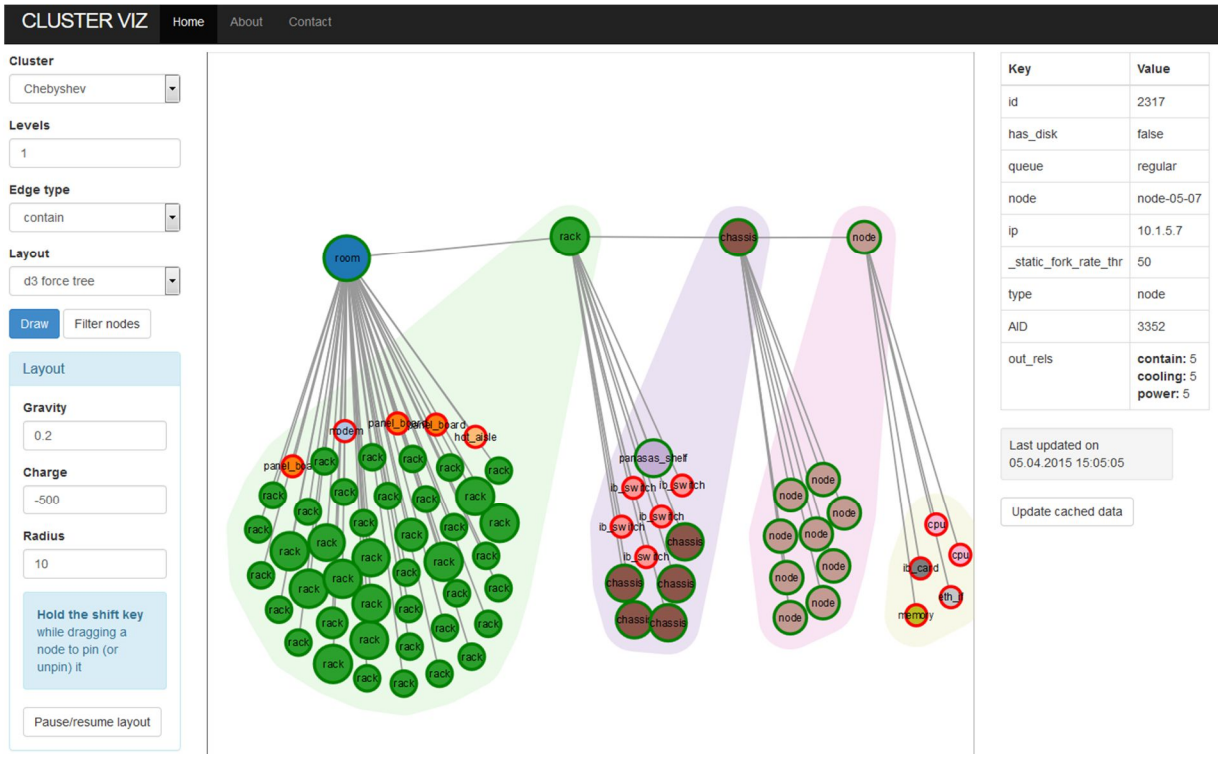


Рис. 4. Фрагмент модели суперкомпьютера «Чебышёв»: новые возможности отображения графа

2. Источники данных о состоянии суперкомпьютера

Основным поставщиком данных о работе компонентов суперкомпьютера «Чебышёв» является система мониторинга на основе collectd. С инфраструктурного оборудования данные поступают по протоколу SNMP. Все собираемые данные периодически импорти-

руются в Octotron. Заметим, что сама система Octotron жестко не привязана к какой-либо конкретной системе мониторинга. Использована может быть любая система мониторинга, а для ее совместной работы с системы Octotron потребуется разработка несложного модуля импорта данных. Подчеркнем также, что Octotron самостоятельно не опрашивает оборудование и не обращается к датчиками напрямую. В системе есть средства получения сигналов SNMP traps, отправляемые поддерживающим стандарт SNMP оборудованием в критических случаях. Для «Чебышёва» таким образом определяются критические сбои в системе электропитания.

С головных машин суперкомпьютера «Чебышёв» раз в 10 минут снимаются следующие данные:

- число активных ssh-сессий пользователей;
- число активных лицензий на ПО;
- количество задач в каждой из очередей: общее, ожидающих, готовящихся к запуску, выполняющихся, завершенных;
- число процессоров: общее, доступных для запуска задач, заблокированных;
- баланс счета GSM-модема, подключенного к одной из головных машин и использующегося для рассылки экстренных SMS-оповещений.

Данные, собираемые раз в 10 минут со всех вычислительных узлов:

- температура внутри узла;
- температура каждого процессора;
- идентификатор выполняющейся на узле задачи;
- состояние файловой системы;
- состояние памяти (общий объем, объем свободной/занятой памяти);
- состояние карты Infiniband (счетчики переданных/принятых пакетов, ошибки);
- состояние карты Ethernet (ошибки);
- другие системные данные: средняя загруженность узла, количество процессов-зомби и т.д.

Для узлов с жесткими дисками (около 100 шт.) дополнительно собирается информация SMART о состоянии HDD. Кроме того, один раз в час на каждом узле проверяется работа сервиса ssh, видимость в сети Infiniband и работоспособность MPI.

Для общей файловой системы на основе Panasas раз в 10 минут собираются общие данные (объемы свободного/занятого пространства, производительность), а также статус и загруженность каждого blade-модуля (всего их 132). С той же периодичностью собираются данные с коммутаторов Ethernet.

Информация о работе климатической системы вычислительного комплекса собирается чаще — 1 раз в минуту. Она включает в себя значения с нескольких датчиков температуры и влажности воздуха в помещении, а также состояние каждого из 8 кондиционеров (температура воздуха и охлаждающей жидкости на входе и на выходе, различные предупреждения). С той же периодичностью собираются данные с пяти источников бесперебойного питания – около 60 параметров с каждого: состояние внешнего питания, собственных режимов, аккумуляторных батарей и т.д.

3. Правила определения сбоев в работе суперкомпьютера

Правила в системе Octotron представляют собой функции, имеющие доступ к атрибутам вершин графа модели суперкомпьютера. Правила могут получать доступ к зна-

чениям атрибутов соседних вершин по заданному типу связи. Octotron позволяет формировать правила нескольких типов:

- сравнение значения датчика с константой. Пример: выход температуры компонента за пределы заданного порога;
- наличие аварийных значений одновременно у нескольких датчиков. Пример: сообщение о повышении температуры несколькими датчиками горячего коридора;
- значения датчиков у смежных (по графу модели) компонентов не соответствуют друг другу. Пример: порты на двух концах связи Ethernet находятся в разных режимах;
- сохранение определенного уровня значения датчика в течение заданного промежутка времени. Пример: уровень загруженности узла может ненадолго превысить штатные значения, но достаточно продолжительный высокий уровня загруженности свидетельствует о проблеме на узле;
- получение ошибочных значений датчика несколько раз подряд. Пример: узел более трех раз подряд не проходит проверку доступа по ssh.

Все сбои в работе суперкомпьютера, определяемые правилами, имеют свой уровень критичности. В настоящее время мы используем 4 уровня: Info (информация), Warning (предупреждение), Danger (опасность), Critical (авария). Сбои уровня Critical в основном связаны с повышением температуры воздуха в помещении, горячем коридоре и на компонентах. Подобные сбои могут нанести существенные повреждения оборудованию и помещениям вычислительного комплекса. К сбоям уровня Danger отнесены ситуации, существенно затрагивающие работу всех пользователей суперкомпьютера, например, отказы системы хранения данных, проблемы с очередями, а также незначительные отклонения в работе климатической и энергетической инфраструктуры. Сбои уровня Warning — локальные проблемы на узлах.

Для контроля работы суперкомпьютера «Чебышёв» в настоящий момент используется около 160 правил. Вот некоторые из них:

- баланс счета GSM-модема близок к порогу отключения;
- сбои в работе двух или трех холодильных установок;
- значительный рост ошибок на сетевых интерфейсах;
- слишком малое количество пользовательских сессий на головной машине;
- слишком большое число заблокированных узлов;
- рассинхронизация времени на узлах;
- значение LoadAVG на «свободном» узле превышает значение 3.

4. Методы реагирования на сбои

Если правило определило сбой в работе суперкомпьютера, система может выполнить некоторую реакцию. По умолчанию для каждого события производится запись о нем в лог-файл и отправка e-mail администраторам. Информация о событиях уровня Critical дублируется по SMS. В критических случаях (срабатывание пожарной сигнализации, резкое повышение температуры в помещении, низкий уровень заряда аккумуляторных батарей при питании от них) суперкомпьютер может быть автоматически выключен. В случае недоступности вычислительных узлов по протоколу ssh или неработоспособности на них сервисов MPI они автоматически выводятся системой из счетного поля.

```
*** reported events ***

13, DANGER, "ntpd drift on node is too big"
2, DANGER, "ems sensor: front temp is very high"

4, WARNING, "bad system temp on node"
1, WARNING, "zombies present on node for last 1000 seconds"

13, RECOVER, "ntpd drift on node is ok"
3, RECOVER, "system temp on node is ok"
2, RECOVER, "ems sensor: front temp is back to normal"

*** suppressed events ***

8, DANGER, "too many free cpus"
```

Рис. 5. Пример дайджеста событий за сутки

На практике множество сбоев, обнаруживаемых системой Octotron в ходе работы суперкомпьютера, не имеют критического характера и позволяют продолжать штатное функционирование вычислительного комплекса. Чтобы уменьшить интенсивность потока сообщений администраторам суперкомпьютера, в системе предусмотрена возможность блокировки повторяющихся оповещений. Она применяется в абсолютно типичных ситуациях: проблема не является критичной, о ней известно администраторам, но по каким-либо причинам устранить ее не удается.

Еще один полезный для администратора суперкомпьютера сервис – ежедневный дайджест событий, рассылаемый по e-mail. На рис. 5 приведен пример дайджеста с перечнем событий, случившихся на суперкомпьютере «Чебышёв» 11 ноября 2014 г. В секции «reported events» перечислены актуальные обнаруженные сбои, в секции «suppressed events» – сбои, для которых была отключена отправка отдельных сообщений.

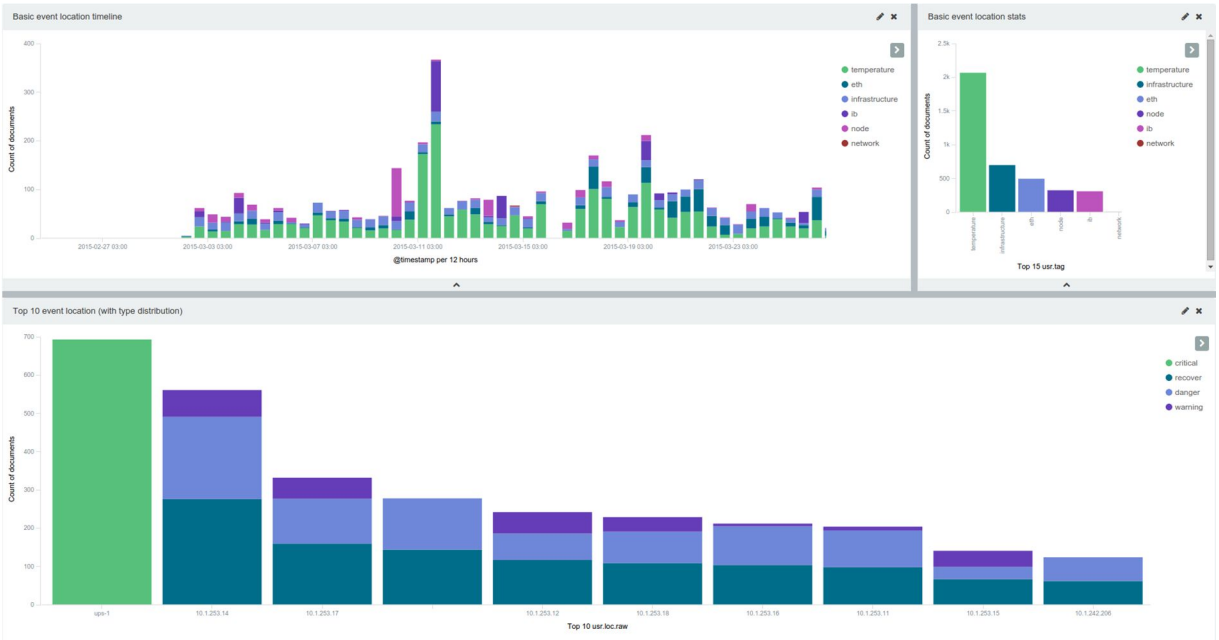


Рис. 6. Графическое представление потока событий, фиксируемого системой Octotron.

Лог-файл системы Octotron ведется в формате JSON. Это позволяет использовать внешние утилиты для его обработки. С помощью набора инструментов logstash, elasticsearch и kibana было реализовано средство построения статистики и базового визуального анализа потока событий, фиксируемого системой Octotron (рис. 6). Левый верхний график отображает число событий различного типа за интервал времени (каждому типу событий соответствует свой цвет), правый верхний — общее число событий различных типов в исследуемом интервале, нижний график — распределение событий по уровню критичности (цвет) и их источникам (горизонтальная ось).

Заключение

Система Octotron разрабатывается с конца 2012 г., и уже сейчас она успешно эксплуатируется в Суперкомпьютерном комплексе МГУ. На текущий момент она полностью контролирует работу суперкомпьютера «Чебышёв». В тестовом режиме, дублируя штатную систему автоматического отключения оборудования, Octotron работает на суперкомпьютере «Ломоносов».

Развитие системы ведется одновременно в нескольких направлениях. Так, статистика сбоев суперкомпьютера представляет исключительно ценный материал для анализа и прогнозирования его поведения. Методы определения типичных сбоев, выраженные единообразно в терминах модели Octotron, могут отчуждаться и тиражироваться; крайне перспективным направлением представляется создание коллективного банка неисправностей суперкомпьютерных комплексов и методов реагирования на них. Наличие в основе системы модели суперкомпьютера позволяет реализовать различные средства визуализации сбоев для оперативного поиска и устранения неисправностей. Сама по себе разработка модели суперкомпьютера является нетривиальным процессом, который, однако, может быть автоматизирован [8]. Важно, что вся подобная функциональность может быть реализована с помощью подключаемых к системе модулей, в то время как ядро системы Ocotron уже полнофункционально, удовлетворяет заданным при разработке требованиям и выполняет поставленные задачи.

Разработанная система доступна под открытой MIT лицензией [9, 10].

Работа выполнена при финансовой поддержке РФФИ, грант №12-07-33047.

Литература

1. Антонов, А.С. Разработка принципов построения и реализация прототипа системы обеспечения оперативного контроля и эффективной автономной работы суперкомпьютерных комплексов / А.С. Антонов, Вад.В. Воеводин, Вл.В. Воеводин, С.А. Жуматий, Д.А. Никитенко, С.И. Соболев, К.С. Стефанов, П.А. Швец // Вестник УГАТУ. — 2014. — Т. 18, № 2. — С. 227–236.
2. Соболев, С.И. Суперкомпьютер в штатном режиме / С.И. Соболев // Открытые системы. — 2014. — № 8.
3. Швец, П.А. Об одном подходе к моделированию суперкомпьютерных комплексов / П.А. Швец, Вад.В. Воеводин, С.И. Соболев // Научный сервис в сети Интернет: многообразие суперкомпьютерных миров: Труды Международной суперкомпьютерной конференции (22–27 сентября 2014 г., г. Новороссийск). — Изд-во МГУ Москва, 2014. — С. 197–204.

4. HP OpenView. URL: http://www.openview.hp.com/solutions/ams/ams_bb.pdf (дата обращения: 06.04.2015).
5. xCAT, An extreme cluster/cloud administration toolkit. URL: http://sourceforge.net/p/xcat/wiki/Main_Page/ (дата обращения: 06.04.2015).
6. Lu K. et al. Iaso: an autonomous fault-tolerant management system for supercomputers //Frontiers of Computer Science. — 2014. — Т. 8. — №. 3. — С. 378–390. DOI: 10.1007/s11704-014-3503-1
7. Программное обеспечение компании Т-Платформы. URL: http://www.t-platforms.ru/products/software.html (дата обращения: 06.04.2015).
8. Воеводин, Вад В. Автоматическое определение и описание сетевой инфраструктуры суперкомпьютеров / Вад.В. Воеводин, К.С. Стефанов // Вычислительные методы и программирование: Новые вычислительные технологии — 2014. — Т. 15, № 3. — С. 560–568.
9. Полный исходный код Octotron. URL: https://github.com/srcc-msu/octotron_core (дата обращения: 06.04.2015).
10. Рабочее окружение Octotron для создания модели на языке Python. URL: https://github.com/srcc-msu/octotron (дата обращения: 06.04.2015).

Антонов Александр Сергеевич, к.ф.-м.н., ведущий научный сотрудник лаборатории параллельных информационных технологий НИВЦ, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), asa@parallel.ru.

Воеводин Вадим Владимирович, к.ф.-м.н., научный сотрудник лаборатории параллельных информационных технологий НИВЦ, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), vadim@parallel.ru.

Даугель-Дауге Артем Александрович, студент факультета Вычислительной математики и кибернетики, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), daugeldauge@gmail.com.

Жуматий Сергей Анатольевич, к.ф.-м.н., ведущий научный сотрудник лаборатории параллельных информационных технологий НИВЦ, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), serg@parallel.ru.

Никитенко Дмитрий Александрович, к.ф.-м.н., научный сотрудник лаборатории параллельных информационных технологий НИВЦ, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), dan@parallel.ru.

Соболев Сергей Игоревич, к.ф.-м.н., старший научный сотрудник лаборатории параллельных информационных технологий НИВЦ, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), sergeys@parallel.ru.

Стефанов Константин Сергеевич, к.ф.-м.н., старший научный сотрудник лаборатории параллельных информационных технологий НИВЦ, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), cstef@parallel.ru.

Швец Павел Артемович, программист лаборатории параллельных информационных технологий НИВЦ, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), shpavel@parallel.ru.

Поступила в редакцию 9 марта 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 2, pp. 33–43

DOI: 10.14529/cmse150203

SECURING OF ACTIVE CONTROL AND EFFICIENT AUTONOMOUS OPERATING OF MSU SUPERCOMPUTING CENTER

A.S. Antonov, *Research Computing Center, Moscow State University (Moscow, Russian Federation)* asa@parallel.ru,
Vad.V. Voevodin, *Research Computing Center, Moscow State University (Moscow, Russian Federation)* vadim@parallel.ru,
A.A. Daugel-Dauge, *Faculty of Computational Mathematics and Cybernetics, Moscow State University (Moscow, Russian Federation)* daugeldauge@gmail.com,
S.A. Zhumatiy, *Research Computing Center, Moscow State University (Moscow, Russian Federation)* serg@parallel.ru,
D.A. Nikitenko, *Research Computing Center, Moscow State University (Moscow, Russian Federation)* dan@parallel.ru,
S.I. Sobolev, *Research Computing Center, Moscow State University (Moscow, Russian Federation)* sergeys@parallel.ru,
K.S. Stefanov, *Research Computing Center, Moscow State University (Moscow, Russian Federation)* cstef@parallel.ru,
P.A. Shvets, *Research Computing Center, Moscow State University (Moscow, Russian Federation)* shpavel@parallel.ru

At RCC MSU we are working on the system for securing of active control and efficient autonomous operating of supercomputers. This system is been implemented at MSU Supercomputing Center. The paper describes system installation, setting and usage experience for the control of the "Chebyshev" supercomputer.

Keywords: supercomputer, graph, model, monitoring, active control, autonomous operating, Octotron.

References

1. Antonov A.S., Voevodin Vad.V., Voevodin Vl.V., Zhumatiy S.A., Nikitenko D.A., Sobolev S.I., Stefanov K.S., Shvets P.A. Razrabotka printsipov postroeniya i realizatsiya prototipa sistemy obespecheniya operativnogo kontrolya i effektivnoy avtonomnoy raboty superkomp'yuternykh kompleksov [Securing of Reliable and Efficient Autonomous Functioning of Supercomputers: Basic Principles and System Prototype] // Vestnik UGATU [Vestnik UGATU]. 2014. Vol. 18, No. 2. P. 227–236.
2. Sobolev S.I. Superkomp'yuter v shtatnom rezhime [Supercomputer in Regular Conditions] // Otkrytye sistemy [Open Systems]. 2014. No. 8.
3. Shvets P.A., Voevodin V.V., Sobolev S.I. Ob odnom podkhode k modelirovaniyu superkomp'yuternykh kompleksov [On a One Approach to Supercomputers Simulation] //

Nauchnyy servis v seti Internet: mnogoobrazie superkomp'yuternykh mirov: Trudy Mezhdunarodnoy superkomp'yuternoy konferentsii (22–27 sentyabrya 2014 g., g. Novorossiysk) [Internet Services & Internet: Variety of Supercomputing Worlds. International Supercomputing Conference Proceedings, Sep 22–27, 2014, Novorossiysk]. Izd-vo MGU, Moskva [MSU Publishing, Moscow]. 2014. P. 197–204.

4. HP OpenView. URL: http://www.openview.hp.com/solutions/ams/ams_bb.pdf (accessed: 06.04.2015).
5. xCAT, An extreme cluster/cloud administration toolkit. URL: http://sourceforge.net/p/xcat/wiki/Main_Page/ (accessed: 06.04.2015).
6. Lu K. et al. Iaso: an autonomous fault-tolerant management system for supercomputers // Frontiers of Computer Science. 2014. Vol. 8, No. 3. P. 378–390. DOI: 10.1007/s11704-014-3503-1
7. Programmnoe obespechenie kompanii T-Platformy [T-Platforms Software]. URL: http://www.t-platforms.ru/products/software.html (accessed: 06.04.2015).
8. Voevodin Vad V., Stefanov K.S. Avtomaticheskoe opredelenie i opisanie setevoy infrastruktury superkomp'yuterov [Automated Detection and Description of Supercomputer Network Structure] // Vychislitel'nye metody i programmirovanie: Novye vychislitel'nye tekhnologii [Numerical Methods and Programming]. 2014. Vol. 15, No. 3. P. 560–568.
9. Polnyy iskhodnyy kod Octotron [Octotron Full Source Code]. URL: https://github.com/srcc-msu/octotron_core (accessed: 06.04.2015).
1. 10. Rabochee okruzhenie Octotron dlya sozdaniya modeli na yazyke Python [Octotron Environment for Model Creating with Python]. URL: https://github.com/srcc-msu/octotron (accessed: 06.04.2015).

Received March 9, 2015.

УДК 004.75 DOI: 10.14529/cmse150204

ФОРМИРОВАНИЕ И ПЛАНИРОВАНИЕ ПАКЕТОВ ЗАДАНИЙ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СРЕДАХ[1]

В.В. Топорков, Д.М. Емельянов, П.А. Потехин

В работе рассматриваются подходы к формированию системы ранжированных заданий в модели циклического планирования в виртуальных организациях распределенных вычислительных сред. Предлагаются и сравниваются две различные методологии отбора заданий для планирования: первая из них базируется на решении задачи о заполнении ранца, для второй — вводится эвристический показатель «совместимости» заданий и доменов вычислительных узлов. Приводятся результаты экспериментального исследования, позволяющего оценить эффективность предложенных решений, проводится их сравнительный анализ со случайным отбором заданий в пакет.

Ключевые слова: распределенные вычисления, планирование, слот, циклическая схема планирования, пакет заданий, формирование пакета.

Введение

Сложность планирования вычислений в таких средах, как грид, обусловлена динамичностью, географической распределенностью ресурсов, а также разнородностью заданий от пользователей виртуальных организаций (ВО) и требований к их выполнению [1]. Важнейшей задачей планирования в ВО является обеспечение эффективного использования доступных ресурсов с учетом интересов основных участников вычислений: пользователей, владельцев ресурсов и администраторов. В РВС с большим числом участников с различными и, зачастую, противоречащими друг другу, интересами свою эффективность показали подходы к планированию на основе экономических принципов [2–6].

Решения задач планирования можно классифицировать в зависимости от используемого типа диспетчеризации заданий. При децентрализованной диспетчеризации планировщики ресурсов, как правило, работают локально на стороне клиента для реализации интересов конкретного пользователя (AppLes [7], PAUA [8]). Централизованная (иерархическая) диспетчеризация предполагает наличие метапланировщика, обеспечивающего более эффективное использование доступных ресурсов. Метапланировщик РВС в процессе планирования работает с метазаданиями, сопровождающимися ресурсными запросами — спецификациями характеристик вычислительных ресурсов, необходимых для выполнения задания. Иерархическая модель используется, например, в таких системах, как X-Com [9], GrADS [10]. Возможна оценка ресурсных требований заданий на основе статистики или с использованием экспертных систем [11]. Как правило, задача планирования потока заданий решается с использованием стандартных наборов средств и алгоритмов, включающих FCFS (First Come First Served), бэкфиллинг, механизмы вычисления приоритетов пользователей и заданий, и разделения ресурсов [12–14]. Важным аспектом в этих подходах является преимущественное соблюдение очереди и пользова-

[1] Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии – 2015».

тельских приоритетов при выполнении заданий. Еще более «справедливое» формирование очереди [6] основывается на экономических принципах и учитывает особенности отдельных заданий и их влияние на выполнение всей очереди.

Циклическое планирование потока заданий [15] позволяет реализовать политику ВО с использованием различных критериев. Планирование осуществляется на основе динамически обновляющихся данных о состоянии загрузки доступных вычислительных узлов домена РВС. Таким образом, в каждом цикле планирования потока заданий решаются следующие задачи: распределение заданий из глобальной очереди в потоки; формирование системы (пакетов) заданий; планирование системы заданий в соответствии с принятой в ВО политикой. При планировании пакета заданий, интересы ВО, как правило, выше приоритетов отдельных заданий, что позволяет оптимизировать общие параметры выполнения системы заданий. Например, в [16] при планировании пакета заданий решается задача минимизации суммарного потребления электроэнергии. При этом дисциплина очереди может быть и нарушена.

В циклической схеме планирования (ЦСП) [15] предполагается реализация двух этапов: во-первых, поиск для каждого задания нескольких вариантов (альтернатив) выполнения на интервале планирования и, во-вторых, последующий выбор комбинации альтернатив (по одной для каждого задания) по критерию эффективности, принятому в ВО [17]. Наличие нескольких альтернатив выполнения создает возможности для оптимизации плана выполнения пакета независимых заданий. Поиск альтернатив выполнения осуществляется последовательно для каждого задания, что подчеркивает важность приоритетов заданий в пакете. При введении пользовательских критериев оптимизации отдельных заданий можно обеспечить «справедливое» планирование пакета заданий [15, 17]. Однако стоит отметить, что отбор заданий согласно простым приоритетам пользователей может негативно сказаться на эффективности планирования всего потока заданий. Другими словами, для повышения эффективности планирования потока заданий в целом, в соответствии с политикой ВО, необходимо рассмотреть альтернативные способы ранжирования системы заданий.

Данная статья посвящена вопросам формирования системы заданий для ЦСП. Предлагается эвристический коэффициент «совместимости» характеристик заданий и домена вычислительных узлов для принятия решений о распределении потоков заданий между доменами или для отбора заданий в пакет. Предлагаются и сравниваются два различных подхода к формированию пакета: первый базируется на решении известной задачи о заполнении ранца, второй использует показатель «совместимости» заданий с доменами ресурсов. Остальная часть статьи организована следующим образом. В разделе 2 описываются предлагаемые решения для формирования системы заданий. Раздел 3 посвящен экспериментальным исследованиям. В заключении приводятся основные результаты и обсуждаются направления дальнейшей работы.

1. Формирование системы заданий

В ЦСП выделяются следующие основные требования к вычислительным ресурсам: минимально необходимая для выполнения задания производительность p, максимальная суммарная стоимость (бюджет) S выполнения задания, требуемое число n вычислительных узлов, а также время t, на которое необходимо зарезервировать ресурсы (в расчете на ресурсы производительности p). Эта модель опирается на пользовательскую оценку

времени выполнения задания. Использование относительных единиц производительности ресурса позволяет оценить время выполнения задания на ресурсах различной производительности. Время выполнения может сокращаться в зависимости от производительности узлов, на которые назначено задание. Если пользовательская оценка неточна, выполнение задания может быть либо прервано по истечении времени выделения ресурсов, либо ресурсы могут быть освобождены преждевременно. Система независимых заданий в каждом цикле планирования представляет собой пакет определенным образом отобранных из потока заданий от пользователей. Отбор заданий в пакеты позволяет повысить общую эффективность планирования в виртуальной организации по сравнению с планированием каждого задания в отдельности за счет оптимизации общего критерия, формализующего политику ВО, и справедливого разделения ресурсов на основе предпочтений ключевых стейкхолдеров [2–6, 11, 15].

1.1. Ограничение размера пакета

Важным этапом, на первый взгляд, не имеющим прямого отношения к эффективности циклического планирования потока заданий в ВО, является определение размера пакета заданий в каждом цикле. За счет варьирования ограничения на размер пакета заданий, который может быть выражен в количестве заданий, их суммарной стоимости и т.д., можно повысить эффективность планирования согласно одному или нескольким различным критериям. В нашей модели можно выделить следующие критерии эффективности планирования.

- Загрузка вычислительных узлов.
- Критерий оптимизации, формализующий политику ВО. Примером может служить минимизация времени выполнения потока заданий при ограничении на его суммарную стоимость.
- Число альтернатив для выполнения каждого задания. Данный критерий напрямую связан с предыдущим. В ЦСП большее число альтернатив создает больше возможностей для оптимизации планирования.
- Количество циклов планирования, требуемых для выполнения некоторого множества (потока) заданий. Минимизация данного показателя обеспечивает более высокую пропускную способность распределенной вычислительной среды.

Непосредственное задание размера пакета, например, администраторами ВО, нецелесообразно. В условиях когда локальные расписания доступных вычислительных узлов динамически изменяются, а характеристики входящих заданий существенно различаются и базируются на пользовательских, зачастую неточных, оценках, невозможно заранее выбрать ограничение, которое бы позволило повысить эффективность планирования согласно выбранному в ВО критерию. Более гибкий механизм ограничения размера пакета можно построить на основе соотношения требований заданий и характеристик вычислительной среды. В качестве таких характеристик в рамках экономических принципов логично выбрать стоимость использования и время резервирования (предоставления) ресурсов.

При использовании ограничения по времени для каждого задания вычисляется суммарное время занятия слотов в пересчете на ресурс единичной производительности. Для выполнения задания в РВС необходимо выделение набора подходящих слотов, каждый из которых характеризуются временем старта, длительностью и стоимостью использования [17]. Данный набор слотов образует «окно», для которого можно рассчи-

тать суммарное используемое процессорное время и суммарную стоимость. Стоит отметить, что для нормализации их значений дополнительно требуется произвести пересчет на ресурс единичной производительности. Таким образом, суммарное процессорное время можно вычислить как произведение $p \cdot t \cdot n$. Для РВС рассчитывается суммарная длина слотов, также в пересчете на ресурс с производительностью, принятой за единицу. Задания в пакет следует набирать таким образом, чтобы суммарное время занятия слотов не превышало суммарной длины слотов домена РВС, взятой с некоторым коэффициентом limitCoefficient $\in (0; 1]$. Администраторы ВО могут оперировать этим коэффициентом для регулирования процесса выполнения потока заданий в РВС. Ограничение по стоимости аналогично ограничению по времени: максимальный бюджет S выполнения задания задается пользователем в ресурсном запросе. Для РВС в текущем цикле планирования рассчитывается суммарная стоимость доступных для использования слотов.

В отличие от пакета с фиксированным количеством заданий, ограничения по времени и по стоимости позволяют подстраивать размер пакета в условиях динамически изменяющейся загрузки узлов вычислительной среды и разнородности потока заданий. Для этого возможно задавать значение ограничивающего коэффициента limitCoefficient $\in (0; 1]$. Далее, в разделе 3, проводится экспериментальное исследование этого подхода.

1.2. Показатель совместимости задания и вычислительной среды

Схемы формирования пакета, предлагаемые в данной работе, производят выбор заданий в пакет на основании совместимости характеристик задания и вычислительной среды. Таким образом, в пакет отбираются задания, которые по ресурсным запросам наилучшим образом подходят для выполнения в текущем интервале планирования. В качестве меры совместимости отдельного задания и домена ресурсов в РВС предлагается эмпирический коэффициент Dq (Distribution quality). Коэффициент Dq характеризует шансы успешного планирования и выполнения задания в текущем состоянии вычислительной среды. Dq может принимать как положительные (высокий шанс выполнения), так и отрицательные значения (низкий шанс выполнения). Для расчета коэффициента Dq и выделения значимых характеристик заданий и доменов вычислительных ресурсов были проведены экспериментальные исследования, основанные на моделировании нескольких тысяч циклов планирования. В результате выявлены следующие характеристики среды и параметры ресурсного запроса, которые оказывают наибольшее влияние на вероятность удачного исхода планирования.

1. Соотношение «цена/качество» узлов среды (Q_0) и заданий (Q). Для отдельного вычислительного узла Q_0 рассчитывается как отношение удельной стоимости использования ресурса к его показателю производительности $\frac{c}{p}$. Для домена ресурсов берется среднее Q_0значение Q_0 по всем узлам. Для отдельного задания показатель формируется аналогично: $Q = \frac{S}{ntp}$.
2. Количество доступных ресурсов (узлов) n_0 в домене РВС и, соответственно, количество вычислительных n узлов, необходимых для выполнения задания.
3. Средняя длина слота l_s в интервале планирования в пересчете на ресурс единичной производительности и требуемая длительность резервирования ресурсов в пересчете на ресурс с базовой производительностью $t \cdot p$.

4. Суммарное процессорное время V_s в расчете на ресурс единичной производительности (суммарная длина всех используемых слотов), и необходимое для выполнения задания процессорное время $t \cdot p \cdot n$.

Коэффициент Dq состоит из четырех слагаемых, соответствующих приведенным характеристикам домена РВС и пользовательского задания. Для каждого слагаемого вводятся подстроечные параметры: K_q, K_n, K_l и K_v – весовые коэффициенты слагаемых; C_q, C_n, C_l и C_v – пороговые значения, определяющие величину, при которой для задания будет найдена хотя бы одна альтернатива выполнения. Значения подстроечных параметров могут формироваться на основе статистики предыдущих циклов планирования или экспертной оценки. Коэффициент Dq определяется как сумма следующих слагаемых.

$$Dq_1 = \frac{K_q}{C_q}\left(\frac{Q}{Q_0} - C_q\right) \quad (1)$$

Слагаемое нормализует соотношение коэффициентов Q задания и Q_0 среды.

$$Dq_2 = \frac{K_n}{C_n}\left(C_n - \frac{n}{n_0}\right) \quad (2)$$

Нормализует соотношение количества подходящих вычислительных узлов в домене ресурсов с количеством узлов, необходимых для выполнения задания $\frac{n}{n_0}$.

$$Dq_3 = \frac{K_l}{C_l}\left(C_l - \frac{t \cdot p}{l_s}\right) \quad (3)$$

Данное слагаемое нормализует отношение требуемого заданию времени резервирования ресурсов и средней длины слотов системы.

$$Dq_4 = \frac{K_v}{C_v}\left(C_v - \frac{t \cdot p \cdot n}{V_s}\right) \quad (4)$$

Определяет отношение необходимого для выполнения задания процессорного времени к суммарному процессорному времени в текущем цикле планирования.

Выбирать задания в пакет при помощи коэффициента Dq, представляющего собой сумму (1) – (4), возможно различными способами. Например, отбирать задания с максимальным значением Dq. Однако в таком случае возможна ситуация, когда в первых циклах будут распланированы самые «ценные» задания, и на последующих циклах эффективность планирования резко упадет. Поэтому методики формирования системы заданий, предлагаемые в данной работе, используют другую политику и основаны на отборе заданий с минимальным положительным значением Dq, т.е. самых «проблемных» заданий из тех, которые могут быть успешно выполнены в текущем интервале планирования. Данная политика позволяет сбалансировать выполнение потока заданий в течение множества циклов и обеспечить наиболее эффективное использование ресурсов.

1.3. Методики формирования пакета заданий

В работе предлагаются две принципиально различные методики формирования пакета. Первая из них сводит процесс формирования пакета заданий к решению задачи динамического программирования об оптимальном заполнении ранца. Данный подход представляется наиболее естественным, так как позволяет формализовать процедуру отбора заданий при заранее известных характеристиках заданий и домена РВС. Вторая методика основана на использовании коэффициента Dq и позволяет гибко подстраиваться под динамически изменяющийся состав ресурсов и заданий потока.

Идея использования задачи о рюкзаке при организации планирования не нова, однако в известных подходах [16–18] она чаще всего используется для оптимальной аллокации заданий на неотчуждаемых ресурсах. Нами же предлагается использовать ее для заполнения пакета заданий, в качестве подготовительного этапа перед планированием.

Известно, что задача о рюкзаке формулируется следующим образом. Имеется набор предметов, каждый из которых характеризуется двумя основными параметрами: весом и ценностью. Имеется рюкзак ограниченной вместимости по весу. Задача состоит в том, чтобы собрать рюкзак с максимальной ценностью, соблюдая ограничение по весу. Модель задачи о рюкзаке хорошо подходит для описанной задачи формирования пакета заданий. Весовое ограничение пакета и вес каждого задания могут быть стоимостными или временными, в зависимости от выбранного типа ограничения. Весовое ограничение выбирается на основе суммарных характеристик домена вычислительных ресурсов с некоторым коэффициентом limitCoefficient, как описано в разделе 2.1. Ценность каждого задания предлагается вычислять как $\frac{1}{D_q}$: то есть, чем меньше значение коэффициента Dq, тем больше ценность задания. Данное предположение основано на логике выбора коэффициента Dq, описанной в разделе 2.2. Можно отметить, что задания, ценность которых меньше или равна нулю, никогда не попадут в пакет, так как они не вносят положительного вклада в суммарную ценность пакета, но при этом используют часть «полезного веса».

Другой подход, рассматриваемый в работе, предполагает формирование пакета последовательно по одному заданию. Подход использует коэффициент Dq, но данный коэффициент изменен, так чтобы учитывать задания, уже находящиеся в пакете. Например, в случае использования ограничения по времени, слагаемое Dq_4 вычисляется следующим образом:

$$Dq_4 = \frac{K_v}{C_v}\left(C_v - \frac{t \cdot p \cdot n + \sum_{i=1}^{N} t_i \cdot p_i \cdot n_i}{V_S'}\right) \quad (5)$$

Сумма в (5) берется по всем N заданиям, уже отобранным в пакет, а V_S' – суммарное процессорное время для всех задний. В случае, если используется ограничение по стоимости, в числителе этого отношения присутствует суммарный бюджет выполнения уже отобранных заданий пакета, а в знаменателе – суммарная стоимость всех используемых слотов. Таким образом, самый высокий приоритет имеют задания с минимальным положительным значением коэффициента Dq. При увеличении количества заданий в пакете значение Dq_4 уменьшается и может принимать отрицательные значения. Формирование пакета продолжается до тех пор, пока в потоке остаются задания с положительным значением коэффициента Dq. Следует отметить, что в данной методике формирования пакета ограничивающим коэффициентом, которым оперируют администраторы ВО, является подстроечный коэффициент C_v.

В отличие от методики формирования пакета на основе решения задачи о рюкзаке, ограничение размера пакета для данной методики не является жестким. При решении задачи о рюкзаке ограничение является строгим и не может быть нарушено. При использовании данной методики превышение ограничения приведет к тому, что Dq_4 примет отрицательное значение, однако сам коэффициент Dq может остаться положительным, и тогда задание попадет в пакет. Отметим, что для обеих описанных методик формирования пакета используется ранжирование заданий в соответствии с коэффициентом Dq по убыванию, т.е. в начало пакета помещаются самые «проблемные» задания. Это позволяет улучшить показатели планирования.

2. Экспериментальные исследования

Эффективность предложенных методик исследуется с использованием симулятора PBC [19]. Исследуются следующие схемы формирования пакетов заданий:

1. Random — в пакет попадает фиксированное число заданий, которые выбираются из потока случайным образом;
2. KnapsackT — пакет заданий формируется на основе решения задачи о рюкзаке, при этом используется ограничение по времени;
3. KnapsackC — задания в пакет отбираются на основе решения задачи о рюкзаке с ограничением по стоимости;
4. MDqT — формирование пакета с использованием коэффициента Dq, вводится ограничение по времени;
5. MDqC — формирование пакета на основе коэффициента Dq с ограничением по стоимости.

Планирование потока в каждом эксперименте осуществляется циклично: перед началом цикла планирования из потока отбирается пакет заданий, а затем происходит его планирование и выполнение. Задания, которые не удастся выполнить в данном цикле планирования, возвращаются в поток. Циклический процесс планирования продолжается до тех пор, пока все задания потока не будут выполнены. Исследована эффективность планирования при различных значениях ограничивающего коэффициента limitCoefficient. Для схемы Random размер пакета выбирался в соответствии со средним значением размера пакетов, набранных с помощью остальных подходов.

В таблице приведены значения ограничений, введенных при моделировании.

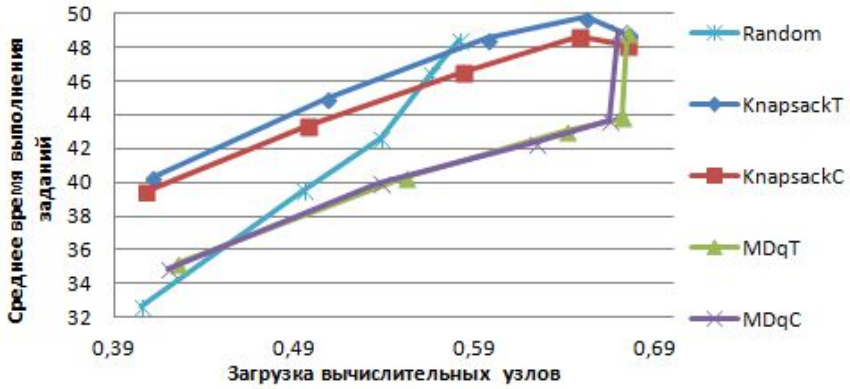
Таблица

Параметры схем формирования пакета в зависимости от серии экспериментов

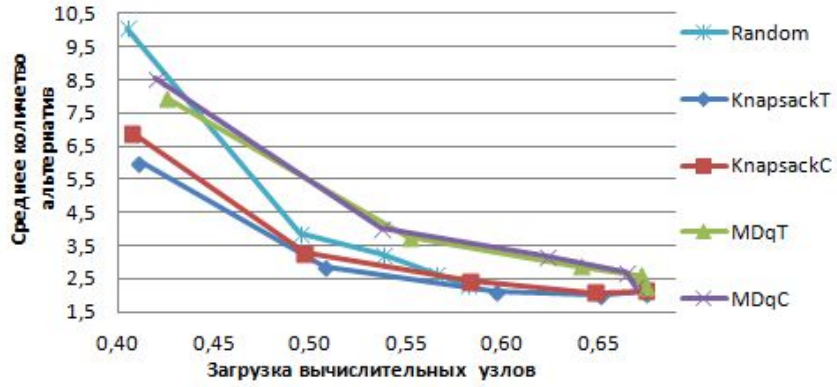
№ серии экспериментов	**1**	**2**	**3**	**4**	**5**
Размер пакета для схемы Random	6	20	30	40	50
Значение limitCoefficient для схем KnapsackT, KnapsackC, MDqT, MDqC	0,1	0,3	0,5	0,7	0,9

Оценим эффективность рассматриваемых схем формирования пакета по следующим критериям: уровень загрузки вычислительных узлов, время выполнения задания, количество альтернатив выполнения задания, число циклов планирования, требуемых для выполнения потока заданий.

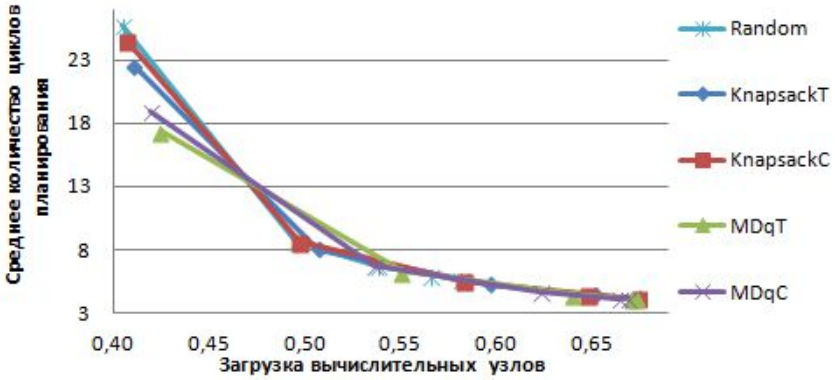
На рис. 1 (а, б, в) представлены соответственно время выполнения, количество альтернатив и среднее число циклов планирования в зависимости от загрузки узлов. За счет наличия нескольких слагаемых в показателе Dq даже при превышении ограничения в слагаемом Dq_4 (по стоимости или по времени) из-за заполненности пакета, коэффициент может сохранять положительное значение. В результате этого в схемах MDqT и MDqC в пакет попадает большее число заданий, которые могут быть успешно спланированы. Это объясняет лучшие значения критериев и более высокий уровень загрузки вычислительных узлов, полученные при использовании данного подхода. Из рис. 1 (а) видно, что схемы MDqT и MDqC обеспечивают преимущество над KnapsackT и KnapsackC по времени выполнения заданий (критерий ВО) в среднем на 15% при примерно равном количестве циклов планирования.



(а)



(б)



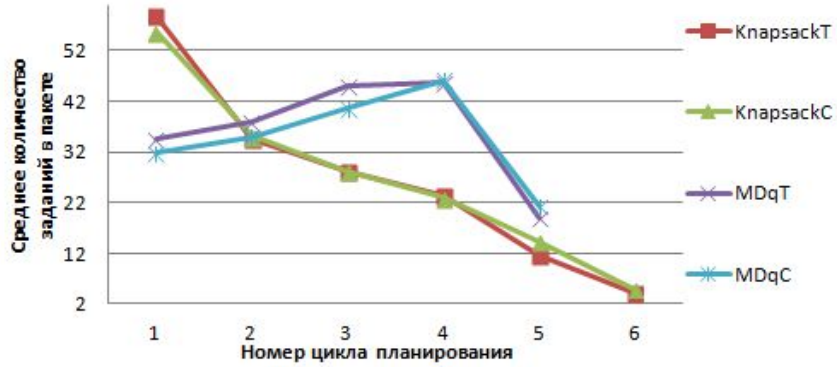
(в)

Рис. 1. Средние значения времени выполнения (а), количества альтернатив (б), числа циклов планирования (в) в зависимости от средней загрузки узлов

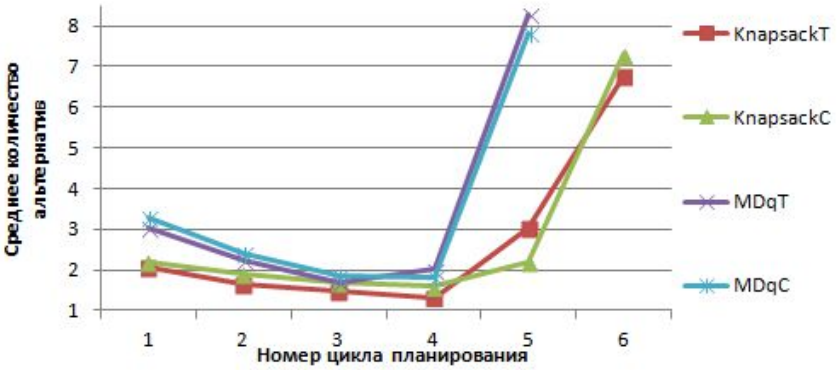
Как можно видеть из рис. 1 (а), почти на всем рассматриваемом интервале по оси загрузки ресурсов лучший результат обеспечивают схемы MDqT и MDqC. Графики на рис. 1 (б) подтверждают результаты, представленные на рис. 1 (а): подходы, обеспечившие наилучшие значения критерия BO, позволяют получить и большее количество альтернатив для выполнения заданий пакета. Средние значения количества циклов планирования в зависимости от загрузки вычислительных узлов (рис. 1 (в)) в этих схемах примерно одинаковы.

На рис. 2 приведены средние значения числа альтернатив выполнения задания, количества заданий в пакете и числа возвратов в поток для рассматриваемых подходов по циклам планирования в серии экспериментов № 3 (табл. 1). Как было отмечено выше, схемы с использованием коэффициента Dq отбирают задания с минимальным положи-

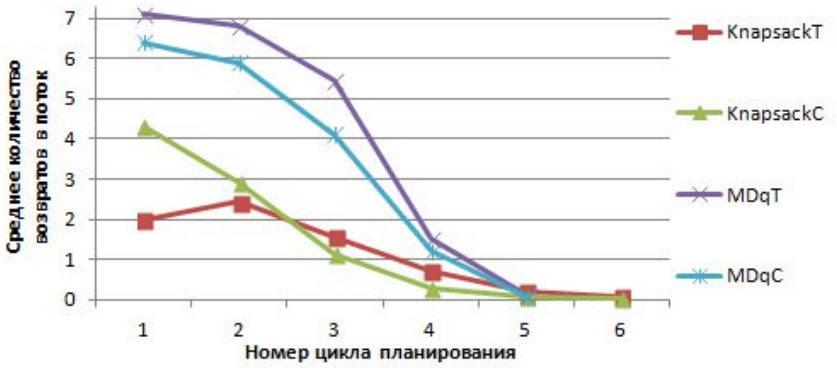
тельным значением этого коэффициента. При этом «вес» задания (необходимое ему процессорное время или бюджет выполнения) не принимается в расчет: задания отбираются из потока равным образом в соответствии со значением коэффициента Dq, пока не будет превышено ограничение, заданное слагаемым (4).



(а)



(б)



(в)

Рис. 2. Средние значения количества альтернатив на задание (а), заданий в пакете (б) и возвратов в поток (в) в зависимости от цикла планирования

С другой стороны, подход KnapsackT(C) максимизирует сумму значений Dq, используя строгое весовое ограничение. Оказывается, что данная политика выбирает задания, требующие относительно меньшее количество ресурсов, т.к. такие задания вносят меньший вклад в общий вес пакета. Поэтому планирование потока происходит неравномерно: на первых циклах планирования отбираются относительно небольшие задания, тогда как задания, требующие больше ресурсов, остаются в потоке и планируются последними (что можно видеть на рис. 2). На первых циклах KnapsackT(C) формирует пакеты большего размера, а на последующих циклах с тем же весовым ограничением размер

пакета уменьшается в несколько раз. Аналогично можно объяснить количество альтернатив, найденных подходом KnapsackT(C). На первых циклах планирования большое число относительно небольших заданий конкурируют за ограниченные ресурсы, что приводит к фрагментации ресурсов. Фрагментация усложняет задачу поиска альтернатив на первых циклах планирования, а на последующих циклах становится сложным выделить даже небольшое число альтернатив для заданий, т.к. оставшиеся задания требуют больше ресурсов. С другой стороны, планирование на основе подхода MDqT(C) происходит более равномерно: на рис. 2 (а) видно, что значения графиков MDqT(C) относительно меньше изменяются от цикла к циклу. Резкое уменьшение числа возвратов в последних циклах планирования для данного подхода можно объяснить тем, что в потоке остаются задания с относительно более высоким значением Dq (т.е. более «благоприятные» задания), и шанс удачного планирования для таких заданий выше.

Полученные результаты позволяют сделать вывод о том, что для схем на основе решения задачи о рюкзаке лучшие значения рассматриваемых критериев получены с использованием ограничения по стоимости. С другой стороны, для подходов MDqT(C) принципиальных различий в зависимости от типа ограничения не отмечается.

Схема случайного формирования пакета Random в некоторых экспериментах обеспечивает наилучшее значение таких критериев, как время выполнения заданий или количество альтернатив выполнения. Однако, объясняется это тем, что пакет заданий набирается без учета «совместимости» заданий и состояния домена ресурсов, а единственным ограничением является предварительно заданный размер пакета.

При фиксированном размере пакета задания группируются неоднородно: их суммарные показатели могут существенно быть больше или меньше ограничения, используемого, например, в KnapsackT(C). Как следствие, получается относительно низкий уровень загрузки вычислительных узлов и большее число циклов планирования, необходимых для выполнения всего потока. Из этого можно сделать вывод о неэффективном использовании ресурсов данной схемой.

Заключение

В работе исследована проблема формирования системы заданий при планировании ресурсов в виртуальной организации распределенной вычислительной среды. Для повышения эффективности планирования в рамках циклической схемы предложен общий вид коэффициента «совместимости» Dq задания и домена РВС. Предложены и исследованы две методики формирования пакета заданий, основанные на принципиально различных подходах к отбору заданий на основе Dq.

Одна из предложенных методик KnapsackT(C) формирует пакет на основе решения задачи о рюкзаке для предварительно вычисленного значения Dq для каждого задания и имеет жесткое ограничение на суммарное время или стоимость выполнения заданий пакета.

Другой подход MDqT(C) использует для отбора заданий коэффициент Dq, который динамически изменяется в зависимости от характеристик заданий, уже находящихся в пакете, и использует более мягкое ограничение на размер пакета заданий.

Результаты эксперимента демонстрируют значительное преимущество эвристического подхода MDqT(C) над подходом на основе решения задачи о рюкзаке KnapsackT(C). Объясняется это, прежде всего, тем, что в условиях динамически изме-

няющегося состава и локального расписания вычислительных узлов, а также разнородности заданий потока, важно учитывать как можно больше динамических параметров состояния среды. Таким образом, мягкое ограничение на размер пакета и динамично изменяемое значение коэффициента Dq позволяет подходу MDqT(C) обеспечить наилучшие значения рассматриваемых критериев эффективности выполнения потока заданий в РВС.

Следует отметить, что обе предложенные методики обеспечили лучшие результаты планирования и большую эффективность использования доступных ресурсов по сравнению с традиционной схемой формирования пакета заданий случайным образом.

Дальнейшие исследования будут посвящены вопросам формирования системы заданий с учетом предпочтений всех участников вычислений ВО.

Работа выполнена при содействии Совета по грантам Президента Российской Федерации для поддержки молодых российских ученых и ведущих научных школ (шифры НШ-362.2014.9, МК-4148.2015.9), РФФИ (проекты 15-07-02259, 15-07-03401), Минобрнауки России, задание № 2014/123 на выполнение государственных работ в сфере научной деятельности в рамках базовой части государственного задания (проект 2268).

Литература

1. Foster I. The Anatomy of the Grid: Enabling Scalable Virtual Organizations / I. Foster, C. Kesselman, S. Tuecke // International J. Supercomputer Applications. — 2001. — Vol. 15, No. 3. — P. 200–220. DOI: 10.1177/109434200101500302
2. Garg S.K. Scheduling Parallel Applications on Utility Grids: Time and Cost Trade-off Management / S.K. Garg, R. Buyya, H.J. Siegel // 32nd Australasian Computer Science Conference, Wellington, New Zealand, Proceedings. — 2009. — Vol. 91. — P. 151–159.
3. Buyya R. Economic Models for Resource Management and Scheduling in Grid Computing / R. Buyya, D. Abramson, J. Giddy // J. Concurrency and Computation. — 2002. — Vol. 14, No. 5. — P. 1507–1542. DOI: 10.1002/cpe.690
4. Топорков В.В. Экономическая модель планирования и справедливого разделения ресурсов в распределенных вычислениях / В.В. Топорков, Д.М. Емельянов // Программирование. — 2014. — № 1. — С. 54–65.
5. Мультиагентный подход к управлению распределенными вычислениями в кластерной GRID-системе / В.Г. Богданова, И.В. Бычков, А.С. Корсуков // Известия РАН. ТиСУ. — 2014. — № 5. — С. 95-105. DOI: 10.7868/s0002338814040039
6. Mutz A. Eliciting Honest Value Information in a Batch-queue Environment / A. Mutz, R. Wolski, J. Brevik // 2007 8th IEEE/ACM International Conference on Grid Computing. — 2007. — P. 291–297. DOI: 10.1109/GRID.2007.4354145
7. Adaptive Computing on the Grid Using AppLeS / F. Berman, R. Wolski, H. Casanova et al. // IEEE Trans. On Parallel and Distributed Systems. — 2003. — Vol. 14, No. 4. — P. 369-382. DOI: 10.1109/TPDS.2003.1195409
8. Scheduling in Bag-of-task Grids: The PAUÁ Case / W. Cirne, F. Brasileiro, L. Costa et al. // 16th Symposium on Computer Architecture and High Performance Computing. — 2004. — P. 124–131. DOI: 10.1109/SBAC-PAD.2004.37

9. Эволюция системы метакомпьютинга X-Com / Вл.В. Воеводин, Ю.А. Жолудев, С.И. Соболев, К.С. Стефанов // Вестник Нижегородского университета им. Н.И. Лобачевского. — 2009. — № 4. — С. 157–164.
10. Scheduling in the Grid Application Development Software Project / H. Dail, O. Sievert, F. Berman et al. // Grid resource management. State of the Art and Future Trends. / Eds J. Nabrzyski, J.M. Schopf, J. Weglarz. — Kluwer Acad. Publ., 2003. — P. 73–98.
11. Multi-criteria Grid Resource Management Using Performance Prediction Techniques / K. Kurowski, A. Oleksiak, J. Nabrzyski et al. // Integrated Research in GRID Computing. / Eds. S. Gorlatch, M. Danelutto. — Springer, 2007. — P. 215–225.
12. Moab HPC Suite Enterprise Edition. URL: http://www.adaptivecomputing.com/products/hpc-products/moab-hpc-suite-enterprise-edition (дата обращения: 08.02.2014).
13. Workload Management with LoadLeveler. / S. Kannan, M. Roberts, P. Mayes et al. — IBM, First ed., 2001. — 210 p.
14. Tsafrir D. Backfilling Using System-generated Predictions Rather than User Runtime Estimates / D. Tsafrir, Y. Etsion, D. Feitelson // IEEE Transactions on Parallel and Distributed Systems. — 2007. — Vol. 18, No. 6. — P. 789–803.
15. Preference-Based Fair Resource Sharing and Scheduling Optimization in Grid VOs / V. Toporkov, A. Toporkova, A. Tselishchev et al. // Procedia Computer Science. — 2014. — Vol. 29. — P. 831–843. DOI: 10.1016/j.procs.2014.05.075
16. Reducing Energy Costs for IBM Blue Gene/P via Power-Aware Job Scheduling / Z. Zhou, Z. Lan, W. Tang, N. Desai // Seventeenth Workshop on Job Scheduling Strategies for Parallel Processing. — May 2013. — P. 96–115.
17. Slot Selection Algorithms in Distributed Computing / V. Toporkov, A. Toporkova, A. Tselishchev, D. Yemelyanov // Journal of Supercomputing. — 2014. — Vol. 69, No. 1. — P. 53–60. DOI: 10.1007/s11227-014-1210-1
18. Soner S. Integer Programming Based Heterogeneous CPU-GPU Cluster Scheduler for SLURM Resource Manager / S. Soner, C. Özturan // Fourteenth IEEE International Conference on High Performance Computing and Communication & Ninth IEEE International Conference on Embedded Software and Systems. — June 2012. — P. 418–424.
19. Методы и эвристики планирования в распределенных вычислениях с неотчуждаемыми ресурсами / В.В. Топорков, А.В. Бобченков, Д.М. Емельянов, А.С. Целищев // Вестник ЮУрГУ, серия «Вычислительная математика и информатика». — 2014. — Т. 3., № 2. — С. 43–62.

Топорков Виктор Васильевич, д.т.н., профессор, заведующий кафедрой Вычислительной техники, Национальный исследовательский университет «МЭИ» (Москва, Российская Федерация), ToporkovVV@mpei.ru.

Емельянов Дмитрий Михайлович, к.т.н., ассистент, кафедра Вычислительной техники, Национальный исследовательский университет «МЭИ» (Москва, Российская Федерация), YemelyanovDM@mpei.ru.

Потехин Петр Анатольевич, аспирант, кафедра Вычислительной техники, Национальный исследовательский университет «МЭИ» (Москва, Российская Федерация), PotekhinPA@mpei.ru.

Поступила в редакцию 17 марта 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 2, pp. 44–57

DOI: 10.14529/cmse150204

JOB BATCH GENERATION AND SCHEDULING IN DISTRIBUTED COMPUTING ENVIRONMENTS

V.V. Toporkov, National Research University "MPEI" (Moscow, Russian Federation) ToporkovVV@mpei.ru,

D.M. Yemelyanov, National Research University "MPEI" (Moscow, Russian Federation) YemelyanovDM@mpei.ru,

P.A. Potekhin, National Research University "MPEI" (Moscow, Russian Federation) PotekhinPA@mpei.ru

The paper considers approaches to ranked jobs system generation in the model of cyclic scheduling in the virtual organizations of distributed computing environments. Two different methodologies of job selection for scheduling are proposed and compared: the first one is based on the solution of knapsack problem, the second one utilizes a heuristic "compatibility" indicator of jobs and computing domains. Experimental results that allow estimating of the proposed solutions efficiency are presented, the solutions are compared to random job selection.

Keywords: distributed computing, scheduling, slot, cyclic scheduling scheme, job batch, job generation.

References

1. Foster I, Kesselman C., Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations // International J. Supercomputer Applications. 2001. Vol. 15, No. 3. P. 200–220. DOI: 10.1177/109434200101500302
2. Garg S.K., Buyya R., Siegel H.J. Scheduling. Parallel Applications on Utility Grids: Time and Cost Trade-off Management // 32nd Australasian Computer Science Conference, Wellington, New Zealand, Proceedings. 2009. Vol. 91. P. 151–159.
3. Buyya R., Abramson D., Giddy J. Economic Models for Resource Management and Scheduling in Grid Computing // J. Concurrency and Computation. 2002. Vol. 14, No. 5. P. 1507–1542. DOI: 10.1002/cpe.690
4. Toporkov V.V., Yemelyanov D.M. Ekonomicheskaya model' planirovaniya i spravedlivogo razdeleniya resursov v raspredelennykh vychisleniyakh [Economic model of scheduling and fair resource sharing in distributed computing]. Programmirovanie [Programming and Computer Software]. 2014. No. 1. P. 54–65.
5. Bogdanova V.G., Bychkov I.V., Korsukov A.S. Mul'tiagentnyy podkhod k upravleniyu raspredelennymi vychisleniyami v klasternoy GRID-sisteme [Multiagent approach to distributed computing management in a cluster GRID system]. Izvestiya RAN. Teoriya i sistemy upravleniya [Journal of Computer and Systems Sciences International]. 2014. No. 5. P. 95-105. DOI: 10.7868/s0002338814040039
6. Mutz A., Wolski R., Brevik J. Eliciting Honest Value Information in a Batch-queue Environment // 2007 8th IEEE/ACM International Conference on Grid Computing. 2007. P. 291–297. DOI: 10.1109/GRID.2007.4354145

7. Berman F., Wolski R., Casanova H., Cirne W., Dail H., Faerman M., Figueira S., Hayes J., Obertelli G., Schopf J., Shao G., Smallen S., Spring N., Su A., Zagorodnov D. Adaptive Computing on the Grid Using AppLeS // IEEE Trans. On Parallel and Distributed Systems. 2003. Vol. 14, No. 4. P. 369-382. DOI: 10.1109/TPDS.2003.1195409
8. Cirne W., Brasileiro F., Costa L., Paranhos D., Santos-Neto E., Andrade N., De Rose C., Ferreto T., Mowbray M., Scheer R., Jornada J. Scheduling in Bag-of-task Grids: The PAUÁ Case // 16th Symposium on Computer Architecture and High Performance Computing. 2004. P. 124–131. DOI: 10.1109/SBAC-PAD.2004.37
9. Voevodin Vl.V., Zholudev Y.A., Sobolev S.I., Stefanov K.S. Evolyutsiya sistemy metakomp'yutinga X-Com [The evolution of X-Com metacomputing system]. Vestnik Nizhegorodskogo universiteta im. N.I. Lobachevskogo [The bulletin of Lobachevsky State University of Nizhny Novgorod]. 2009. No. 4. P. 157–164.
10. Dail H., Sievert O., Berman F., Casanova H., YarKhan A., Vadhiyar S., Dongarra J., Liu C., Yang L., Angulo D., Foster I. Scheduling in the Grid Application Development Software Project // Grid resource management. State of the Art and Future Trends. Kluwer Acad. Publ., 2003. P. 73–98.
11. Kurowski K., Oleksiak A., Nabrzyski J., Kwiecień A., Wojtkiewicz M., Dyczkowski M., Guim F., Corbalan J., Labarta J. Multi-criteria Grid Resource Management Using Performance Prediction Techniques // Integrated Research in GRID Computing. Springer, 2007. P. 215–225.
12. Moab HPC Suite Enterprise Edition. URL:
13. http://www.adaptivecomputing.com/products/hpc-products/moab-hpc-suite-enterprise-edition (accessed: 08.02.2014).
14. Kannan S., Roberts M., Mayes P., Brelsford D., Skovira J. F. Workload Management with LoadLeveler. IBM, First ed., 2001. 210 p.
15. Tsafrir D., Etsion Y., Feitelson D. Backfilling Using System-generated Predictions Rather than User Runtime Estimates // IEEE Transactions on Parallel and Distributed Systems. 2007. Vol. 18, No. 6. P. 789–803.
16. Toporkov V., Toporkova A., Tselishchev A., Yemelyanov D., Potekhin P. Preference-Based Fair Resource Sharing and Scheduling Optimization in Grid VOs // Procedia Computer Science. 2014. Vol. 29. P. 831–843. DOI: 10.1016/j.procs.2014.05.075
17. Zhou Z., Lan Z., Tang W., Desai N. Reducing Energy Costs for IBM Blue Gene/P via Power-Aware Job Scheduling // Seventeenth Workshop on Job Scheduling Strategies for Parallel Processing. May 2013. P. 96–115. DOI: 10.1007/s11227-014-1210-1
18. Toporkov V., Toporkova A., Tselishchev A., Yemelyanov D. Slot Selection Algorithms in Distributed Computing // Journal of Supercomputing. 2014. Vol. 69, No. 1. P. 53–60.
19. Soner S., Özturan C. Integer Programming Based Heterogeneous CPU-GPU Cluster Scheduler for SLURM Resource Manager // Fourteenth IEEE International Conference on High Performance Computing and Communication & Ninth IEEE International Conference on Embedded Software and Systems. June 2012. P. 418–424.
20. Toporkov V.V., Bobchenkov A.V., Yemelyanov D.M., Tselishchev A.S. Metody i evristiki planirovaniya v raspredelennykh vychisleniyakh s neotchuzhdae-mymi resursami [Scheduling methods and heuristics in distributed computing with non-dedicated resources]. Vestnik YuUrGU, seriya «Vychislitel'naya matematika i informatika» [Bulletin of the South Ural State University. Series “Computational Mathematics and Software Engineering”] 2014. Vol. 3, No. 2. P. 43–62.

Received March 17, 2015.

УДК 004.052.42 DOI: 10.14529/cmse150205

ИНСТРУМЕНТАЛЬНАЯ ПОДДЕРЖКА ФОРМАЛЬНОЙ ВЕРИФИКАЦИИ ПРОГРАММ, НАПИСАННЫХ НА ЯЗЫКЕ ФУНКЦИОНАЛЬНО-ПОТОКОВОГО ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ[1]

М.С. Ушакова, А.И. Легалов

Работа посвящена разработке архитектуры инструментальных средств для поддержки формальной верификации функционально-потоковых параллельных программ на языке Пифагор. Используемый метод формальной верификации — дедуктивный анализ на базе исчисления Хоара. Процесс доказательства корректности программы представляется в виде дерева, каждый узел которого — информационный граф программы в котором дуги размечены формулами на языке спецификации. Корнем дерева является исходная тройка Хоара: информационный граф с предусловием и постусловием. В работе рассматриваются основные преобразования, применяемые к информационному графу программы: разметка дуг, эквивалентное преобразование, расщепление, свертка программы. Посредством данных преобразований исходная тройка модифицируется и в конечном счете сводится к набору формул на языке спецификации, истинность которых будет свидетельствовать о корректности программы. Предложена архитектура системы поддержки формальной верификации функционально-потоковых параллельных программ, которая позволяет строить дерево доказательства. Представлена реализация этой системы, описана ее основная функциональность.

Ключевые слова: функционально-потоковое параллельное программирование, язык программирования Пифагор, верификация программ, инструментальные средства для поддержки формальной верификации.

Введение

В настоящее время наблюдается тенденция к усложнению программного обеспечения и увеличению сложности его отладки и рисков в случае сбоя. становятся трудно отлаживаемыми, содержат больше ошибок, а последствия ошибки в системе могут оказаться чрезвычайно дорогими. В результате, стали активно развиваться методы верификации программ, в частности формальная верификация. Под *формальной верификацией* понимается доказательство корректности программы, которое заключается в установлении соответствия между программой и ее спецификацией, описывающей цель разработки [1]. При этом, соответствие программы ее спецификации устанавливается посредством строгого математического доказательства. Главным преимуществом формальной верификации является возможность формально доказать отсутствие ошибок в программе.

Наиболее универсальным методом формальной верификации является дедуктивный анализ на основе исчисления Хоара [2]. Исчисление Хоара — это расширение какой-либо формальной теории $\mathfrak{I}$ введением в нее формул специального вида, называемых тройками Хоара. *Тройка Хоара* — это программа, к которой приписаны две формулы теории $\mathfrak{I}$, описывающие ограничения на входные переменные и требования к результату выполнения программы. Эти формулы называются *предусловие* и *постусловие* соответственно. $\{\psi\}$, где `Prog` — программа, а φ и ψ — предусловие и постусловия для `Prog`. Также для расширения теории $\mathfrak{I}$ вводится набор аксиом для операторов языка и правила вывода, с помощью

[1] Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии — 2015».

которых из аксиом можно выводить утверждения о свойствах программ, в том числе о свойствах корректности. При этом, расширение теории $\mathfrak{I}$ строится таким образом, чтобы корректность программы вытекала из истинности тройки Хоара для этой программы. Основная идея данного подхода заключается в том, чтобы на базе аксиом, с помощью последовательных применений правил вывода, преобразовать тройку Хоара в формулу теории $\mathfrak{I}$, и доказать истинность формулы в этой теории.

Этот метод применим для произвольных языков программирования, а основным его преимуществом является возможность частичной автоматизации процесса доказательства. Стоит также отметить, что теорий $\mathfrak{I}$ для описания спецификаций программ) для каждого языка программирования аксиомы и правила вывода будут своими, так как строятся на основе семантики этого языка. Значит каждому языку программирования будет соответствовать свое исчисление Хоара, и поэтому сложность доказательства корректности программ тоже будет различной.

В настоящее время достигнуты определенные успехи в практическом применении дедуктивного анализа для верификации последовательных программ.языках программирования Для поддержки этого процесса разработан ряд систем. В качестве примера можно привести верификатор программ на языке C Boogie [3] и систему СПЕКТР [4], а также системы для верификации объектно-ориентированных программ на Java: LOOP [5] и KeY [6].

Развитие методов формальной верификации особенно актуально для параллельного программирования. Дедуктивный анализ не нашел широко применения при верификации параллельных императивных программ из–за высокой сложности процесса. Основной проблемой является резкое увеличение сложности формальной верификации параллельной императивной программы по сравнению с последовательной. Главная причина — необходимо учитывать конфликты из-за ресурсов, например, такие как неправильное совместное использование общей памяти или взаимные блокировки процессов в случае работы с распределенной памятью.

Альтернативой императивного подхода является функционально-потоковая параллельная (ФПП) парадигма и ее реализация — язык программирования Пифагор [7, 8]. из-за ресурсов. Модель вычислений, лежащая в основе языка задает вычисления в автоматически выделяемых бесконечных ресурсах. Это позволяет не учитывать возможные ресурсные конфликты, что облегчает процесс написания функционально-потоковой параллельной программы. Каждая программа — это функция, поэтому в языке отсутствуют переменные и циклы, а операции выполняются по готовности данных. В результате, сложность формальной верификации ФПП программ сравнима со сложностью верификации последовательных программ. Другая важная особенность языка — возможность достичь максимального параллелизма программы за счет того, что параллелизм реализуется на уровне операций. После доказательства корректности такой программы, она может быть перенесена на конкретную архитектуру с конечными ресурсами, при необходимости, с ограничением ее параллелизма.

Для языка Пифагор построено исчисление Хоара [9], позволяющее доказывать корректность программ. В качестве языка спецификации выбрана логика исчисления предикатов первого порядка. программы, к дугам которого привязаны формулы на языке спецификации (дуги размечены формулами), а процесс преобразований троек Хоара сводится к разметке дуг графа формулами, модификациям графа и свертке программы. Тройка Хоара, которой соответствует граф со всеми размеченными дугами может быть преобразова-

на в формулу, тождественная истинность которой будет свидетельствовать о корректности программы. Однако процесс доказательства достаточно трудоемок, так как обычно доказательство истинности исходной тройки сводится к доказательству истинности нескольких преобразованных троек. Необходимость рассматривать большое количество вариантов значительно усложняет процесс доказательства. Поэтому целью данной работы является разработка инструментальных средств, обеспечивающих поддержку формальной верификации ФПП программ.

В разделе 1 вводятся основные понятия и описываются преобразования информационного графа с разметкой. Раздел 2 рассматривается архитектура системы поддержки формальной верификации функционально-потоковых параллельных программ, которая позволяет строить дерево доказательства. Реализация предложенной системы и ее основная функциональность описаны в разделе 3.

1. Преобразования информационного графа с разметкой

Программу на языке Пифагор удобно отображать в виде *информационного графа* — ациклического ориентированного графа, определяющего информационную структуру программы, у которого вершины представляют программо-формирующие операторы, а дуги задают пути передачи информации между вершинами [7]. Информационный граф программы, дуги которого помечены формулами на языке спецификации, будем называть *информационным графом с разметкой* (ИГР). Если в информационном графе размечены только входная и выходная дуги, то граф соответствует тройке Хоара, в которой программа представляется графом, разметка входной дуги есть предусловие, выходной — постусловие. Зададим над информационным графом с разметкой следующие преобразования:

1) разметка дуги;
2) изменение информационного графа программы:
   (a) эквивалентное преобразование,
   (b) расщепление;
3) свертка программы.

Процесс доказательства корректности ФПП программы можно рассматривать как последовательность преобразований исходного информационного графа с разметкой, где под «исходным информационным графом с разметкой» понимается ИГР, которому соответствует исходная тройка Хоара, то есть граф у которого входная дуга размечена заданным пользователем предусловием, выходная — постусловием, а остальные дуги не размечены. Последовательными преобразованиями из исходного информационного графа с разметкой получают множество *полностью размеченных ИГР*, к каждой дуге такого графа приписана одна формула. Эти графы с помощью свертки преобразуются в тройки Хоара, которые можно напрямую преобразовать в формулы на языке спецификации для проверки истинности. Если все полученные формулы истинны, то и исходная тройка истинна, а программа корректна.

1.1. Разметка дуг

Принцип разметки дуг графа формулами (приписывание формул к дугам) описан в [9]. Формула, приписанная к дуге информационного графа, описывает свойства данных, которые передаются по этой дуге.

Разметка дуг информационного графа осуществляется на основе аксиом для встроенных функций и теорем для пользовательских функций с доказанной корректностью. Одну функцию могут описывать несколько аксиом или теорем, поэтому в результате разметки может быть получено несколько новых графов.

На множестве формул, помечающих дуги одного ИГР, введем отношение иерархии, назвав формулу, помечающую дугу (a, b), родительской по отношению к формуле, помечающей дугу (b, c) для узлов a, b, c графа.

При разметке дуг информационный граф программы не изменяется, и в случае, если в результате разметки какой-либо дуги получается несколько новых ИГР, то они будут отличаться между собой только формулой у размечаемой дуги. Поэтому, для компактного отображения, эти ИГР можно объединить в один, у которого дуга будет размечена несколькими формулами одновременно. Отношение иерархии между формулами позволяет затем разделить компактное представление на исходные ИГР.

Проиллюстрируем вышесказанное на примере. На рис. 1.А приведена часть графа некоторой программы — четыре оператора соединены информационными связями. Узел 3 принимает входные данные с узлов 1 и 2, узел 4 принимает данные от узла 3. В начале, дуги в графе не размечены.

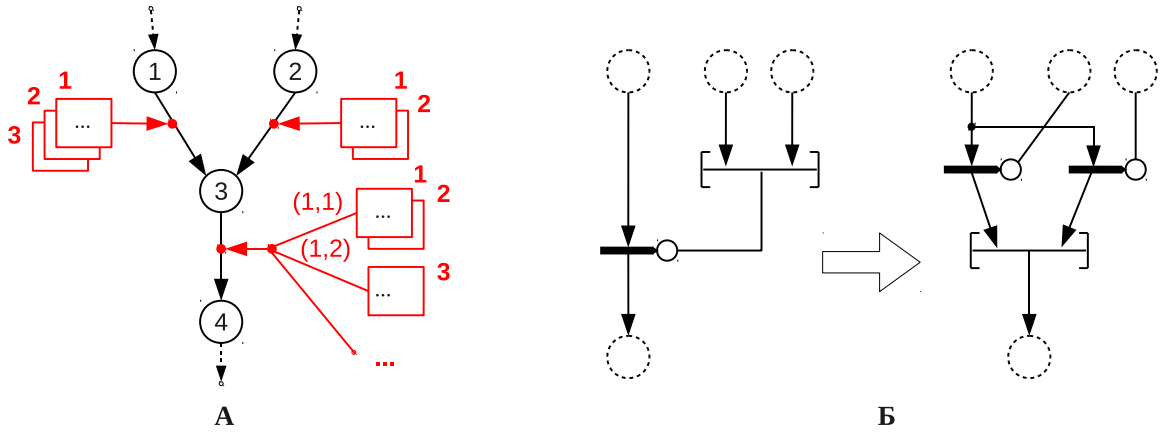


Рис. 1. Схематичное изображение преобразований информационного графа с разметкой. А — часть информационного графа программы, размеченная формулами. Узлы графа (обозначены кружками) представляют операторы программы, дуги — информационные связи, к дугам приписаны формулы (обозначены прямоугольниками), рядом с которыми указаны их номера, в круглых скобках указаны номера родительских формул. Б — эквивалентное преобразование «раскрытие параллельного списка». Параллельный список обозначен квадратными скобками, оператор интерпретации — линией, у которой кружок указывает функциональный вход

На первом шаге разметим дугу $(1, 3)$. Если, при преобразовании графа, для оператора 1 применимо три аксиомы, то в результате преобразования получится три новых ИГР, которые компактно будут отображаться как исходный граф, у которого к дуге $(1, 3)$ приписаны три формулы. На следующем шаге разметим дугу $(2, 3)$. Число применимых аксиом для каждого из трех ИГР первого шага преобразований одинаково, так как операторы 1 и 2 независимы. Если при преобразовании трех ИГР для оператора 2 применимо две аксиомы, то в результате преобразования получится шесть новых графов. В компактном отображе-

нии этих шести графов к дуге $(2,3)$ будут приписаны две формулы. Оператор 3 зависит от операторов 1 и 2, поэтому к каждому из шести полученных ИГР может быть применено разное количество аксиом. Пусть для первого ИГР это число соответствует двум (он распадется на два новых ИГР), для всех остальных — единице. В результате разметки получится семь графов, это число соответствует общему числу применимых аксиом и количеству формул, которые будут приписаны к дуге $(3,4)$. На рис. 1.А индексы (i,j) возле формул, помечающих дугу $(3,4)$, обозначают номер i родительской формулы, помечающей дугу $(1,3)$, и номер j родительской формулы, помечающей дугу $(2,3)$.

1.2. Изменение информационного графа

Второй тип преобразований изменяет информационный граф программы. Эквивалентное преобразование осуществляются по правилам эквивалентных преобразований операторов и связей языка Пифагор (описаны в [8, 10]). В результате преобразования получается один ИГР с измененным информационным графом. В качестве примера эквивалентного преобразования на рис. 1.Б приведено «раскрытие параллельного списка». Параллельный список позволяет явно указывать, что все поступающие в него операторы могут выполняться одновременно. В графе слева параллельный список из двух элементов поступает на функциональный вход оператора интерпретации. Оператор интерпретации имеет два входа для аргумента и функции, и осуществляет функциональное преобразование аргумента. В эквивалентном ему графе справа каждый элемент исходного параллельного списка поступает на функциональный вход своего оператора интерпретации, а уже результат их выполнения передается в параллельный список.

Другой вариант преобразований второго типа — расщепление, оно приводит к получению двух и более ИГР с измененными информационными графами. Расщепление можно использовать для упрощения доказательства, например, при разметке оператора выбора, если точно известны все варианты из которых делается выбор. Это позволяет упростить не только информационный граф программы, но и формулы разметки.

1.3. Свертка

Третий тип преобразований — свертка программы. Свертка проводится над размеченной дугой (кроме входной и выходной дуг программы). Весь порожденный подграф с узлами, из которых достижимо начало рассматриваемой дуги (кроме входного аргумента), может быть заменен на новую переменную, а формула разметки данной дуги добавляется к предусловию исходного ИГР. При замене части кода на новую переменную программа сократится. Свертку можно проводить при частичной или полной разметке дуг графа программы. Если провести свертку над всеми размеченными дугами ИГР, полученному в результате ИГР будет соответствовать тройка Хоара. Назовем такую свертку *полной*. Когда в ИГР все дуги размечены, в результате полной свертки от всего графа остается одна переменная. Ее свойства описаны в предусловии, описывающем теперь свойства всех данных, передаваемых по дугам исходного графа. Она также является результатом работы программы. Такая программа называется *пустой*, и для определения ее корректности достаточно проверить следование постусловия из предусловия.

В результате всех рассмотренных преобразований ИГР ФПП программы можно сделать следующий вывод: весь процесс доказательства можно представить в виде дерева,

корень которого — исходный ИГР, дочерние узлы получаются из родительских выполнением одного из преобразований 1)–3), а листья — полностью размеченные ИГР, над которыми проводится полная свертка и преобразование в формулы. Будем называть такое дерево — *деревом доказательства* или просто доказательством корректности программы.

2. Обобщенная структура системы поддержки формальной верификации

На рис. 2 приведена общая схема системы для поддержки формальной верификации ФПП программ.

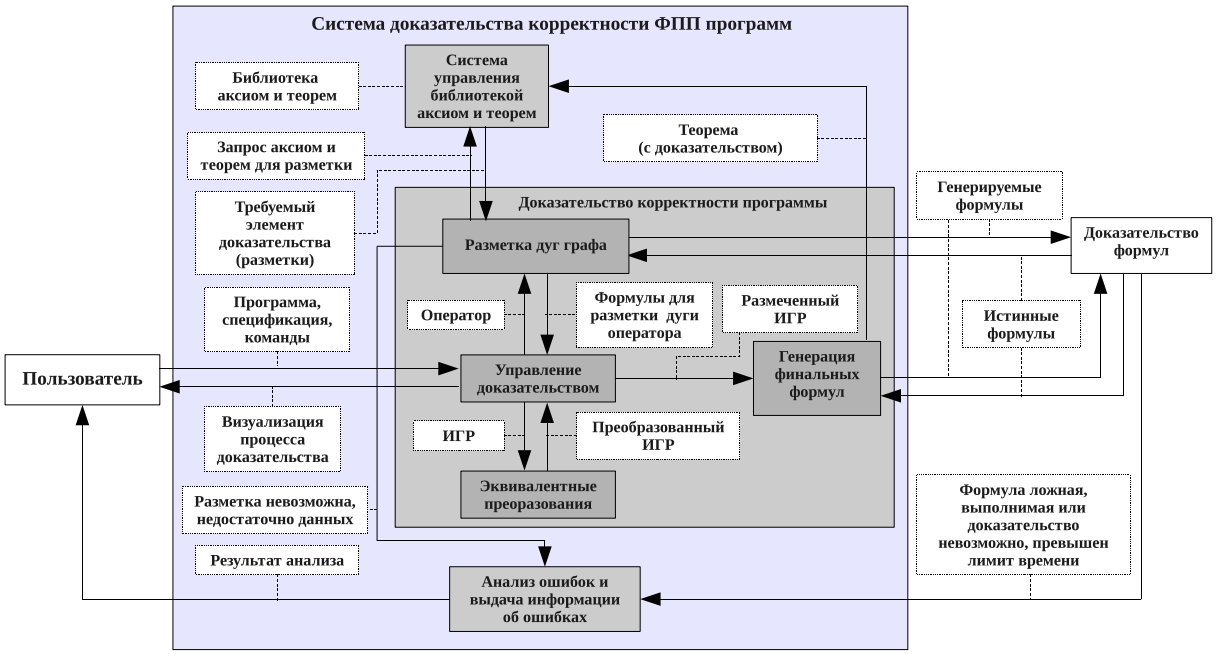


Рис. 2. Обобщенная структура системы поддержки формальной верификации

В системе можно выделить несколько модулей: «Модуль доказательства корректности программы», «Систему управления библиотекой аксиом и теорем» и «Модуль анализа ошибок и выдачи информации об ошибках». «Модуль доказательства формул» (верификатор формул) обособлен от системы, так как является сторонним и может не использоваться при доказательстве, а все его действия будет выполнять пользователь (самостоятельно или используя другой более удобный для него верификатор).

Принцип работы системы состоит в следующем. Пользователь передает системе программу на языке Пифагор и спецификацию программы на языке спецификации. «Модуль доказательства корректности программы» формирует исходный ИГР (которому соответствует исходная тройка Хоара) и начинает процесс доказательства, который заключается в разметке дуг графа программы формулами. Для этого используется информация об аксиомах и уже доказанных теоремах из «Библиотеки аксиом и теорем». «Модуль доказательства корректности программы» посылает запросы к «Системе управления библиотекой аксиом и теорем». В случае если запрашиваемая функция отсутствует в библиотеке, выдается ошиб-

ка о невозможности разметки, которая обрабатывается «Модулем анализа ошибок». Если аксиомы (теоремы) для рассматриваемой функции присутствуют в библиотеке, то «Модуль доказательства корректности программы» проводит их отбор. Для каждой аксиомы (теоремы) из набора аксиом (теорем) для рассматриваемой функции формируется условие применимости данной аксиомы (теоремы) на языке спецификации. Это условие передается верификатору, и, если оно истинно или выполнимо, то аксиома (теорема) используется для разметки, иначе она отбрасывается. После того как все дуги в информационном графе программы будут размечены, проводится полная свертка. Тройки Хоара, соответствующие полученным ИГР, преобразуются в формулы (назовем их *финальными формулами*), которые передаются верификатору для проверки их истинности. Если все формулы истинны, то программа корректна, а ее исходная тройка Хоара — теорема. «Модуль доказательства корректности программы» передает полученную теорему (с доказательством) «Системе управления библиотекой аксиом и теорем», для сохранения в библиотеке. Если истинность формулы не доказана «Модуль анализа ошибок» определяет причину и сообщает об этом пользователю.

Рассмотрим работу «Модуля доказательства корректности программы» более детально (рис. 2). «Блок управления доказательством» является основным, он отвечает за взаимодействие с пользователем, принимает его данные, команды и визуализирует процесс доказательства. Данный блок формирует дерево доказательства. При получении ИГР, он ищет операторы с не размеченными дугами и передает их, в начале, «Блоку эквивалентных преобразований», а затем «Блоку разметки дуг» для получения формул разметки. После того как ИГР становится полностью размеченным, он передается «Блоку генерации финальных формул», который осуществляет полную свертку и генерирует множество финальных формул.

Рассмотренная система может работать в нескольких режимах:

1) полностью ручной;
2) частично автоматизированный;
3) автоматизированный.

В первом случае пользователь использует только «Блок управления доказательством» и «Блок генерации финальных формул», а также «Модуль анализа ошибок». Он сам размечает дуги графа формулами и доказывает истинность финальной формулы. В частично автоматизированном режиме не задействован только «Модуль доказательства формул», и пользователь сам указывает, истинна, ложна или выполнима сгенерированная формула. В автоматизированном режиме задействованы все модули.

3. Реализация системы

В соответствии с рассмотренной обобщенной структурой разработана программа, обеспечивающая поддержку формальной верификации ФПП программ, которая позволяет строить дерево доказательства программы на языке Пифагор.

Пользователю предоставляется графический интерфейс для редактирования дерева доказательства. В качестве входных данных программа принимает формулы разметки в текстовом виде, а код программы — в виде текстового представления реверсивного информационного графа (РИГ) программы [11, 12], который описывает существующие в программе зависимости по данным (отличается от информационного графа тем, что дуги ориентиро-

ванны в противоположном направлении). Также можно загрузить ранее созданный ИГР или дерево доказательства.

Главное окно программы (рис. 3) представляет собой редактор дерева доказательства. В левой его части располагается редактор узлов дерева, а в правой — редактор ИГР, в котором отображается текущий узел дерева. Редактор узлов дерева доказательства позволяет вставлять новые и уже существующие ИГР, копировать, удалять текущие ИГР и вставлять скопированный ИГР, как дочерний к текущему.

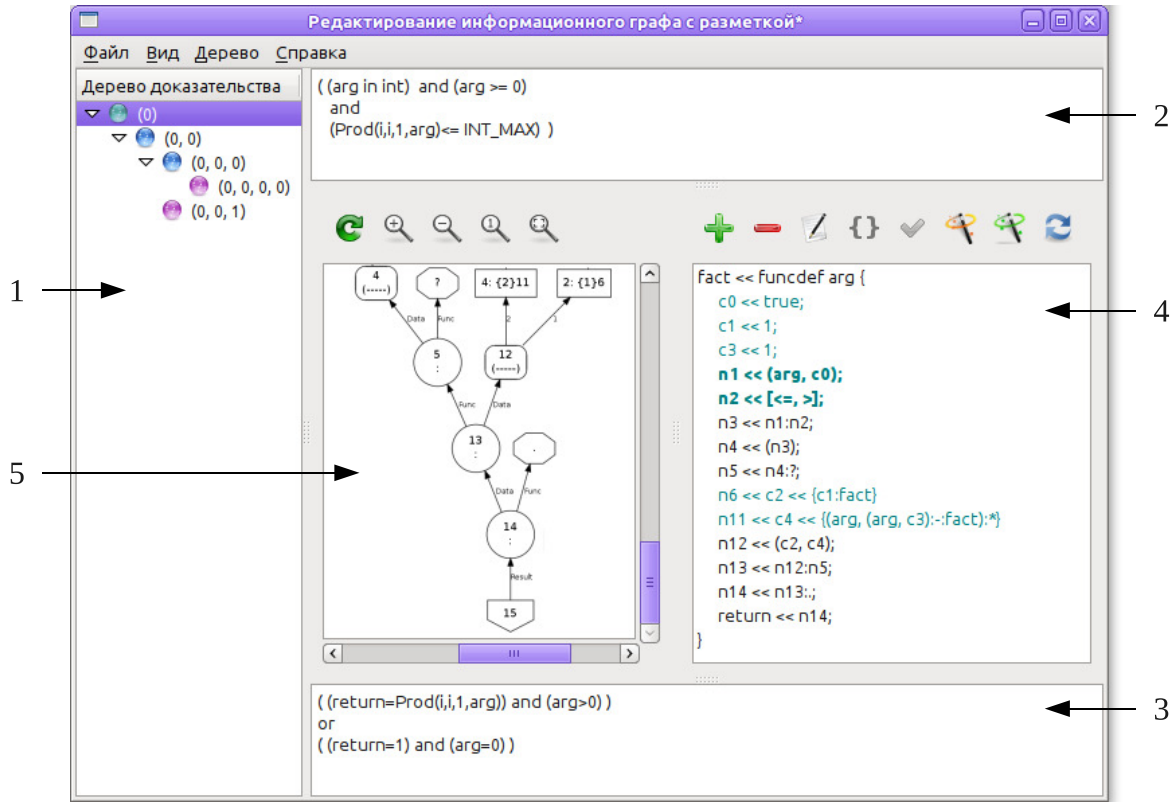


Рис. 3. Главное окно средства инструментальной поддержки формальной верификации функционально-потоковых параллельны программ; 1 — редактор узлов дерева доказательства, 2 — поле ввода предусловия, 3 — поле ввода постусловия, 4 — редактируемый код программы, 5 — графическое представление реверсивного информационного графа программы

Редактор ИГР в своей верхней и нижней части содержит поля для ввода предусловия и постусловия программы соответственно, в левой части отображается графическое представление РИГ, а в правой — редактируемый код программы. Для получения графического представления РИГ используется сторонняя программа GraphViz, которой передается файл формата dot. Редактор кода представлен панелью инструментов и текстом программы, разбитым на строки, где каждая строка соответствует одному узлу РИГ программы. РИГ имеет текстовый формат и создается по программе. Такой РИГ имеет правильный порядок вызовов функций, то есть функция будет вызвана только после того как определена. По такому текстовому представлению однозначно восстанавливается код программы.

Редактор кода программы позволяет добавлять, удалять и редактировать узлы РИГ. При этом, ограничения не позволят пользователю обратиться к оператору до его определе-

ния. Присутствует возможность редактировать задержанные списки. Оператор группировки в задержанный список (или, кратко, задержанный список) содержит подграф программы. Операторы, находящиеся в задержанном списке, не могут выполняться даже при наличии всех аргументов. Их активизация возможна только при снятии задержки, когда ограниченный подграф становится частью всего вычисляемого графа [7, 8]. Редактор кода имеет возможность любой оператор поместить в задержанный список, при этом в задержанный список помещаются все операторы, которые использует данный оператор. Задержанный список рассматривается как константа, поэтому редактирование операторов задержанного списка возможно только после снятия задержки.

Пользователь может вызвать функцию автоматической разметки дуг, либо использовать редактор ИГР для ручной разметки. В редакторе используется компактное отображение получаемых ИГР. Если дуга еще не была размечена, то производится проверка готовности разметки на всех входных дугах. Если хотя бы на одной из входной дуг любого из рассматриваемых в компактном представлении графов разметка отсутствует, то выдается соответствующее сообщение о невозможности разметки текущей дуги. Если все входные дуги размечены, то появляется окно (рис. 4) с иерархией формул дуги текущего узла, где все формулы разделены на группы, соответствующие различным комбинациям формул родительских узлов. Слева представлен список индексов родительских формул, а справа на вкладках размещаются формулы разметки текущей дуги. Пользователь может добавлять новые формулы, что автоматически изменяет индексы всех дочерних узлов (если они уже размечены), у которых появляется необходимое количество пустых вкладок для новых формул. При удалении формул, удаляются все формулы потомков, дочерние по отношению к текущей формуле, и изменяются индексы оставшихся формул.

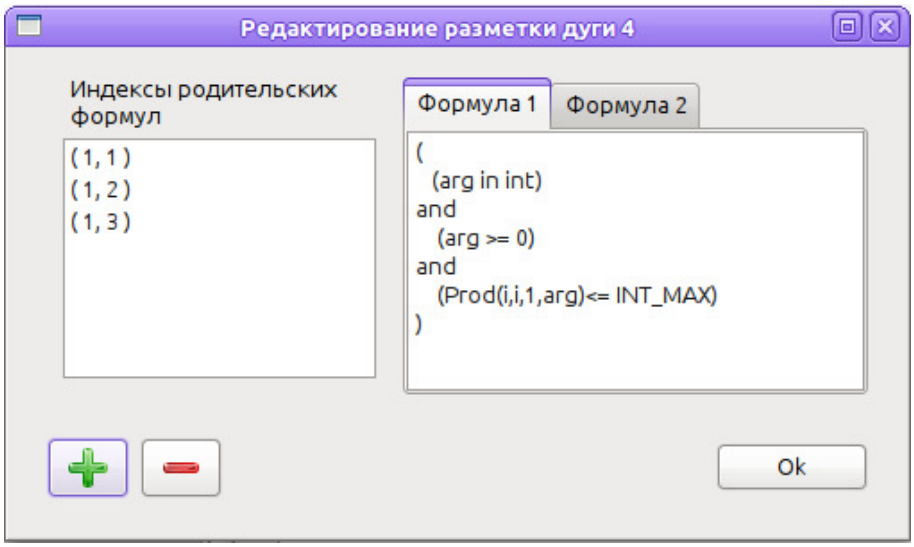


Рис. 4. Окно редактирования иерархии формул размечаемой дуги

Пользователю также доступны следующие функции: автопреобразование, авторазметка ИГР и генерация финальных формул.

Автопреобразование текущего ИГР (автоматическое выполнение эквивалентных преобразований) выполняется в случае, если он является листом дерева доказательства, иначе его дочернее поддерево должно быть удалено. Кроме того, преобразуются только *готовые* операторы, то есть те операторы у которых размечены все входные дуги, а выходная дуга еще не размечена. У всех готовых операторов текущего графа проверяется возможность

применения эквивалентного преобразования. Если к ИГР можно применить эквивалентное преобразование, то новый полученный ИГР становится дочерним к исходному.

Функция автоматической разметки применяется к текущему графу. В начале для него проводятся эквивалентные преобразования, затем находится готовый оператор, для того чтобы провести разметку его выходной дуги. Если оператор является списком, то выходная дуга размечается константой *true*, если это оператор интерпретации, то для функции с его функционального входа, запрашивается список аксиом (для встроенных функций) или теорем (для функций пользователя) в библиотеке аксиом и теорем. Если функция присутствует в библиотеке, то для каждой ее аксиомы или теоремы формируется условие применимости, истинность или выполнимость которого требуется определить пользователю. Аксиомы или теоремы, для которых условие тождественно ложно, отбрасываются, а все оставшиеся используются для разметки дуги рассматриваемого оператора интерпретации несколькими формулами. Далее осуществляется поиск другого готового оператора.

Если все дуги текущего информационного графа размечены, то к нему может быть применена полная свертка, и полученные в результате тройки Хоара с пустой программой можно преобразовать в формулы. Это реализуется функцией генерации финальных формул.

Таким образом, на данный момент реализован ручной и частично автоматизированный режим работы системы.

Заключение

В работе рассмотрены основные преобразования, которые претерпевает информационный граф с разметкой при построении доказательства корректности программы. Разработаны основные концепции архитектуры инструментального средства для поддержки формальной верификации ФПП программ. Задача реализации рассмотренной системы частично решена. В настоящее время ведется работа над расширением функциональности частично автоматизированного и реализацией автоматизированного режима работы системы.

Литература

1. Непомнящий, В.А. Прикладные методы верификации программ / В.А. Непомнящий, О.М. Рякин — М.: Радио и связь, 1988. — 255 с.
2. Hoare, C. A. R. An axiomatic basis for computer programming / C. A. R. Hoare // Communications of the ACM. — 1969. — Vol. 10, No. 12. — P. 576–585. DOI: 10.1145/363235.363259
3. Barnett, M. Boogie: A modular reusable verifier for object-oriented programs / M. Barnett, B.Y.E. Chang, R. Deline, B. Jacobs, K.R.M. Leino // FMCO 2005. LNCS. — 2006. — Vol. 4111. — P. 364–387. DOI: 10.1007/11804192_17
4. Непомнящий, В.А. Верификация С-программ в мультиязыковой системе СПЕКТР / В.А. Непомнящий, И.С. Ануреев, М.М. Атучин, И.В. Марьясов, А.А. Петров, А.В. Промский // Моделирование и анализ информационных систем. — 2010. — № 17 (4). — С. 88–100. DOI: 10.3103/s014641161107011x
5. Van den Berg, J. The LOOP compiler for Java and JML / J. Van den Berg, B. Jacobs // Tools and Algorithms for the Construction and Analysis of Systems. LNCS. — 2001. — Vol. 2031.

—P. 299–312. DOI: 10.1007/3-540-45319-9_21

6. Ahrendt, W. The KeY Tool / W. Ahrendt, T. Baar, B. Beckert, R. Bubel, M. Giese, R. Hähnle, W. Menzel, W. Mostowski, A. Roth, S. Schlager, P.H. Schmitt // Software and System Modeling. — 2005. — Vol. 4 (1). — P. 32–54. DOI: 10.1007/s10270-004-0058-x
7. Легалов, А.И. На пути к переносимым параллельным программам / А.И. Легалов, Д.А. Кузьмин, Ф.А. Казаков, Д.В. Привалихин // Открытые системы. — 2003. — № 5. — С. 36–42.
8. Легалов, А.И. Функциональный язык для создания архитектурно-независимых параллельных программ / А.И. Легалов // Вычислительные технологии. — 2005. — № 1 (10). — С. 71–89.
9. Kropacheva, M.S. Formal Verification of Programs in the Pifagor Language / M.S. Kropacheva, A.I. Legalov // Parallel Computing Technologies (PaCT-2013) 12th International Conference, September 30 - October 4, 2013. Saint-Petersburg, Russia. LNCS. — 2013. — Vol. 7979. — P. 80–89. DOI: 10.1007/978-3-642-39958-9_7
10. Кропачева, М.С. Формализация семантики функционально-потокового языка параллельного программирования Пифагор / М.С. Кропачева // Проблемы информатизации региона (ПИР-2011): Материалы XII Всероссийской научно-практической конференции (Красноярск, 22 – 23 ноября 2011 г.). — Красноярск: Сибирский федеральный университет, 2011. — С. 144–148.
11. Матковский, И.В. Параллельная событийная машина для функционально-потокового языка «Пифагор» / И.В. Матковский // Информационные и математические технологии в науке и управлении: Сборник трудов XVVII Байкальской Всероссийской конференции с международным участием (Иркутск – Байкал, 30 июня – 9 июля 2012 г.). Часть II. — Иркутск: ИСЭМ СО РАН, 2012. — С. 186–193.
12. Легалов, А.И. Событийная модель вычислений, поддерживающая выполнение функционально-потоковых параллельных программ / А.И. Легалов, Г.В. Савченко, В.С. Васильев // Системы. Методы. Технологии. — 2012. — № 1 (13). — С. 113–119.

Ушакова Мария Сергеевна, ассистент кафедры Вычислительной техники института Космических и информационных технологий, Сибирский Федеральный университет (Красноярск, Российская Федерация), ksv@akadem.ru.

Легалов Александр Иванович, д.т.н. профессор и заведующий кафедры Вычислительной техники института Космических и информационных технологий, Сибирский Федеральный университет (Красноярск, Российская Федерация), legalov@mail.ru.

Поступила в редакцию 11 февраля 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 2, pp. 58–70

DOI: 10.14529/cmse150205

A TOOLKIT FOR SUPPORTING FORMAL VERIFICATION OF PROGRAMS IN THE FUNCTIONAL DATA-FLOW PARALLEL PROGRAMMING LANGUAGE

M.S. Ushakova, Siberian Federal University, Institute of Space and Information Technology (Krasnoyarsk, Russian Federation) ksv@akadem.ru,
A.I. Legalov, Siberian Federal University, Institute of Space and Information Technology (Krasnoyarsk, Russian Federation) legalov@mail.ru

The article is devoted to the architecture development of the toolkit for supporting formal verification of functional data-flow parallel programs in the Pifagor Language. The method of deduction based on Hoare logic is used for formal verification. A proof process is considered as a tree where each node is a program data-flow graph, whose edges are marked with formulas in a specification language. The tree root is the initial Hoare triple, namely the program data-flow graph with a precondition and a postcondition. In this article basic transformations of the data-flow graph are considered: edge marking, equivalent transformation, splitting, folding of the program. By means of these transformations the initial triple is being transformed and finally is reduced to a set of formulas in the specification language. If all of these formulas are identically true, then the program is correct. Architecture of the toolkit for supporting formal verification of functional data-flow parallel programs is proposed, which allows to construct a proof three. The implementation of the toolkit is introduced and its main functionality is considered.

Keywords: functional data-flow parallel programming, Pifagor programming language, programs formal verification, toolkit for supporting formal verification.

References

1. Nepomnyaschiy V.A., Ryakin O.M. Prikladnyie metodyi verifikatsii programm [Applied Methods for Programs Verification]. Moscow, Radio i svyaz, 1988. 255 p.
2. Hoare C. A. R. An Axiomatic Basis for Computer Programming // Communications of the ACM. 1969. Vol. 10, No. 12. P. 576–585. DOI: 10.1145/363235.363259
3. Barnett, M., Chang, B.Y.E., Deline, R., et al. Boogie: A Modular Reusable Verifier for Object-Oriented Programs // FMCO 2005. LNCS. 2006. Vol. 4111. P. 364–387. DOI: 10.1007/11804192_17
4. Nepomniaschy V.A., Anureev I.S., Atuchin M.M., et al. C Program Verification in SPECTRUM Multilanguage System // Automatic Control and Computer Sciences. 2011. Vol. 45, No. 7. P. 413–420. DOI: 10.3103/s014641161107011x
5. Van den Berg, J., Jacobs, B. The LOOP compiler for Java and JML // Tools and Algorithms for the Construction and Analysis of Systems. LNCS. 2001. Vol. 2031. P. 299–312. DOI: 10.1007/3-540-45319-9_21
6. Ahrendt, W., Baar, T., Beckert, B., et al. The KeY Tool // Software and System Modeling. 2005. Vol. 4, No.1. P. 32–54. DOI: 10.1007/s10270-004-0058-x

7. Legalov A.I., Kuzmin D.A., Kazakov F.A., et al. Na puti k perenosimyim parallelnyim programmam [An Approach to Portable Parallel Programs] // Otkryityie sistemyi [Open Systems]. 2003. Vol. 5. P. 36–42.
8. Legalov, A.I. Funktsionalnyiy yazyik dlya sozdaniya arhitekturno-nezavisimyih parallelnyih programm [The Functional Programming Language for Creating Architecture-Independent parallel Programs] // Vyichislitelnyie tehnologii [Computational Technologies]. 2005. Vol. 10, No. 1. P. 71–89.
9. Kropacheva, M.S., Legalov A.I. Formal Verification of Programs in the Pifagor Language // Parallel Computing Technologies (PaCT-2013) 12th International Conference, September 30 – October 4, 2013. Saint-Petersburg, Russia. LNCS. 2013. Vol. 7979. P. 80–89. DOI: 10.1007/978-3-642-39958-9_7
10. Kropacheva, M.S. Formalizatsiya semantiki funktsionalno-potokovogo yazyika parallelnogo programmirovaniya Pifagor [Formalization of the Semantics for the Functional Data-Flow Parallel Language Pifagor]. Problemyi informatizatsii regiona (PIR-2011): Materialyi XII Vserossiyskoy nauchno-prakticheskoy konferentsii (Krasnoyarsk, 22 – 23 noyabrya 2011) [The Problems of Region Informatization: 12th Russian Scientific-Practical Conference (Krasnoyarsk, Russia, November, 22 –23, 2011)]. Krasnoyarsk, Publishing of the Siberian Federal University, 2011. P. 144–148.
11. Matkovskiy I.V. Parallelnaya sobyitiynaya mashina dlya funktsionalno-potokovogo yazyika “Pifagor” [Parallel Event-Based Machine for Functional Dataflow Programming Language “Pifagor”]. Informatsionnyie i matematicheskie tehnologii v nauke i upravlenii: Sbornik trudov XVVII Baykalskoy Vserossiyskoy konferentsii s mezhdunarodnyim uchastiem (Irkutsk – Baykal, 30 iyunya – 9 iyulya 2012) [Information and Mathematical Technologies in Science and Management: Proceedings of the 17th Baikal Russian Conference with the International Participant (Irkutsk – Baikal, Russia, June, 30 – July, 9, 2012)]. Irkutsk, Publishing of the Melentiev Energy Systems Institute of Siberian Branch of the Russian Academy of Sciences, 2012. Vol. 2. P. 186–193.
12. Legalov, A.I., Savchenko G.V., Vasilev V.S. Sobyitiynaya model vyichisleniy, podderzhivayuschaya vyipolnenie funktsionalno-potokovyih parallelnyih programm [Computation Event Model Backing the Execution of Functional Data Flow Concurrent Programs] // Sistemyi. Metodyi. Tehnologii [Systems. Methods. Technologies]. 2012. Vol. 1, No. 13. P. 113–119.

Received February 11, 2015.

УДК 681.3.06 DOI: 10.14529/cmse150206

МАСШТАБИРУЕМЫЕ АЛГОРИТМЫ ЦЕЛОЧИСЛЕННОЙ АРИФМЕТИКИ И ОРГАНИЗАЦИЯ ПОДДЕРЖКИ РАЦИОНАЛЬНЫХ ВЫЧИСЛЕНИЙ В ГЕТЕРОГЕННЫХ СРЕДАХ[1]

В.А. Голодов, А.В. Панюков

Для алгоритмического анализа крупномасштабных проблем, чувствительных к ошибкам округления разрабатывается программное обеспечение, реализующее точные дробно-рациональные вычисления для распределенной вычислительной среды с интерфейсом MPI. Дальнейшее повышение эффективности программного обеспечения возможно за счет применения гетерогенных вычислительных систем, позволяющих выполнять локальные арифметические операции с числами сверхбольшой разрядности параллельно в большом числе процессов. В работе представлено исследование масштабируемости алгоритмов основных арифметических операций и методы ее повышения. Показана возможность повышения эффективности программного обеспечения за счет применения массового параллелизма в гетерогенных вычислительных системах. Использование избыточной позиционной системы счисления, предложенной в работе, позволяет выполнять операцию алгебраического сложения за константное время, что позволяет построить хорошо масштабируемые алгоритмы выполнения всех основных арифметических операций с целыми числами. Масштабируемость основных алгоритмов целочисленной арифметики легко переносится на дробно-рациональную арифметику.

Ключевые слова: длинная арифметика, масштабирумые алгоритмы целочисленной арифметики, избыточная система счисления, рациональные вычисления.

Введение

Вычисления повышенной точности, а также безошибочные рациональные вычисления являются необходимым [1–3] и достаточным [4–6] средством алгоритмического анализа крупномасштабных и/или неустойчивых задач. Решение задач дискретной оптимизации методом элипсоидов [7, 8] требует, чтобы число битов в представлении операндов в пять раз превосходило количество итераций.

В настоящее время такие возможности представляет известная библиотека GMP (The GNU Multiple Precision Arithmetic Library) [9]. Библиотека распространяется под лицензией GNU LGPL, актуальная версия библиотеки GMP 6.0.0 доступна для загрузки с официального сайта проекта. Библиотека использует возможности центральных процессоров, в том числе процессоров архитектуры ARM. Программный код оптимизирован с помощью ассемблерных вставок, соответствующих поддерживаемым библиотекой процессорным семействам, но она *не предоставляет* своим объектам возможность их использования в распределенных вычислениях.

В то же время потенциал современных мультипроцессорных архитектур, таких как Fermi, Kepler, Maxwell, Graphics Core Next, Intel Xeon Phi, позволяют не только разгрузить центральный процессор, но и полностью портировать вычислительную задачу на сопроцессор, оставив центральному процессору управляющие функции.

[1]Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии – 2015»

Тем самым встает вопрос поддержки вычислений повышенной точности и безошибочных рациональных вычислений в данных вычислительных устройствах. В рамках современных десктопных ЭВМ и небольших вычислительных станций может объединяться до 10 сопроцессоров в относительно небольшом корпусе, который не требует специальной инфраструктуры и может охлаждаться штатным набором вентиляторов. Это выводит на первый план вопросы координации, интеграции сопроцессоров, обмена данных между устройствами. Производители вычислительных элементов и программного обеспечения уделяют данным вопросам повышенное внимание.

Помимо организации обмена данных между устройствами важнейшей задачей является обеспечение эффективности их использования и конкурентоспособности гетерогенных решений по сравнению с традиционными решениями на базе многопроцессорных комплексов. Эта задача требует существенно переработать алгоритмы базовых арифметических операций, поскольку традиционные последовательные алгоритмы не могут использовать потенциал массового параллелизма устройств, при этом лишены на сопроцессорах возможности использовать высокие тактовые частоты (с среднем тактовые частоты элементов топовых сопроцессоров в 2–3 раза ниже частот топовых центральных процессоров). Требуется глобальный пересмотр алгоритмов базовых арифметических операций для совместимости с архитектурами современных массивно-параллельных сопроцессоров.

Возможности использования современных гетерогенных вычислительных систем уделено внимание при разработке библиотеки «Exact computation 2.0» [10]. Элементами библиотеки являются классы `overlong` и `rational`. Данные классы позволяют производить безошибочные дробно-рациональные вычисления. Дальнейшее повышение эффективности программного обеспечения, использующего «Exact computation 2.0», возможно за счет применения оптимизации локальных операций длинной арифметики с применением распараллеливания на большое количество потоков.

В работе [11] предложены методы построения базовых параллельных вычислительных структур ПЛИС для операций сложения, вычитания, умножения, ориентированных на представлении данных в коде с большой разрядностью. Технология CUDA (Compute Unified Device Architecture) [12] и язык OpenCL дают средства построения современных комплексов программ для гибких гетерогенных систем. В частности, существует опыт использования графических процессоров для реализации операций длинной целочисленной арифметики с применением двоично-десятичного представления [13] и системы остаточных классов [14, 15] в задачах криптографии. В данной статье рассматриваются масштабируемые параллельные алгоритмы выполнения базовых арифметических операций.

Статья организована следующим образом. В разделе 1 рассмотрены параллельные алгоритмы базовых арифметических операций в двоичной позиционной системе счисления. В разделе 2 приведено расширение алгоритмов для незнаковой двоичной позиционной системы счисления, позволяющей более эффективно организовать операции алгебраического сложения и вычитания. В разделе 3 описано построение избыточных систем счисления, в которых макимальная длина цепи переноса в операциях сложения/вычитания равна 1. В разделе 4 описана программная реализация поддержки дробно-рациональной арифметики в гетерогенных вычислительных средах с использованием объектно-ориентированного программирования.

1. Алгоритмы базовых арифметических операций для устройств с массовым параллелизмом

1.1. Сложение неотрицательных целых чисел

Один из возможных способов распараллеливания классического алгоритма сложения n-разрядных чисел в позиционной системе счисления по основанию R столбиком состоит в следующем. На первой стадии выполняется параллельное поразрядное суммирование, в результате которого в некоторых l разрядах возникает переполнение. После этого возможно выполнние l параллельных процессов распространения переноса.

В силу использования двоичного представления данных в ЭВМ выгодно использовать систему счисления с основанием $R = 2^m$, где m — разрядность используемых регистров. Все алгоритмы далее опираются на представление цифры числа как двоичного слова.

Алгоритм Addition (рис. 1) содержит процедуры Digit_Addition, Carry_Propagation, Add_Process, описывающих локальные процессы для каждой цифры, и процедуру Add, в которой определяются параметры и резервируется требуемое количество параллельных локальных процессов.

Для получения суммы $(t_n \dots t_0)_R = (a_{n-1} \dots a_0)_R + (b_{m-1} \dots b_0)_R,\ n \geq m$ требуется вызвать процедуру Add (a, b, n, m, t). Оценим затраты времени, требуемые для выполнения алгоритма Addition и возможный выигрыш от распараллеливания. Для выполнения процедуры Digit_Addition каждому из локальных процессов достаточно время s, равное времени пересылки. Время выполнения процедур Carry_Propagation и Add_Process может изменяться в пределах $[0,\ n \cdot s]$. Поэтому время выполнения алгоритма Addition в зависимости от исходных данных может изменяться в пределах $[s,\ 2 \cdot n \cdot s]$. Время выполнения сложения строго последовательным алгоритмом (т.е. цифра за цифрой) равно $3n \cdot s$.

Найдем оценку среднего времени выполнения алгоритма Addition при условии, что цифры слагаемых являются случайными числами, равномерно распределенными на отрезке $[0,\ R-1]$. С целью упрощения выкладок будем считать $n = m$. Вероятность переполнения в любом разряде $i = 0, 1, \dots, n-1$ равна

$$p = \mathrm{P}\left\{a_i + b_i \geq 2^r\right\} = \sum_{l=1}^{2^r-1} \mathrm{P}\left\{a_i = l\right\} \mathrm{P}\left\{b_i \geq 2^r - l\right\} = \sum_{l=1}^{2^r-1} \frac{1}{2^r} \cdot \frac{l}{2^r} = \frac{1}{2}\left(1 - \frac{1}{2^r}\right). \quad (1)$$

Вероятность получения после сложения цифр величины $2^r - 1$ равна

$$q = \mathrm{P}\left\{a_i + b_i = 2^r - 1\right\} = \sum_{l=0}^{2^r-1} \mathrm{P}\left\{a_i = l\right\} \mathrm{P}\left\{b_i = 2^r - 1 - l\right\} = \sum_{l=0}^{2^r-1} \frac{1}{2^r} \cdot \frac{1}{2^r} = \frac{1}{2^r}. \quad (2)$$

Вероятность отсутствия переноса равна

$$P_0 = \mathrm{P}\left\{(\forall i = 0, 1, \dots, m-1)\,(a_i + b_i \leq 2^r)\right\} = \left(1 - \mathrm{P}\left\{a_i + b_i \geq 2^r\right\}\right)^m = (1-p)^m \to 0. \quad (3)$$

Вероятность отсутствия цепи сквозного переноса (т.е. цепи из более одного последовательного переноса) равна $P_1 = (1-pq)^m \to 1$. Следовательно, в асимптотике среднее время выполнения алгоритма будет равно $3s$. Таким образом, рассмотренный алгоритм в сравнении с последовательным имеет среднее быстродействие в n раз выше, и не зависит от длины складываемых чисел.

Require: $R = 2^r$, $n \geq m$, $a_i = \left(a_i^{r-1} \ldots a_i^0\right)_2$, $i = 0, 1, 2, \ldots, n-1$ и $b_j = \left(b_j^{r-1} \ldots b_j^0\right)_2$, $j = 0, 1, 2, \ldots, m-1$;
Ensure: $t = (t_n, ..., t_0)_R = (a_{n-1} \ldots a_0)_R + (b_{m-1} \ldots b_0)_R$, $t_n \in \{0, R\}$.

1: **procedure** DIGIT_ADDITION(In: a, b, i, Out: c_i, t_i)
2: $\left(s_i^r\, s_i^{r-1} \ldots s_i^1\, s_i^0\right)_2 \leftarrow \left(a_i^{r-1}\, a_i^{r-2} \ldots a_i^1\, a_i^0\right)_2 + \left(b_i^{r-1}\, b_i^{r-2} \ldots b_i^1\, b_i^0\right)_2$;
3: $t_i \leftarrow \left(s_i^{r-1} \ldots s_i^1\, s_i^0\right)_2$ ▷ i-я цифра до распространения переноса
4: $c_i \leftarrow s_i^r$ ▷ значение переноса в $(i+1)$-ю цифру
5: **end procedure**

6: **procedure** CARRY_PROPAGATION(In: n, i, c_i, t, Out: t)
7: **while** $c_i \neq 0$ **do** ▷ если $c_i = 0$, то нет переноса
8: $i \leftarrow i + 1$;
9: $\left(s_i^r\, s_i^{r-1} \ldots s_i^1\, s_i^0\right)_2 \leftarrow t_i + c$
10: $t_i \leftarrow \left(s_i^{r-1} \ldots s_i^1\, s_i^0\right)_2$ ▷ i-я цифра после распространения переноса
11: $c_i \leftarrow s_i^r$ ▷ значение переноса в следующую цифру
12: **end while**
13: **end procedure**

14: **procedure** ADD_PROCESS(In: a, b, i, Out: t)
15: **var** c_i ▷ переносы данного локального процесса
16: DIGIT_ADDITION(a, b, i, c_i, t_i)
17: SYNCHRONIZATION
18: CARRY_PROPAGATION(n, i, c_i, t)
19: **end procedure**

20: **procedure** _GLOBAL_ADD(In: a, b, Out: n, m, t) ▷ выполнение сложения
21: $n \leftarrow \max\{\text{sizeof}\ (a), \text{sizeof}\ (b)\}$
22: $m \leftarrow \min\{\text{sizeof}\ (a), \text{sizeof}\ (b)\}$
23: Fork m parallel processes
24: ADD_PROCESS(a, b, i, t)
25: Join
26: **end procedure**

Рис. 1. Алгоритм Addition

1.2. Распространение переносов

Максимальное количество последовательных операций, выполняемых одним процессом, определяется максимальной длинной возникающих цепей переноса, в самом неблагоприятном случае, возможно образование цепочки переносов сравнимой с числом разрядов одного из слагаемых. Время выполнения сложения можно улучшить за счет совершенствования алгоритма распространения переносов.

Одним из возможных способов такого совершенствования является вычисление цепей распространения переносов параллельно с их распространением. Если перенос попадает на вычисленную цепь, то его распространение по данной цепи осуществляется за один такт.

```
1: procedure CARRY_PROPAGATION(In: n, i, Out: c, t)
2:     L ← 1, V ← i                                   ▷ длина и граница фрагмента
3:     while L ≤ n do                                 ▷ есть фрагменты для объединения
4:         M ← i mod 2L
5:         if (M < L) then                            ▷ i принадлежит младшему фрагменту
6:             if (M = L − 1) then                    ▷ i – старшая цифра младшего фрагмента
7:                 j ← min{i + L, n − 1}              ▷ старшая цифра присоединяемого фрагмента
8:                 NotCarry ← (t_i ≠ 2^r − 1) ∪ (V ≠ i)   ▷ отсутствует сквозной перенос
9:                 send {c, NotCarry, V} to process j
10:                if ((c ≠ 0) ∪ NotCarry) then
11:                    terminate process
12:                end if
13:            end if
14:        else                                       ▷ i принадлежит старшему фрагменту
15:            j ← i + L − M − 1                      ▷ j – старшая цифра младшего фрагмента
16:            flag ← ((M = 2L − 1) ∪ (i = n − 1))
17:            if flag then                           ▷ i – старшая цифра старшего фрагмента
18:                receive {Cj, NotCarry, Vj} from process j
19:                send {Cj, NotCarry} to processes k = j + 1, ..., V
20:                if (NotCarry) then
21:                    V ← Vj
22:                else if (i = V) then
23:                    (s^r s^{r−1} ... s^1 s^0)_2 ← (c t_i^{r−1} ... t_i^1 t_i^0)_2 + Cj
24:                    t_i ← (s^{r−1} ... s^1 s^0)_2, c ← s^r
25:                end if
26:            else                                   ▷ i не является старшей цифрой старшего фрагмента
27:                if (i ≤ V) then                    ▷ i принадлежит цепи сквозного переноса
28:                    receive {Cj, NotCarry} from process j + L
29:                    if (Cj ≠ 0) then
30:                        t_i ← 0
31:                        if (i = V) then
32:                            t_{i+1} ← t_{i+1} + 1
33:                        end if
34:                        terminate process
35:                    else if NotCarry then
36:                        terminate process
37:                    end if
38:                end if
39:            end if
40:        end if
41:        L ← 2L
42:    end while
43:    terminate process
44: end procedure
```

Рис. 2. Процедура Carry_Propagation

Процедура Carry_Propagation, представленная на рис. 2, описывает такое ускоренное распространение сквозных переносов.

Алгоритм состоит в следующем. Первоначально результаты процедуры Digit_Addition (т.е. число t и значения переносов из каждого разряда c) представлены в виде n фрагментов, при этом каждая цифра d_i, $i = 0, 1, \ldots, n-1$ соответствует фрагменту с выделенным процессом i.

На k-й итерации **while** цикла фрагменты, ассоциированные с процессами $l2^k$ и $(l+1)2^k$, объединяются в один фрагмент, ассоциированный с процессом $(l+1)2^k$.

При объединении младший процесс $l2^k$ посылает старшему процессу $(l+1)2^k$ флаг $NotCarry$ отсутствия сквозной цепи переноса через объединенный фрагмент, значение c переноса из младшего фрагмента и возможную границу V распространения сквозного переноса в объединенном фрагменте. Старший процесс $(l+1)2^k$ при ненулевом переносе из младшего фрагмента $l2^k$ пересылает его всем ожидающим его процессам (от $l2^k+1$-й до $V((l+1)2^k)$-й), устанавливает границу V распространения сквозного переноса в объединенном фрагменте и завершает выполнение процессов, которые далее не учавствуют в распространении переносов.

Оценим затраты времени необходимые для выполнения процедуры Carry_Propagation. Данная процедура содержит цикл ''while'', который выполняется любым и созданных процессов не более $\lceil \log_2 n \rceil$ раз. Каждый из активных процессов выполняет операторы, указанные в строках 2 – 5, и 41 данного цикла. При соответствующей оптимизации гетерогенной среды эти операции могут быть выполнены за один такт. Каждый из активных процессов также выполняет также выполняет не более одной посылки и не более одного приема сообщений объемом $3r$ бит. Подготовка и выполнение данных коммуникаций можгут быть осуществлены за два такта. Таким образом, при соответствующей оптимизации гетерогенной среды, время выполнения процедуры Carry_Propagation в среднем не превосходит величины $3s$, а в худшем случае — величины $3s\lceil \log_2 n \rceil$.

Среднее время выполнения операции сложения с применением процедуры процедуры Carry_Propagation оказывается равным $4s$, т.е. превышает среднее время выполнения алгоритма Addition (рис. 1) в 4/3 раз. Эффективность использования алгоритма 2 очевидна при наличии цепей сквозного переноса длины более двух цифр.

1.3. Бинарные отношения

Для проверки истинности бинарных отношений $a \rho b \ : \rho \in \{<, \leq, =, \geq, >, \neq\}$ достаточна проверка истинности отношений $\rho \in \{=, >\}$. Действительно, $(a \geq b) = (a > b) \vee (a = b)$, $(a \leq b) = \neg(a > b)$, $(a \neq b) = \neg(a = b)$, $(a < b) = \neg(a \geq b)$.

Алгоритм вычисления истинности бинарных отношений $(a = b)$ и $(a > b)$ для заданных неотрицательных целых a и b приведен на рис. 3. Сущность алгоритма состоит в следующем. Первоначально данные представляют в виде n фрагментов с выделенными процессами $i = 0, 1, \ldots, n-1$, а локальные переменные p_i и q_i процесса i отображают истинность отношений $(a_i = b_i)$ и $(a_i > b_i)$.

При k-м выполнении цикла **for** осуществляется слияние фрагментов $l2^k$ и $(l+1)2^k$ в один фрагмент, ассоциированный с процессом $l2^{k-1}$. При этом вычисляются значения p_i и q_i объединенного фрагмента $i = l2^{k-1}$, процессы $l2^k$ и $(l+1)2^k$ завершаются.

Легко заметить, что алгоритм на рис. 3 выполняется не менее чем в $n/\log_2 n$ быстрее строго последовательного алгоритма.

Require: $a = (a_{n-1} \dots a_0)_R$, and $b = (b_{n-1} \dots b_0)_R$, $R = 2^r$, $n > 0$, $a_i = \left(a_i^{r-1} \dots a_i^0\right)_2$, $b_i = \left(b_i^{r-1} \dots b_i^0\right)_2$, $i = 0, 1, 2, \dots, n-1$
Ensure: p – значение истинности для $a = b$, q – значение истинности для $a > b$.

```
 1: procedure EqG_Process(In: a, b, n, i, Out: p, q)
 2:     p ← (a = b)
 3:     q = (a > b)
 4:     L ← n/2
 5:     while (L > 1) do
 6:         if (i > 0) then
 7:             send {p, q} to process i/2
 8:         end if
 9:         if i < L then
10:             if there is sending from process 2i then
11:                 receive {p_0, q_0}
12:             else p_0 = true, q_0 = false
13:             end if
14:             if there is sending from process 2i + 1 then
15:                 receive {p_1, q_1}
16:             else p_1 = true, q_1 = false
17:             end if
18:             read timing
19:             p ← p_1 ∧ p_0
20:             q ← q_1 ∨ (p_1 ∧ q_0)
21:         else
22:             terminate process
23:         end if
24:         L ← L/2
25:     end while
26: end procedure

27: procedure _global_EqG(In: a, b, n, Out: p, q)
28:     Fork n parallel processes
29:     ExecInParallel EqG_Process(a_i, b_i, n, i, p_i, q_i)        ▷ i – номер процесса
30:     Join
31:     p ← p_0, q ← q_0
32: end procedure
```

Рис. 3. Алгоритм проверки истинности бинарных отношений $(a = b)$ и $(a > b)$.

1.4. Определение количества значащих цифр

Для рационального распределения вычислительных ресурсов для результата выполнения любой арифметической операции необходимо знать количество значащих цифр ее операндов.

В таких операциях как сложение, умножение и деление количество значащих цифр результата операции определяется по количеству значащих цифр операндов с погрешностью в одну цифру. Количество значащих цифр результата операции вычитания можно определить только после ее выполнения.

Следовательно, количество значащих цифр разумнее использовать в качестве одного из атрибутов объекта, представляющего число, а количество значащих цифр вычислять только после выполнения операции вычитания.

1.5. Умножение многозначного числа на однозначное

Алгоритм, приведенный на рис. 4, вычисляет произведение $(c_n, ..., c_0)_R$ заданных неотрицательных целых чисел $a = (a_{n-1}, ..., a_0)_R$ и $b = (b_0)_R$. Из описания алгоритма на рис. 4

Require: $a = (a_{n-1} \ \dots \ a_0)_R$, $n > 0$, $b = (b_0)_R$, $R = 2^r$.
Ensure: $(t_n \ t_{n-1} \ ... \ t_0)_R$ – произведение $a \cdot b$.

```
 1: procedure M_Process(In: ad, b, i, n, dt)
 2:     if (i < n) then
 3:         (x_1 x_0)_R ← ad · b
 4:         send x_1 to process (i + 1)
 5:     else
 6:         x_0 ← 0
 7:     end if
 8:     if (i > 0) then
 9:         receive x_1 from process (i − 1)
10:         (s^r s^{r−1} ... s^1 s^0)_2 ← x_0 + x_1
11:         c ← s^r,   dt ← (s^{r−1} ... s^1 s^0)
12:         send c to process (i + 1)
13:         receive c from process (i − 1)
14:         dt ← dt + c
15:     else
16:         dt ← x_0
17:     end if
18: end procedure

19: procedure _global_ M(a, b, n, t)
20:     Fork n parallel processes
21:     ExecInParallel M_Process(a_i, b, i, n, t_i)        ▷ i – номер процесса
22:     Join
23: end procedure
```

Рис. 4. Вычисление произведения числа a и цифры b

видно, что время его выполнения не превосходит величины $4s$. Таким образом, затраты времени на выполнение алгоритма на рис. 4 в n раз меньше времени работы последовательного алгоритма и не зависят от длины операндов.

1.6. Умножение многозначных чисел

Алгоритм, приведенный на рис. 5, вычисляет произведение двух неотрицательных чисел.

Require: $a = (a_{n-1} \dots a_0)_R$, $b = (b_{m-1} \dots b_0)_R$, $n \geq m > 0$, $R = 2^r$
Ensure: $(c_{n+m-1}, ..., c_0)_R$ – произведение of $a \cdot b$

```
 1: procedure MM_Process(a, b, i, n, m, c)
 2:     _global_M(a, b_i, n, z)
 3:     L ← m/2, B ← R
 4:     while (L > 1) do
 5:         if (i > 0) then
 6:             send z to process i/2
 7:         end if
 8:         send timing
 9:         if i < L then
10:             if there is sending from process 2i then
11:                 receive value for s_0
12:             else s_0 ← 0
13:             end if
14:             if there is sending from process 2i + 1 then
15:                 receive value for s_1
16:             else s_1 ← 0
17:             end if
18:             s_1 ← s_1 · B
19:             _global_Add(s_0, s_1, n, m, z)
20:         else
21:             terminate process
22:         end if
23:         L ← L/2, B ← B^2
24:     end while
25:     c ← z
26: end procedure

27: procedure _global_MM(a, b, n, m, c)
28:     Fork m parallel processes
29:     ExecInParallel MM_Process(a, b, i, n, m, z)     ▷ i – количество процессов
30:     Join
31:     c ← z
32: end procedure
```

Рис. 5. Вычисление произведения $c = a \cdot b$

Процедура _global_MM (рис. 5) порождает m процессов, каждый из которых вызывает процедуру M (строка 2), создающую n процессов. Следовательно, общее количество процессов равно mn.

Время выполнения процедуры MM_Process равно $4s$. Тело цикла **while** (строки 4–23) выполняется не более $\lceil \log_2 m \rceil$ раз. Оно содержит не более одной отправляющей (строка 6) и двух принимающих коммуникации (строки 11 и 15) коммуникаций «точка – точка», и одно сложение $(n+m-2L)$-разрядных чисел. Выполнение остальных операторов можно организовать за один такт. Таким образом, в среднем время, необходимое для вычисления произведения, не превосходит величины $(4+3\log_2 m)\,s$. В наихудшем случае время вычисления произведения не превосходит величины $(4+3(n+m)\log_2 m)\,s$.

В наихудшем случае с увеличением разрядности слагаемых время выполнения растет немного быстрее линейной функции, тогда как последовательный алгоритм умножения столбиком имеет квадратичную зависимость времени выполнения от разрядности сомножителей.

1.7. Деление

Классический алгоритм деления «столбиком» не является масштабируемым. В соответствии с данным алгоритмом при его реализации необходимо последовательно выполнить $(n+m-1)$ операций умножения-вычитания m-разрядных чисел. В работах [16, 17] предложено решение проблемы повышения эффективности алгоритма операции деления посредством применения метода Ньютона.

Чтобы разделить целое число $u = (u[n-1]\,u[n]\,\ldots\,u[1]\,u[0])_R$ на другое целое число $v = (v[m-1]\,\ldots\,v[0])_R$, сначала предлагается найти достаточно точное приближение к числу $1/v$, затем умножить его на u, что даст приближение к u/v. Очевидно, что длина целочисленного ответа будет не более $n-m+1$. Число $1/v$ содержит не более m незначащих нулей в старших разрядах, для получения правильного результата деления достаточно, чтобы приближенное значение $1/v$ содержало еще хотя бы $n-m+1$ значащих цифр. Таким образом, достаточная точность вычисления величины $1/v$ определяется величиной R^{-n+1}.

Применение метода Ньютона к задаче нахождения корня уравнения $f(x)=0$, где $f(x)=v-1/x$, состоит в последовательном вычислении $x_{k+1}=(2-v\cdot x_k)\cdot x_k,\ k=0,1,2,\ldots$, где x_0 – начальное приближение, вычисленное с достаточной точностью. При $x \geq 1$ функция $f(x)$ является дважды непрерывно дифференцируемой и строго выпуклой. В этом случае метод Ньютона обладает квадратичной скоростью сходимости, т.е. количество правильно вычисленных разрядов после выполнения очередной итерации будет удваиваться. Начальное приближение $x_0 = 1/(v[m-1]$ величины $1/v$ имеет погрешность

$$\frac{1}{v[m-1]\cdot R^{m-1}} - \frac{1}{v} = \frac{v - v[m-1]\cdot R^{m-1}}{v\cdot v[m-1]\cdot R^{m-1}} \leq \frac{1}{v\cdot v[m-1]} \leq R^{-m+1}, \tag{4}$$

т.е. в нем правильно вычислено m разрядов. Таким образом, количество итераций, которые потребуется выполнить по методу Ньютона, будет не более $4\log_2(n+1) - \log_2 m$.

Алгоритм, приведенный на рис 6, вычисляет частное двух неотрицательных целых.

На итерации $k = 0,1,2,\ldots,l < \log_2(n+1) - \log_2 m$ цикла **for** переменная x представляет целое $(2^{k+1}-1)$-разрядное число. Поэтому в теле цикла с помощью параллельных алгоритмов выполняется одна операция умножения переменной x на m-разрядное число b, одна операция вычитания (2^k)-разрядных чисел и одна операция перемножения (2^k)-разрядных чисел. Следовательно, время, необходимое для выполнения шага тела цикла, не превосходит величины $11 + 3(\log_2 m + k)] \cdot s$ в среднем случае и величины $\left[3\cdot 2^k k + 2.5\cdot 2^k + 6k + 3m\log_2 m + 10\right] s$ в наихудшем случае.

Require: $a = (a_{n-1} \dots a_0)_R$, $b = (b_{m-1} \dots b_0)_R$, $n \geq m > 0$, $R = 2^r$
Ensure: $(c_{n+m-1}, \dots, c_0)_R$ – частное $c = a/b$.

1: **procedure** _GLOBAL_D(a, b, n, m, c)
2: $x \leftarrow \left\lfloor \frac{R-1}{b[m-1]} \right\rfloor_R$, $\tilde{R} \leftarrow R^m$ ▷ Начальное приближение
3: **for** $i = 0, 1, \dots, \lceil \log_2 \frac{n+1}{m} \rceil$ **do** ▷ Уточнение
4: $d \leftarrow x$, $x \leftarrow \left(2 \cdot \tilde{R} - b \cdot d\right) \cdot d$, $\tilde{R} \leftarrow \tilde{R} \cdot \tilde{R}$
5: **end for**
6: $z = a \cdot x$ ▷ Умножение
7: $c = z/\tilde{R}$ ▷ Формирование ответа
8: **end procedure**

Рис. 6. Вычисление частного $c = a/b$, a и b — неотрицательные целые

Поскольку

$$\sum_{k=0}^{l} k = \frac{l(l+1)}{2}, \quad \sum_{k=0}^{l} 2^k = 2^{l+1} - 1, \tag{5}$$

$$\sum_{k=0}^{l} \left(k \cdot 2^k\right) = 2^{l+1}(l-1) + 2 \tag{6}$$

то время выполнения тела цикла **for** в среднем и наихудшем случаях не превысит соответственно величин $O\left(\log_2 n \cdot \log_2 \frac{n}{m}\right)$ и $O\left(\frac{n}{m} \log_2 \frac{n}{m}\right)$.

Перемножение n-значных чисел завершает выполнение алгоритма D. Время выполнения данного шага не больше величины $O(\log_2 n)$ в среднем и величины $O(n \cdot \log_2 n)$ в наихудшем случаях. Таким образом, окончательные оценки времени выполнения алгоритма 6 в среднем и наихудшем случаях равны соответственно

$$O\left(\log_2 n \cdot \log_2 \frac{n}{m}\right) \text{ и } O\left(\frac{n}{m} \log_2{}^{3/2} \frac{n}{m} + n \log_2 n\right). \tag{7}$$

2. Применение знаковых позиционных систем счисления

Приведенная выше система счисления является незнаковой, цифрами в позиционной системе по основанию R являются числа 0, 1, 2, …, $R-2$, $R-1$. Ее недостатком является сложность реализации операций алгебраического сложения и вычитания, т.к. она требует выполнения операции сравнения чисел. Устранить отмеченный недостаток позволяет применение знаковых позиционных систем счисления. Цифрами в знаковой позиционной системе по основанию R являются числа

$$-\left\lfloor \frac{R}{2} \right\rfloor, -\left\lfloor \frac{R}{2} \right\rfloor + 1, \dots, -1, 0, 1, \dots, \left\lceil \frac{R}{2} \right\rceil - 2, \left\lceil \frac{R}{2} \right\rceil - 1. \tag{8}$$

Отметим, что при нечетном R количество положительных и отрицательных цифр одинаково, а при четном R количество положительных на одно меньше количества отрицательных.

Представление числа в знаковой позиционной системе счисления по основанию $R = 2^r$ имеет вид $(a_{n-1}, ..., a_0)_{\pm R}$, а его цифры как $a_i = (a_i^{r-1}\, a_i^{r-2} \dots a_i^1\, a_i^0)_{\pm 2}$, $i = 0, 1, \dots, n-1$. Старший бит в представлении цифры определяет знак цифры (0 для положительных, 1

для отрицательных). В этом случае цифрами системы счисления являются C++-объекты типа `integer`.

Заметим, что все основные алгоритмы для незнаковых систем счисления, за исключением алгоритмов сложения/вычитания, переносятся на знаковые системы без изменения. Алгоритмы сложения/вычитания объединяются в общий алгоритм алгебраического сложения.

Применение знаковых позиционных систем упрощает алгоритм алгебраического сложения, но не решает проблему повышения эффективности вычислений при наличии сквозных переносов. Преимущества и недостатки ускоренного распространения переносов такие же как в случае безнаковых систем.

Истинность бинарных отношений в знаковых системах реализуется посредством операции вычитания. Знак числа определяется знаком старшего разряда. Алгоритмы определения количества значащих цифр, умножения и деления такие же как в безнаковых системах.

3. Применение избыточных позиционных систем счисления

Изложенный выше анализ показывает на высокую в среднем эффективность применения распараллеливания в алгоритмах всех арифметических операций. При этом среднее время вычисления результатов сложения, вычитания, умножения на одноразрядное число и бинарных отношений имеет величину $O(1)$, среднее время вычисления умножения и деления чисел разрядности n не превосходит величины $O(\log_2^2 n)$.

Однако в наихудшем случае время вычисления результатов любой операции с числами разрядности n оказывается не меньше $O(n)$ при обычном распространении переносов и не меньше величины $O(\log_2 n)$ при ускоренном распространении переносов. Причиной отклонений от среднего значения является то, что при выполнении сложения (вычитания) для распространения переносов появляются цепи длины более единицы.

Исключить отмеченные прецеденты можно за счет использования избыточных позиционных систем счисления, в которых разрешаются цепи переносов длины не более единицы. В качестве недостатка отметим множественность представления чисел в избыточной системе счисления, что позволяет эффективно вычислять бинарные отношения и количество значащих цифр только после распространения переносов, устраняющих избыточность представления. Напомним, что в асимптотике вероятность появления дополнительных переносов стремится к нулю.

4. Реализация поддержки дробно-рациональной арифметики для гетерогенной вычислительной среды на языке C++

Объектами класса `rational` являются обыкновенные дроби p/q, где p, q — объекты класса `overlong`. Класс `overlong` предназначен для расширения логических возможностей целочисленных вычислений на компьютере. Объем памяти, занимаемый такими объектами, определяется значениями представляемых чисел, их диапазон ограничен только объемом адресуемой памяти. Диапазон чисел, представляемых объектами класса `overlong`, расширен до $(-2^{(r\cdot 2^{r-1})}, 2^{(r\cdot 2^{r-1})})$, где r — разрядность используемых цифр. Минимальный шаг дискретизации чисел, представляемых объектами класса `rational`, может достигать $2^{-r2^{r-1}}$. Для объектов классов `overlong` и `rational` определены все операторы, операции и бинарные отношения, используемые для стандартных числовых типов данных, а так-

же интерфейс MPI. Применение библиотеки «Exact computation 2.0» для решения плохо обусловленных систем линейных алгебраических уравнений [5, 18] и задач линейного программирования [4] показало ее высокую эффективность.

Для решения задач, требующих применения точной дробно-рациональной арифметики, наиболее эффективным является применение позиционной системы счисления с основанием $R = 2^r$. В работе изложены основные алгоритмы целочисленной арифметики в позиционной системе счисления по основанию R для гетерогенных вычислительных систем с анализом их масштабируемости, анонсированные на конференциях [5, 17, 19, 20]. Для более полноценного использования современных вычислительных архитектур классы `overlong` и `rational` оперируют числами по основанию 2^{32}, это позволяет оптимизировать проведение операций над числами с помощью быстрых бинарных логических операций.

Оптимизации применяются также при работе с памятью. Поскольку в C++ нет автоматического сборщика мусора, то избыточные перевыделения памяти приводят ее фрагментации и снижению быстродействия приложения в целом. Краткое описание современных реализаций классов для процессоров семейств x86 и x86-64 дано в [21], описание особенностей реализации классов в гетерогенной среде приводится также в [22].

Операции с памятью инкапсулированы в отдельный класс `MemHandle`, а выполнение базовых арифметических операций с разрядами полностью производится в рамках класса `ArifRealization` (рис. 7). Реализации классов отделены от интерфейсов, для каждой из используемых архитектур написаны соответствующим образом оптимизированные реализации, учитывающие большое количество особенностей применяемой архитектуры.

```
class overlongNM {
 private:
 static ArifRealization realization;
 MemHandle mhandle;
 ...
 public:
 inline int32 size() const;//length
 inline int32 sign() const;//sign
 ...
 //addition
 template<typename Type>
 friend const overlongNM operator+
   (const overlongNM &num,Type v)
 {overlongNM rez(num); return (rez+=v);}
 friend const overlongNM operator+
   (const overlongNM&,const overlongNM&);
 ...
}
```

Фрагмент класса `overlong`

Объект класса `overlong` содержит в себе объект типа `MemHandle`, и все действия с памятью происходят через интерфейс класса `MemHandle`. Все арифметические операции с данными осуществляются вызовом соответствующих методов класса `ArifRealization`. Таким образом, данные всегда хранятся в памяти того же вычислительного устройства, на котором обрабатываются. Для CPU используется оперативная память, для GPU — глобальная память графического ускорителя.

Эти особенности реализации позволяют в момент запуска приложения выбрать оптимальное вычислительное устройство и, тем самым, место хранения разрядов чисел. В рамках MPI приложения все процессы выбирают вычислительное устройство независимо друг от друга, то есть приложение запущенное на разнородном кластере будет использовать наиболее предпочтительное устройство для каждого потока. Особенностью GPU реализаций является требование к масштабируемости алгоритмов, применяемых для работы с разрядами чисел. В силу меньшей тактовой частоты ядер чипа GPU и их большого числа, применение параллельных масштабируемых алгоритмов является не только оптимизационной особенностью, но и необходимостью.

Заключение

Повышение эффективности программного обеспечения операций дробно-рациональной арифметики возможно за счет применения массового параллелизма в гетерогенных вычислительных средах. Эффективное использование массового параллелизма GPU требует переработки алгоритмов основных арифметических операций в их масштабируемые варианты. Реализация аппарата поддержки арифметики высокой точности требует, помимо проработки пользовательского интерфейса, существенно учитывать возможность применения различных устройств. Дальнейшее повышение эффективности возможно, например, за счет использования предложенной в работе избыточной позиционной системы счисления, которая позволяет строить хорошо масштабируемые алгоритмы выполнения основных арифметических операций.

Литература

1. Alt, R. On the accuracy of the solution of linear problems on the CELL processor / R. Alt, J.-L. Lamotte, S. Markov // Reliable Computing. — 2011. — Vol. 15. — P. 1–12.
2. Beaumont, O. Linear interval tolerance problem and linear programming techniques / O. Beaumont, B. Philippe // Reliable Computing. — 2001. — Vol. 6, No. 4. — P. 365–390.
3. Coxson, G. E. Computing exact bounds on elements of an inverse interval matrix is NP-hard / G. E. Coxson // Reliable Computing. — 1999. — Vol. 5, No. 2. — P. 137–142. DOI: 10.1023/a:1009901405160
4. Panyukov, A. V. Using massively parallel computations for absolutely precise solution of the linear programming problems / A. V. Panyukov, V. V. Gorbik // Automation and Remote Control. — 2012. — Vol. 73, No. 2. — P. 276–290. DOI: 10.1134/s0005117912020063
5. Panyukov, A. V. Exact and guaranteed accuracy solutions of linear programming problems by distributed computer systems with MPI / A. V. Panyukov, V. V. Gorbik // Tambov University Reports. Series: Natural and Technical Sciences. — 2010. — Vol. 15, No. 4. — P. 1392–1404.
6. Panyukov, A. V. Computing the best possible pseudo-solutions to interval linear systems of equations // 15th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Verified Numeric (SCAN'2012, Novosibirsk, Russia, September 23-29, 2012): Book of abstracts. — Institute of Computational Technologies Publisher, 2012. — P. 134–135.
7. Panyukov, A. Polynomial solvability of n np-complete problems / A. Panyukov // ScienceOpen Research. — 2015. https://www.scienceopen.com/document/vid/c0597241-52c2-47a7-8af0-dd09bbb57888 (дата обращения 13.03.2015).

8. Grötschel, M. The ellipsoid method and its consequences in combinatorial optimization / M. Grötschel, L. Lovász, A. Schrijver // Combinatorica. — 1981. — Vol. 1, No. 2. — P. 169–197. DOI: 10.1007/bf02579273
9. The gnu mp bignum library. — February 2013. `http://gmplib.org/` (дата обращения 13.03.2015).
10. Голодов, В. Библиотека классов «Exact Computation 2.0». Свидетельство о государственной регистрации программы для ЭВМ № 2013612818 от 14.03.2013 / В. Голодов, А. В. Панюков // Программы для ЭВМ, базы данных, топологии интегральных микросхем. Официальный бюллетень Российского агентства по патентам и товарным знакам. — М.: ФИПС, 2013. — № 3. — С. 251.
11. Золотовский, В. Реализация «длинных вычислений» на параллельных структурах / В. Золотовский, М. Ф. Гильванов // Известия высших учебных заведений. Северо-Кавказский регион. Серия: Технические науки. — 2008. — № 4. — С. 9–13.
12. Parallel programming and computing platform | cuda | nvidia. — February 2013. `htpp://www.nvidia.com/object/cuda_home.html` (дата обращения 13.03.2015).
13. Желтов, С. А. Реализация арифметических операций с «длинными» числами на устройствах gpgpu / С. А. Желтов // Вопросы защиты информации. — 2012. — № 3. — С. 2–4.
14. Орлов, Д. Реализация арифметики повышенной разрядности на графических процессорах / Д. Орлов // Программная инженерия. — 2012. — № 4. — С. 33–43.
15. Дзегеленок, И. И. Алгебраизация числовых представлений для обеспечения высокоточных суперкомпьютерных вычислений / И. И. Дзегеленок, Ш. А. Оцопков // Вестник Московского энергетического института. — 2010. — № 3. — С. 107–116.
16. Knuth, D. E. The Art of Computer Programming / D. E. Knuth. — 2-nd edition. — Addison-Wesley Longman, 1981. — Vol. 2. — 688 p.
17. Panyukov, A. V. Application of redundant positional notations for increasing of arithmetic algorithms scalability / A. V. Panyukov // 15th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Verified Numeric (SCAN'2012, Novosibirsk, Russia, September 23–29, 2012): Book of abstracts. — Institute of Computational Technologies Publisher, 2012.
18. Панюков, А. В. Сложность нахождения гарантированной оценки решения приближенно заданной системы линейных алгебраических уравнений / А. В. Панюков, М. И. Германенко // Известия Челябинского научного центра УрО РАН. — 2000. — № 4. — С. 21–30.
19. Панюков, А. В. Реализация базовых операций целочисленной арифметики в гетерогенных системах. / А. В. Панюков, С. Ю. Лесовой // Параллельные вычислительные технологии (ПаВТ'2012). – Труды международной научной конференции (Новосибирск, 26 – 30 марта 2012 г.). — Челябинск: Издательский центр ЮУрГУ, 2012. — С. 77–84.
20. Панюков, А. В. Применение массивно-параллельных вычислений для реализации основных операций целочисленной арифметики / А. В. Панюков, С. Ю. Лесовой // Высокопроизводительные параллельные вычисления на кластерных системах (HPC - 2010). Материалы X Международной конференции (г. Пермь, 1 – 3 ноября 2010 г.). В 2-х томах. — Т. 2. — Пермь: Изд-во ПермГТУ, 2010. — С. 77–84.
21. Голодов, В. А. Распределенные символические дробно-рациональные вычисления на процессорах серий x86 и x64 / В. А. Голодов // Труды Международной конференции «Параллельные вычислительные технологии - 2012» (Новосибирск, 2012, 26 – 30 марта). — Челябинск: Издательский центр ЮУрГУ, 2012. — С. 774.

22. Панюков, А. В. Техника программной реализации алгоритма решения системы линейных алгебраических уравнений с интервальной неопределенностью в исходных данных А. В. Панюков, В. А. Голодов // «Параллельные вычисления и задачи управления» PACO'2012. Шестая международная конференция, Москва, 2012 г., 24 – 26 октября. Труды в 3 т. — Т. 2. — М.: ИПУ РАН, 2012. — С. 155–166.

Голодов Валентин Александрович, доцент кафедры экономико-математических методов и статистики, Южно-Уральский государственный университет (Челябинск, Российская Федерация), golodovva@susu.ac.ru.

Анатолий Васильевич Панюков, доктор физико-математических наук, профессор кафедры экономико-математических методов и статистики, Южно-Уральский государственный университет (Челябинск, Российская Федерация), paniukovav@susu.ac.ru.

Поступила в редакцию 9 февраля 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 2, pp. 71–88

DOI: 10.14529/cmse150206

SCALABLE ALGORITHMS FOR THE INTEGER ARITHMETICS AND RATIONAL CALULATIONS IN HETEROGENEOUS COMPUTATION ENVIRONMENT

V.A. Golodov, South Ural State University (Chelyabinsk, Russian Federation)
golodovva@susu.ac.ru,
A.V. Panyukov, South Ural State University (Chelyabinsk, Russian Federation)
paniukovav@susu.ac.ru

Algorithmic analysis of large-scale problems that are sensitive to rounding errors requires precise rational calculations in the distributed computing environment. Enhanced efficiency of the software my be gained to heterogeneous computing systems that perform local basic arithmetic operations simultaneously using large number of ultralight threads. This paper examines the scalability algorithms for basic arithmetic operations and methods of its improvement. The possibility of increasing of the software efficiency using massive parallelism in heterogeneous computing systems is described. The use of reduntant number system allows you to perform the operation of algebraic addition in constant time and to construct scalable algorithms for all basic arithmetic operations. Scalability of the basic integer arithmetic algorithms can be easily transferred to a rational arithmetic.

Keywords: integer arithmetics, rational arithmetics, sclable alrorithms, position number system, reduntant number system.

References

1. Alt R., Lamotte J.-L., Markov S On the accuracy of the solution of linear problems on the CELL processor // Reliable Computing. 2011. Vol. 15. P. 1–12.

2. Beaumont O., Philippe B. Linear interval tolerance problem and linear programming techniques // Reliable Computing. 2001. Vol. 6, No. 4. P. 365–390.
3. Coxson, G. E. Computing exact bounds on elements of an inverse interval matrix is NP-hard // Reliable Computing. 1999. Vol. 5, No. 2. P. 137–142. DOI: 10.1023/a:1009901405160
4. Panyukov A. V., Gorbik V.V. Using massively parallel computations for absolutely precise solution of the linear programming problems // Automation and Remote Control. — 2012. Vol. 73, No. 2. P. 276–290. DOI: 10.1134/s0005117912020063
5. Panyukov A. V., Gorbik V.V. Exact and guaranteed accuracy solutions of linear programming problems by distributed computer systems with MPI // Tambov University Reports. Series: Natural and Technical Sciences. 2010. Vol. 15, No. 4. P. 1392–1404.
6. Panyukov A. V. Computing the best possible pseudo-solutions to interval linear systems of equations // 15th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Verified Numeric (SCAN'2012, Novosibirsk, Russia, September 23–29, 2012): Book of abstracts. Institute of Computational Technologies Publisher, 2012. P. 134–135.
7. Panyukov A. Polynomial solvability of n np-complete problems // ScienceOpen Research. 2015. `https://www.scienceopen.com/document/vid/c0597241-52c2-47a7-8af0-dd09bbb57888` (accessed 13.03.2015).
8. Grötschel M., Lovász L., Schrijver A. The ellipsoid method and its consequences in combinatorial optimization // Combinatorica. 1981. Vol. 1, No. 2. P. 169–197. DOI: 10.1007/bf02579273
9. The gnu mp bignum library. February 2013. `http://gmplib.org/` (accessed 13.03.2015).
10. Golodov V.A., Panyukov A.V. Biblioteka klassov «Exact Computation 2.0». Svidetel'stvo o gosudarstvennoj registracii programmy dlja JeVM № 2013612818 ot 14 marta [Class Library «Exact Computation 2.0», State Registration Certificate № 2013612818 for Computer Code] // Programmy dlja JeVM, bazy dannyh, topologii integral'nyh mikroshem. Oficial'nyj bjulleten' Rossijskogo agentstva po patentam i tovarnym znakam. [Computer Codes, Data Bases, VLSI topologies. Official Bulletin of Russian Agency on Patents and trademarks] M.: FIPS, 2013. № 3. P. 251.
11. Zolotovskij V., Ghilvanov M. Realizacija «dlinnyh vychislenij» na parallel'nyh strukturah [Implementation of Long Calculations for Parallel Structures] // Izvestija vysshih uchebnyh zavedenij. Severo-Kavkazskij region. Serija: Tehnicheskie nauki [Proceedings of North Caucasian Scientific Center. Technical Sciences]. 2008. № 4. P. 9–13.
12. Parallel programming and computing platform | cuda | nvidia. Febbuary 2013. `htpp://www.nvidia.com/object/cuda_home.html` (accessed 13.03.2015).
13. Zheltov, S. A. Realizacija arifmeticheskih operacij s "dlinnymi"chislami na ustrojstvah gpgpu [Implementation of Long Arithmetics Operations for GPGPU devices] // Voprosy zashhity informacii [Information Security Problems]. 2012. № 3. P. 2–4.
14. Orlov D. Realizacija arifmetiki povyshennoj razrjadnosti na graficheskih processorah [Implementation of Long Arithmetics with GPU] // Programmnaja inzhenerija [Software Engendering]. 2012. № 4. P. 33–43.
15. Dzegelenok I.I., Ocopkov Sh.A. Algebraizacija chislovyh predstavlenij dlja obespechenija vysokotochnyh superkomp'juternyh vychislenij [Algebraization of Number Representations for Implementation of High-precision Supercomputer Calculations] // Vestnik Moskovskogo jenergeticheskogo instituta [Herald of Moscow Exegetical Institute]. 2010. № 3. P. 107–116.
16. Knuth, D. E. The Art of Computer Programming / D. E. Knuth. 2-nd edition. Addison-Wesley Longman, 1981. Vol. 2. 688 p.
17. Panyukov A. V. Application of redundant positional notations for increasing of arithmetic al-

gorithms scalability // 15th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic and Verified Numeric (SCAN'2012, Novosibirsk, Russia, September 23-29, 2012): Book of abstracts. Institute of Computational Technologies Publisher, 2012.

18. Panyukov A. V., Germanenko M.I. Slozhnost' nahozhdenija garantirovannoj ocenki reshenija priblizhenno zadannoj sistemy linejnyh algebraicheskih uravnenij [Complexity of Assured Estimating of Solution for Approximately Linear Algebraic Equations System] // Izvestija Cheljabinskogo nauchnogo centra UrO RAN [Proceedings of Chelybinsk Scientific Center]. 2000. № 4. P. 21–30.
19. Panyukov A. V., Lesovoy S. Realizacija bazovyh operacij celochislennoj arifmetiki v geterogennyh sistemah. // Parallel'nye vychislitel'nye tehnologii (PaVT'2012). Trudy mezhdunarodnoj nauchnoj konferencii (Novosibirsk, 26 – 30 marta 2012 g.) [Realization of Base Arithmetical Operations for Heterogeneous Systems // Parallel Computational Technologies (PCT'2012). Proceedings of the International Conference, Novosibirsk, March, 26 – 30, 2012.] Chelyabinsk: Publishing Center of SUSU, 2012. P. 77–84.
20. Panukov A. V., Lesovoy S. Primenenie massivno-parallel'nyh vychislenij dlja realizacii osnovnyh operacij celochislennoj arifmetiki [Application Massive Parallel Calculations for Realization of Base Integer Arithmetic Operations] // Vysokoproizvoditel'nye parallel'nye vychislenija na klasternyh sistemah (HPC - 2010). Materialy X Mezhdunarodnoj konferencii (g. Perm', 1 – 3 nojabrja 2010 g.). [High Performance Calculations with Cluster Systems (HPC-2010). X International Conference (Perm, 2010, November, 1–3)]. Vol. 2. Perm: PermGTU, 2010. P. 77–84.
21. Golodov V. A. Raspredelennye simvolicheskie drobno-racional'nye vychislenija na processorah serij x86 i x64 [Distributed symbolic rational-fractional calculations on the processors of series of x86 and x64] // Trudy Mezhdunarodnoj konferencii «Parallel'nye vychislitel'nye tehnologii - 2012». (Novosibirsk, 2012, on March 26 to 30) [Parallel Computational Technologies (PCT'2012). Proceedings of the International Conference, Novosibirsk, March, 26 – 30, 2012.]. Chelyabinsk: Publishing center of SUSU, 2012. P. 774.
22. Panyukov A. V., Golodov V. A. Tehnika programmnoj realizacii algoritma reshenija sistemy linejnyh algebraicheskih uravnenij s interval'noj neopredelennost'ju v ishodnyh dannyh [Software Engineering for Algorithm to Solve the Linear Algebraic Equation Set with Input Interval Uncertainties] // «Parallel'nye vychislenija i zadachi upravlenija» PACO'2012. Shestaja mezhdunarodnaja konferencja, Moskva, 24–26 oktjabrja. Trudy. [Parallel Computation and Management Problems (PACO'2012). VI International Conference (Moscow, 2012, October, 24–26). Conference Proceedings]. Vol. 2. Moscow: IPU RAN, 2012. P. 155–166.

Received February 9, 2015.

УДК 004.94, 532.685 **DOI: 10.14529/cmse150207**

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ ПРОЦЕССА ФИЛЬТРАЦИИ ФЛЮИДОВ В ПОРИСТЫХ СРЕДАХ

Н. Равшанов, Н.М. Курбонов

Для ускорения разработки нефтегазовых месторождений, повышения их технико-экономических показателей с целью максимального извлечения продуктов из старых нефтегазовых залежей, необходимо проведение комплексных исследований с помощью математического инструментария. Примером такого математического аппарата является триада «математическая модель, численный алгоритм и программно-инструментальный комплекс» для реализации задачи и проведения вычислительных экспериментов на ЭВМ при различных входных технологических параметрах объекта исследования. В настоящей работе рассматриваются математическая модель, консервативный численный алгоритм и программное средство для проведения вычислительных экспериментов, разработанные на основе основных законов гидродинамики. Созданные математическое и программное обеспечения служат цели исследования, прогнозирования и принятия решений по разработке и проектированию нефтяных и газовых месторождений. Также в работе приведены результаты вычислительных экспериментов в виде графиков при неустановившейся фильтрации флюидов в пористых средах.

Ключевые слова: компьютерное моделирование, численный метод, вычислительный эксперимент, пористая среда, флюид, программное средство.

Введение

На настоящем уровне научно-технического прогресса ведущую роль в энергетическом и сырьевом балансе мирового сообщества играют нефтеуглеводороды. Для удовлетворения нужд реального сектора экономики этим ресурсом необходимо ускорение процессов проектирования, разработки и ввода в действие новых нефтяных и газовых месторождений, а также наиболее полное извлечение продуктов из старых залежей.

Достижение указанной цели невозможно без разработки соответствующих математических моделей, эффективных консервативных конечно-разностных методов и программных средств для комплексного исследования процессов, происходящих при разнообразных естественных и искусственных условиях воздействия на продуктивные пласты.

В нефтепромысловой практике для поддержания пластового давления широко используется закачка сухого, попутного газа в нефтяной пласт. С другой стороны, границы нефтяных залежей связаны с естественным напором водных пластов, что в некотором смысле повышает энергию разрабатываемого пласта. Тем самым на нефтяную залежь воздействуют силы с двух сторон: с одной стороны — сила закачиваемого газа, с другой стороны — сила напора грунтовых вод.

Поддерживая давление в пласте, эти силы увеличивают время бескомпрессорной эксплуатации нефтеуглеводородных пластов. Однако они могут отрицательно влиять на обводненные эксплуатационные скважины. В связи с этим возникает задача о необходимости определения значений дебитов эксплуатационных и нагнетательных скважин и их регулирования. А это уже непосредственно связанно с разработкой математических моделей, адекватно описывающих указанные процессы и с созданием соответствующих алгоритмов, а также программно-инструментальных средств для проведения вычислительных экспериментов, в том числе на высокопроизводительных платформах. Матема-

тические модели этих сложных процессов, происходящих в пластовых условиях, основаны на методах механики многофазных сред. Они, как правило, формулируются в виде задач типа Стефана.

Конструктивная системная методология математического моделирования и вычислительный эксперимент необходимы для подробного и глубокого изучения процесса фильтрации жидкости и газа в многослойных системах. Так как в них изменяется состояния объекта по пространственной и временной переменным, а так же имеет место переход энергии из одного вида в другой.

В настоящей статье проведен краткий анализ ряда работ, посвященных моделированию процесса массопереноса в пористых средах, и дана их критическая оценка. Приведена постановка задачи трехфазной фильтрации при воздействии на продуктивный пласт объемом газа и продвижения жидкости. Рассматриваются полученные математическая модель и консервативный численный алгоритм решения задачи. Приведены результаты вычислительных экспериментов, проведенных с помощью программного обеспечения, реализующего разработанную модель и алгоритм расчёта, и дан анализ полученных результатов.

Надо отметить, что проблемами моделирования с целью исследования процесса массопереноса в пористых средах занимаются многие исследователи. К настоящему времени получен ряд значительных теоретических и прикладных результатов.

В работе [1] предлагается математическая модель течения в нано-пористых породах. В математической модели предполагается, что фильтрационный слой состоит из двух компонентов: трещиновато-пористой среды и специфических органических включений, состоящих из керогена, содержащих большую часть жидкости: нефти и газа. Вывод модели процесса основан на гипотезе, что проницаемость включений существенно зависит от градиента давления.

Усовершенствованная математическая модель для неравновесных двухфазных (например, вода – масло) потоков в пористых средах приведена в работе [2]. Полученные результаты проведенных расчетов сопоставлены с экспериментальными данными.

В работе [3] приведена модель процесса фильтрации многокомпонентных сред, в которой относительная фазовая проницаемость газовой фазы заменена новым выражением, учитывающим влияние вязкости, плотности и капиллярного эффекта смеси.

Изучение капиллярных давлений, соответствующих треугольному тензору капиллярной диффузии в трехфазной жидкости рассмотрено в работе [4]. Приведенная система — интегродифференциальная, так как искомыми являются суммарный расход и распределение фазовых насыщенностей в условиях заданного перепада давления в одной из фаз на границах области течения. Показано, что в задаче капиллярного вытеснения вырождающаяся система может быть исследована на основе специального принципа максимума.

В работе [5] приведена математическая модель для разработки нефтегазовых месторождений с учетом вероятностного распределения параметров процесса.

Проблемы построения интегрированных (комплексных) математических моделей фильтрации флюидов в пластах и течения газожидкостных смесей в нефтегазосборных сетях трубопроводов рассмотрены в работе [6]. Моделирование такой комплексной системы определяется как процесс вычисления обобщенного решения начально-краевой задачи для системы уравнений, описывающей реальные физические процессы в нефте-

носных пластах, стволах (лифтах) скважин и наземных нефтегазосборных сетях трубопроводов. Предложены и исследованы методы решения систем нелинейных алгебраических уравнений, получаемых после дискретной (сеточной) пространственно-временной аппроксимации начально-краевых задач рассматриваемого класса.

Вопросы математического моделирования процесса неизотермической фильтрации в пористой среде в случае, когда задан полный расход смеси $v = v(t)$ рассматриваются в работе [7].

В работе [8] рассматривается задача о распространении поля давления в низкопроницаемой пористой среде с двумя скважинами, которые соединены техногенной трещиной гидроразрыва. Получено приближенное численное решение этой задачи.

В работе [9] рассмотрена задача переноса в пористой среде трехфазной смеси «вода – газ – нефть» в случае, когда вода содержит мелкодисперсную газовую фазу в виде пузырьков микро- или наноразмеров. Предполагается, что перенос пузырьков, в основном, определяется течением дисперсной фазы (воды). При этом крупные скопления газовой фазы в поровом пространстве, а также вода и нефть переносятся в соответствии с модифицированным законом Дарси для многофазных смесей. Построена математическая модель движения смеси, когда основные фазы (вода, газ, нефть) подчиняются уравнениям фильтрации, а мелкодисперсная газовая фаза описывается кинетическим уравнением типа Больцмана.

Задача фильтрации трех вязких, несжимаемых и взаимно несмешивающихся жидкостей в пористой среде без учета массовых сил и капиллярных давлений между фазами рассмотрена в работе [10]. Там же получено решение для трехфазного течения, аналогичное решению Баклея—Леверетта для двух фаз, и показано, что характер распределения насыщенностей существенно зависит от начального насыщения пористого пласта и фазового состава нагнетаемой смеси.

В работе [11] численно решается и исследуется задача одномерной трехфазной фильтрации в неоднородном пласте с учетом растворимости газа в нефтяной и водной фазах, сжимаемости фаз и пористой среды, а также силы тяжести.

Анализ указанных источников показал, что в исследованиях авторов не рассмотрен процесс двухстороннего вытеснения нефти газом и водой с двух сторон, в результате которого образуются зоны чистого газа, смеси нефти – газа – воды, чистой нефти. В настоящей работе предприняты усилия для восполнения данного пробела. Как было упомянуто выше, математическую модель процесса, происходящего в пластовых условиях, необходимо сформулировать на основе положений механики многофазных сред в виде задачи типа Стефана с неизвестными границами фаз.

Обычно, при моделировании таких процессов используются два принципиально различных подхода. В первом случае, строятся модели поршневого вытеснения с четким выделением границы вытесняемой и вытесняющей фаз, а во втором — строятся модели непоршневого вытеснения без четкого выделения границы.

Во втором случае, считается, что в каждой точке пласта имеется определенное количество из всех фаз, но некоторые из них имеют насыщенность больше, чем остальные и участвуют в движении, а остальные считаются в среде без движения. Таким образом, образуются зоны чистого газа, смеси нефти – газа – воды, чистой нефти.

1. Постановка задачи

Рассмотрим сложные динамические процессы, происходящие в пластовых условиях при двухстороннем вытеснении нефти газом и водой в одномерной постановке в условиях пренебрежения капиллярными давлениями между фазами, растворимостью газа и гравитационными силами. С учетом высказанного, представим для простоты понимания сути рассматриваемого процесса, пласт, в следующем виде (рис. 1). Здесь г — газ, в — вода, $q_{\text{г}}$ и $q_{\text{ж}}$ — соответственно интенсивности работы нагнетательной и эксплуатационной скважин.

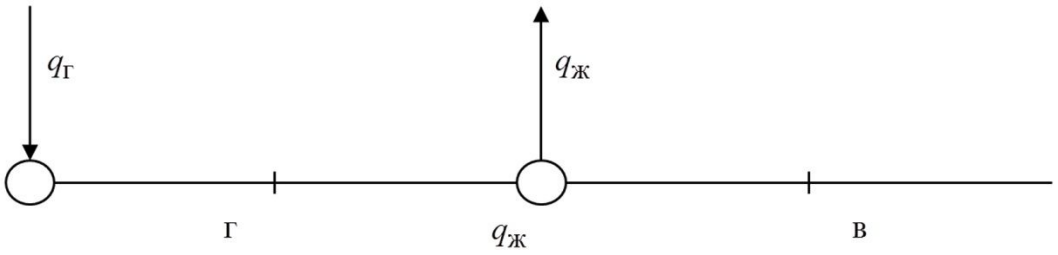


Рис. 1. Продуктивный пласт

Тогда, используя законы газогидродинамики, можно сформулировать математическую модель процесса воздействия на пласт объемом газа и продвижения жидкости в пласте, которая приводится к решению следующей системы нелинейных дифференциальных уравнений в безразмерном виде:

$$\frac{\partial^2 P^2}{\partial x^2} = \frac{K\rho_{\text{н}} RTZ\mu_{\text{г}}}{K_{\text{г}} P_{\text{н}} \mu_{\text{н}} P}\frac{\partial P^2}{\partial \tau},\ 0 < x < \frac{l(\tau)}{L}, \tag{1}$$

$$\begin{cases} \dfrac{\partial}{\partial x}\left(K\dfrac{\partial P}{\partial x}\right) = \dfrac{\partial}{\partial \tau}\left(PS_{\text{г}}\right) + B\dfrac{\partial}{\partial \tau}\left(S_{\text{н}} + B_{\text{вн}} + S_{\text{в}}\right), \\ \dfrac{\partial}{\partial x}\left(K_{\text{н}}\dfrac{\partial P}{\partial x}\right) = B\dfrac{\partial S_{\text{н}}}{\partial \tau}, \\ \dfrac{\partial}{\partial x}\left(K_{\text{в}}\dfrac{\partial P}{\partial x}\right) = B\dfrac{\mu_{\text{в}}}{\mu_{\text{н}}}\dfrac{\partial S_{\text{в}}}{\partial \tau},\ \dfrac{l(\tau)}{L} < x < 1, \\ S_{\text{н}} + S_{\text{в}} + S_{\text{г}} = 1, \end{cases} \tag{2}$$

$$P(x,\tau) = f(x,\tau),\ 1 < x < \infty,\ \tau > 0, \tag{3}$$

$$\left.\frac{\partial P^2}{\partial x}\right|_{x=0} = -\alpha\frac{L}{P_{\text{н}}}q_{\text{г}}, \tag{4}$$

$$\left.\frac{\partial P}{\partial x}\right|_{\zeta_i+0} - \left.\frac{\partial P}{\partial x}\right|_{\zeta_i-0} = \beta\frac{L}{P_{\text{н}}}q_{\text{ж}}, \tag{5}$$

$$\left.\frac{K_{\text{г}}P_{\text{н}}}{2\rho_{\text{г}}RTZ}\frac{\partial P^2}{\partial x}\right|_{x=\frac{l(\tau)}{L}-0} = K\left(\frac{\rho_{\text{н}}K_{\text{н}}}{\mu_{\text{н}}} + \frac{\rho_{\text{в}}K_{\text{в}}}{\mu_{\text{в}}} + \frac{K_{\text{г}}P_{\text{н}}P}{RTZ\mu_{\text{г}}}\right)\left.\frac{\partial P}{\partial x}\right|_{x=\frac{l(\tau)}{L}+0}, \tag{6}$$

$$\frac{dl}{d\tau} = -\left.\frac{\partial P}{\partial x}\right|_{x=\frac{l(\tau)}{L}+0}, \tag{7}$$

$$P_{\text{г}}\Big|_{x=\frac{l(\tau)}{L}-0} = P_{\text{н}}\Big|_{x=\frac{l(\tau)}{L}+0}, \tag{8}$$

$$\left.\begin{array}{l} P(x,0)=P_{\text{н}},\ S_{\text{н}}(x,0)=S_{\text{н}}^0,\ S_{\text{г}}(x,0)=S_{\text{г}}^0,\\ S_{\text{в}}(x,0)=S_{\text{в}}^0,\ l(0)=l^0,\ 0<x<1. \end{array}\right\} \tag{9}$$

В формулах (1) – (9) приняты следующие обозначения: $P(x,\tau)$ — давление; $S_{\text{г}}$, $S_{\text{н}}$, $S_{\text{в}}$ — насыщенность породы газом, нефтью и водой; K — абсолютная проницаемость породы; $K_{\text{г}}$, $K_{\text{н}}$, $K_{\text{в}}$, $\mu_{\text{г}}$, $\mu_{\text{н}}$, $\mu_{\text{в}}$ — соответственно фазовые и вязкости газа, нефти и воды; $P(x,0)=P_{\text{н}}$, $S_{\text{г}}(x,0)=S_{\text{г}}^0$, $S_{\text{н}}(x,0)=S_{\text{н}}^0$, $S_{\text{в}}(x,0)=S_{\text{в}}^0$ — соответственно начальные распределения давления, газонасыщенности, нефтенасыщенности и водонасыщенности; $\rho_{\text{г}}$, $\rho_{\text{н}}$, $\rho_{\text{в}}$ — соответственно плотность газа, нефти и воды; R — газовая постоянная; T — абсолютная температура; Z — сжимаемость газа; $P_{\text{н}}$ — начальное пластовое давление; m — пористость пласта; ζ_i — внутренняя особая точка (нагнетательная или эксплуатационная скважина); $x=l(\tau)$ — граница раздела; L — длина пласта; $q_{\text{г}}$, $q_{\text{ж}}$ — соответственно интенсивности работы нагнетательной и эксплуатационной скважин; α, β — некоторые постоянные величины для приведения в размерность.

Таким образом, получена замкнутая система нелинейных дифференциальных уравнений, описывающая функционирование системы «пласт – скважина», которая состоит из трех частей: первая описывает функционирование воздействия через нагнетательные скважины, вторая описывает динамику продвижения смеси нефти, воды и газа в пласте, а третья динамику продвижения краевой воды.

2. Метод решения

Для численного решения рассматриваемой задачи применим основные идеи метода выпрямления фазовых фронтов.

Пусть требуется найти функции $P(x,\tau)$, $S_{\text{г}}(x,\tau)$, $S_{\text{н}}(x,\tau)$, $S_{\text{в}}(x,\tau)$, $l(\tau)$, удовлетворяющие условиям (1) – (9). Для решения данной задачи введем новые независимые переменные

$$\xi=\frac{x}{l(\tau)}\ \text{ и }\ \varsigma=1+\frac{x-l(\tau)}{1-l(\tau)}$$

и получим:

$$\left\{\begin{array}{l} \dfrac{\partial^2 P^2}{\partial\xi^2}=\dfrac{\lambda l(\tau)}{P}\left[l(\tau)\dfrac{\partial P^2}{\partial\tau}-l'(\tau)\xi\dfrac{\partial P^2}{\partial\xi}\right],\ 0<\xi<1,\\ \dfrac{\partial}{\partial\varsigma}\left(K_{\text{н}}\dfrac{\partial P}{\partial\varsigma}\right)=\left(1-l(\tau)\right)^2\left[\dfrac{\partial(PS_{\text{г}})}{\partial\tau}+B\dfrac{\partial}{\partial\tau}\left(S_{\text{н}}+CS_{\text{в}}\right)\right]-\\ -\left(1-l(\tau)\right)(2-\varsigma)l'(\tau)\left[\dfrac{\partial(PS_{\text{г}})}{\partial\tau}+B\dfrac{\partial}{\partial\tau}\left(S_{\text{н}}+CS_{\text{в}}\right)\right],\\ \dfrac{\partial}{\partial\varsigma}\left(K_{\text{н}}\dfrac{\partial P}{\partial\varsigma}\right)=B\left(1-l(\tau)\right)^2\left[\dfrac{\partial(S_{\text{н}})}{\partial\tau}-l'(\tau)\dfrac{2-\varsigma}{1-l(\tau)}\dfrac{\partial S_{\text{н}}}{\partial\varsigma}\right],\\ \dfrac{\partial}{\partial\varsigma}\left(K_{\text{в}}\dfrac{\partial P}{\partial\varsigma}\right)=B\left(1-l(\tau)\right)^2\dfrac{\mu_{\text{в}}}{\mu_{\text{н}}}\left[\dfrac{\partial S_{\text{в}}}{\partial\tau}-l'(\tau)\dfrac{2-\varsigma}{1-l(\tau)}\dfrac{\partial S_{\text{в}}}{\partial\varsigma}\right],\\ S_{\text{г}}+S_{\text{н}}+S_{\text{в}}=1,\ 0\le\varsigma\le1,\ P(L,\tau)=1,\ l(\tau)\le x\le L. \end{array}\right. \tag{10}$$

$$\left.\frac{\partial P^2}{\partial \xi}\right|_{\xi=0} = -\alpha_{\text{н}} q_{\text{г}}, \tag{11}$$

$$\frac{dl}{d\tau} = -\left.\frac{\partial P}{l(\tau)\partial \xi}\right|_{\xi=l-0}, \tag{12}$$

$$D\left.\frac{\partial P^2}{l(\tau)\partial \xi}\right|_{\xi=l-0} = \frac{1}{1-l(\tau)}\left[K\left(\frac{\rho_{\text{н}}K_{\text{н}}}{\mu_{\text{н}}} + \frac{\rho_{\text{в}}K_{\text{в}}}{\mu_{\text{в}}} + \frac{K_{\text{г}}P_{\text{н}}P}{RTZ\mu_{\text{г}}}\right)\right]\left.\frac{\partial P}{\partial \xi}\right|_{\xi=l+0}, \tag{13}$$

$$0 \le \xi \le 1,\ 0 \le x \le l(\tau),$$

где $\lambda = \dfrac{K\rho_{\text{н}}RZT\mu_{\text{г}}}{K_{\text{г}}P_{\text{н}}\mu_{\text{н}}P}$; $B = \dfrac{\rho_{\text{н}}RZT}{P_{\text{н}}}$; $C = \dfrac{\rho_{\text{в}}}{\rho_{\text{н}}}$; $\alpha_{\text{н}} = \alpha\dfrac{l(\tau)L}{P_{\text{н}}}$; $D = \dfrac{K_{\text{г}}P_{\text{н}}}{2\rho_{\text{г}}RZT}$; $P_{\text{н}}$ — начальное распределение функции состояния.

Система уравнений, описывающая сформулированную задачу, нелинейная относительно искомых функций $l(\tau)$, $P(x,\tau)$, $S_{\text{н}}(x,\tau)$, $S_{\text{в}}(x,\tau)$, $S_{\text{г}}(x,\tau)$. Поэтому получить точное аналитическое решение задачи невозможно. Для ее решения применяем метод конечной разности.

Дискретный алгоритм решения задачи (10) – (13) основан на применении интегро-интерполяционного метода, позволяющего построить консервативную разностную схему, которая удовлетворяет закону сохранения в каждом узле пространственно-временной сетки.

Для первого уравнения системы (10) запишем интегральный закон сохранения на отрезке $\xi_{i-1/2} \le \xi_i \le \xi_{i+1/2}$, $\xi_{i-1/2} = \xi_i - h/2$, $x_i = ih$:

$$\int_{\xi_{i-1/2}}^{\xi_{i+1/2}} \frac{\partial^2 P^2}{\partial \xi^2} d\xi = \lambda l^2 \int_{\xi_{i-1/2}}^{\xi_{i+1/2}} \frac{1}{P}\frac{\partial P^2}{\partial \tau} d\xi - \lambda l l' \int_{\xi_{i-1/2}}^{\xi_{i+1/2}} \frac{\xi}{P}\frac{\partial P^2}{\partial \xi} d\xi. \tag{14}$$

В результате получаем следующую систему конечно-разностных уравнений:

$$A_i P_{i-1}^2 - C_i P_i^2 + B_i P_{i+1}^2 = -F_i,\ i = \overline{2,N}. \tag{15}$$

В (15) коэффициенты трехточечных уравнений определяются из следующих выражений

$$A_i = 1 - \frac{\xi l h\lambda\left(l_{\text{к}} - \overline{l}_{\text{к}}\right)}{2P_i^0\tau},\ B_i = 1 + \frac{\xi l h\lambda\left(l_{\text{к}} - \overline{l}_{\text{к}}\right)}{2P_i^0\tau},\ C_i = 2 + \frac{h^2 l^2\lambda}{\tau P_i^0},\ F_i = \frac{h^2 l^2}{\tau P_i^0}\lambda\overline{P}_i^2.$$

Применяя интегро-интерполяционный метод для второго уравнения системы (10), получим

$$\int_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}} \frac{\partial}{\partial \varsigma}\left(K\frac{\partial P}{\partial \varsigma}\right) d\varsigma = (1-l)^2 \int_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}} \left[\frac{\partial\left(PS_{\text{г}}\right)}{\partial \tau} + B\frac{\partial}{\partial \tau}\left(S_{\text{н}} + CS_{\text{в}}\right)\right] d\varsigma -$$

$$-(1-l)l' \int_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}} (2-\varsigma)\left[\frac{\partial\left(PS_{\text{г}}\right)}{\partial \varsigma} + B\frac{\partial}{\partial \varsigma}\left(S_{\text{н}} + CS_{\text{в}}\right)\right] d\varsigma.$$

Для правых частей интеграла применяем теорему о среднем и получаем:

$$\int_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}} (2-\varsigma)\frac{\partial\left(PS_{\text{г}}\right)}{\partial \varsigma} d\varsigma = \int_{\varsigma_{i-1/2}}^{\varsigma_{i-0}} (2-\varsigma)\frac{\partial\left(PS_{\text{г}}\right)}{\partial \varsigma} d\varsigma + \int_{\varsigma_{i+0}}^{\varsigma_{i+1/2}} (2-\varsigma)\frac{\partial\left(PS_{\text{г}}\right)}{\partial \varsigma} d\varsigma = \left(2-\varsigma_{i_1}\right)\int_{\varsigma_{i-1/2}}^{\varsigma_{i-0}} \frac{\partial\left(PS_{\text{г}}\right)}{\partial \varsigma} d\varsigma +$$

$$+\left(2-\varsigma_{i_2}\right)\int_{\varsigma_{i+0}}^{\varsigma_{i+1/2}} \frac{\partial\left(PS_{\text{г}}\right)}{\partial \varsigma} d\varsigma = \left(2-\varsigma_{i_1}\right)\frac{\left(PS_{\text{г}}\right)_i - \left(PS_{\text{г}}\right)_{i-1}}{2} + \left(2-\varsigma_{i_2}\right)\frac{\left(PS_{\text{г}}\right)_{i+1} - \left(PS_{\text{г}}\right)_i}{2};$$

$$\int_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}}(2-\varsigma)\frac{\partial}{\partial\varsigma}\left(S_{\text{н}}+CS_{\text{в}}\right)d\varsigma=\int_{\varsigma_{i-1/2}}^{\varsigma_{i-0}}(2-\varsigma)\frac{\partial}{\partial\varsigma}\left(S_{\text{н}}+CS_{\text{в}}\right)d\varsigma+$$

$$+\int_{\varsigma_{i+0}}^{\varsigma_{i+1/2}}(2-\varsigma)\frac{\partial}{\partial\varsigma}\left(S_{\text{н}}+CS_{\text{в}}\right)d\varsigma=(2-\varsigma_1)\int_{\varsigma_{i-1/2}}^{\varsigma_{i-0}}\frac{\partial}{\partial\varsigma}\left(S_{\text{н}}+CS_{\text{в}}\right)d\varsigma+$$

$$+(2-\varsigma_2)\int_{\varsigma_{i+0}}^{\varsigma_{i+1/2}}\frac{\partial}{\partial\varsigma}\left(S_{\text{н}}+CS_{\text{в}}\right)d\varsigma=(2-\varsigma_1)\frac{\left(S_{\text{н}}+CS_{\text{в}}\right)_i-\left(S_{\text{н}}+CS_{\text{в}}\right)_{i-1}}{2}+$$

$$+(2-\varsigma_2)\frac{\left(S_{\text{н}}+CS_{\text{в}}\right)_{i+1}-\left(S_{\text{н}}+CS_{\text{в}}\right)_i}{2}.$$

С учетом полученных результатов для второго уравнения системы (10) получаем

$$\left[K_{i-1/2}-h^2(1-l)l'\left(2-\varsigma_{i_1}\right)\frac{S_{\Gamma_{i-1}}}{2}\right]P_{i-1}-\left[K_{i-1/2}+K_{i+1/2}+(1-l)^2\frac{h^2}{\tau}S_{\Gamma_i}+\right.$$

$$\left.+h^2(1-l)l'\frac{S_{\Gamma_i}}{2}\left(\varsigma_{i_1}-\varsigma_{i_2}\right)\right]P_i+\left[K_{i+1/2}+h^2(1-l)l'\left(2-\varsigma_{i_2}\right)\frac{S_{\Gamma_{i+1}}}{2}\right]P_{i+1}=$$

$$=-(1-l)^2\frac{h^2}{\tau}\left(\overline{PS}_{\Gamma}\right)_i+(1-l)^2Bh^2\frac{\left(S_{\text{н}}+CS_{\text{в}}\right)_i-\left(S_{\text{н}}+CS_{\text{в}}\right)_{i+1}}{\tau}-$$

$$-h^2(1-l)l'\left\{B(2-\varsigma_1)\frac{\left(S_{\text{н}}+CS_{\text{в}}\right)_i-\left(S_{\text{н}}+CS_{\text{в}}\right)_{i-1}}{2}+B(2-\varsigma_2)\frac{\left(S_{\text{н}}+CS_{\text{в}}\right)_{i+1}-\left(S_{\text{н}}+CS_{\text{в}}\right)_i}{2}\right\}.$$

Отсюда получаем следующую трехточечную систему уравнений

$$A_iP_{i-1}-C_iP_i+B_iP_{i+1}=-F_i,\ i=\overline{1,N}. \tag{16}$$

В ((16) коэффициенты A_i, B_i, C_i, F_i определяются следующими выражениями:

$$A_i=K_{i-1/2}-h^2(1-l)l'\left(2-\varsigma_{i_1}\right)\frac{S_{\Gamma_{i-1}}}{2};\ B_i=K_{i+1/2}+h^2(1-l)l'\left(2-\varsigma_{i_2}\right)\frac{S_{\Gamma_{i+1}}}{2};$$

$$C_i=K_{i-1/2}+K_{i+1/2}+(1-l)^2\frac{h^2}{\tau}S_{\Gamma_i}+h^2(1-l)l'\frac{S_{\Gamma_i}}{2}\left(\varsigma_{i_1}-\varsigma_{i_2}\right);$$

$$F_i=(1-l)^2\frac{h^2}{\tau}\left(\overline{PS}_{\Gamma}\right)_i+(1-l)^2Bh^2\frac{\left(S_{\text{н}}+CS_{\text{в}}\right)_i-\left(S_{\text{н}}+CS_{\text{в}}\right)_{i+1}}{\tau}-$$

$$-h^2(1-l)l'\left\{B(2-\varsigma_1)\frac{\left(S_{\text{н}}+CS_{\text{в}}\right)_i-\left(S_{\text{н}}+CS_{\text{в}}\right)_{i-1}}{2}+B(2-\varsigma_2)\frac{\left(S_{\text{н}}+CS_{\text{в}}\right)_{i+1}-\left(S_{\text{н}}+CS_{\text{в}}\right)_i}{2}\right\}.$$

Применяя интегро-интерполяционный метод, для третьего уравнения системы (10), получаем

$$\int_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}}\frac{\partial}{\partial\varsigma}\left(K_{\text{н}}\frac{\partial P}{\partial\varsigma}\right)d\varsigma=B(1-l)^2\int_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}}\frac{\partial S_{\text{н}}}{\partial\tau}d\varsigma-\frac{l'}{1-l}\int_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}}(2-\varsigma)\frac{\partial S_{\text{н}}}{\partial\varsigma}d\varsigma.$$

Для второго интеграла правых частей применяем теорему о среднем

$$\int\limits_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}}(2-\varsigma)\frac{\partial S_{\text{н}}}{\partial\varsigma}d\varsigma=\int\limits_{\varsigma_{i-1/2}}^{\varsigma_{i-0}}(2-\varsigma)\frac{\partial S_{\text{н}}}{\partial\varsigma}d\varsigma+\int\limits_{\varsigma_{i+0}}^{\varsigma_{i+1/2}}(2-\varsigma)\frac{\partial S_{\text{н}}}{\partial\varsigma}d\varsigma=$$

$$=(2-\varsigma_1)\int\limits_{\varsigma_{i-1/2}}^{\varsigma_{i-0}}\frac{\partial S_{\text{н}}}{\partial\varsigma}d\varsigma+(2-\varsigma_2)\int\limits_{\varsigma_{i+0}}^{\varsigma_{i+1/2}}\frac{\partial S_{\text{н}}}{\partial\varsigma}d\varsigma=(2-\varsigma_1)\left(S_{\text{н}_{i-0}}-S_{\text{н}_{i-1/2}}\right)+(2-\varsigma_2)\left(S_{\text{н}_{i+1/2}}-S_{\text{н}_{i+0}}\right)=$$

$$=(2-\varsigma_1)\left(\frac{S_{\text{н}_i}-S_{\text{н}_{i-1}}}{2}\right)+(2-\varsigma_2)\left(\frac{S_{\text{н}_{i+1}}-S_{\text{н}_i}}{2}\right)=(2-\varsigma_1)\frac{S_{\text{н}_i}-S_{\text{н}_{i-1}}}{2}+(2-\varsigma_2)\frac{S_{\text{н}_{i+1}}-S_{\text{н}_i}}{2},$$

и получаем в итоге следующую систему конечно-разностных уравнений

$$Bh^2(1-l)l'\frac{2-\varsigma_1}{2}S_{\text{н}_{i-1}}-\left[Bh^2(1-l)l'\frac{2-\varsigma_1}{2}-Bh^2(1-l)l'\frac{2-\varsigma_2}{2}-B\frac{h^2(1-l)^2}{\tau}\right]S_{\text{н}_i}-$$

$$-Bh^2(1-l)l'\frac{2-\varsigma_2}{2}S_{\text{н}_{i+1}}=K_{\text{н}_{i+1/2}}(P_{i+1}-P_i)-K_{\text{н}_{i-1/2}}(P_i-P_{i-1})+B\frac{h^2(1-l)^2}{\tau}\overline{S}_{\text{н}_i},$$

которую можно записать в виде трехточечного уравнения относительно нефтенасыщенности

$$A_iS_{\text{н}_{i-1}}-C_iS_{\text{н}_i}+B_iS_{\text{н}_{i+1}}=-F_i. \tag{17}$$

В (17) коэффициенты полученного трехточечного уравнения определяются с помощью следующих выражений:

$$A_i=Bh^2(1-l)l'\frac{2-\varsigma_1}{2};\ B_i=-Bh^2(1-l)l'\frac{2-\varsigma_2}{2};$$

$$C_i=Bh^2(1-l)l'\frac{2-\varsigma_1}{2}-Bh^2(1-l)l'\frac{2-\varsigma_2}{2}-B\frac{h^2(1-l)^2}{\tau};$$

$$F_i=K_{\text{н}_{i-1/2}}(P_i-P_{i-1})-K_{\text{н}_{i+1/2}}(P_{i+1}-P_i)-B\frac{h^2(1-l)^2}{\tau}\overline{S}_{\text{н}_i}.$$

Применяя аналогично интегро-интерполяционный метод для четвертого уравнения системы (10), получаем

$$\int\limits_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}}\frac{\partial}{\partial\varsigma}\left(K_{\text{в}}\frac{\partial P}{\partial\varsigma}\right)d\varsigma=B(1-l)^2\frac{\mu_{\text{в}}}{\mu_{\text{н}}}\int\limits_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}}\frac{\partial S_{\text{в}}}{\partial\tau}d\varsigma-$$

$$-Bl'(1-l)\frac{\mu_{\text{в}}}{\mu_{\text{н}}}\int\limits_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}}(2-\varsigma)\frac{\partial S_{\text{в}}}{\partial\varsigma}d\varsigma. \tag{18}$$

Для второго интеграла правых частей, применяя теорему о среднем, получим

$$\int\limits_{\varsigma_{i-1/2}}^{\varsigma_{i+1/2}}(2-\varsigma)\frac{\partial S_{\text{в}}}{\partial\varsigma}d\varsigma=\int\limits_{\varsigma_{i-1/2}}^{\varsigma_{i-0}}(2-\varsigma)\frac{\partial S_{\text{в}}}{\partial\varsigma}d\varsigma+\int\limits_{\varsigma_{i+0}}^{\varsigma_{i+1/2}}(2-\varsigma)\frac{\partial S_{\text{в}}}{\partial\varsigma}d\varsigma=$$

$$=(2-\varsigma_1)\int\limits_{\varsigma_{i-1/2}}^{\varsigma_{i-0}}\frac{\partial S_{\text{в}}}{\partial\varsigma}d\varsigma+(2-\varsigma_2)\int\limits_{\varsigma_{i+0}}^{\varsigma_{i+1/2}}\frac{\partial S_{\text{в}}}{\partial\varsigma}d\varsigma=(2-\varsigma_1)\left(S_{\text{в}_{i-0}}-S_{\text{в}_{i-1/2}}\right)+$$

$$+(2-\varsigma_2)\left(S_{\text{в}_{i+1/2}}-S_{\text{в}_{i+0}}\right)=(2-\varsigma_1)\left(\frac{S_{\text{в}_i}-S_{\text{в}_{i-1}}}{2}\right)+(2-\varsigma_2)\left(\frac{S_{\text{в}_{i+1}}-S_{\text{в}_i}}{2}\right).$$

С учетом приведенного выше, получим следующую систему конечно-разностных уравнений

$$K_{\text{в}\,i+1/2}\frac{P_{i+1}-P_i}{h}-K_{\text{в}\,i-1/2}\frac{P_i-P_{i-1}}{h}-\delta_j q_{\text{в}}=hB(1-l)^2\frac{\mu_{\text{в}}}{\mu_{\text{н}}}\frac{S_{\text{в}_i}-\overline{S}_{\text{в}_i}}{\tau}-$$

$$-hB(1-l)^2\frac{\mu_{\text{в}}}{\mu_{\text{н}}}\frac{S_{\text{в}_i}-\overline{S}_{\text{в}_i}}{\tau}-hB(1-l)l'\frac{\mu_{\text{в}}}{\mu_{\text{н}}}\left[(2-\varsigma_1)\frac{S_{\text{в}_i}-S_{\text{в}_{i-1}}}{2}+(2-\varsigma_2)\frac{S_{\text{в}_{i+1}}-S_{\text{в}_i}}{2}\right].$$

После некоторых преобразований, получим

$$\frac{\mu_{\text{в}}}{\mu_{\text{н}}}Bh^2(1-l)l'\frac{2-\varsigma_1}{2}S_{\text{в}_{i-1}}-\left[\frac{\mu_{\text{в}}}{\mu_{\text{н}}}Bh^2(1-l)l'\frac{2-\varsigma_1}{2}-\right.$$

$$\left.-\frac{\mu_{\text{в}}}{\mu_{\text{н}}}Bh^2(1-l)l'\frac{2-\varsigma_2}{2}-\frac{\mu_{\text{в}}}{\mu_{\text{н}}}B\frac{h^2(1-l)^2}{\tau}\right]S_{\text{в}_i}-\frac{\mu_{\text{в}}}{\mu_{\text{н}}}Bh^2(1-l)l'\frac{2-\varsigma_2}{2}S_{\text{в}_{i+1}}=$$

$$=K_{\text{в}\,i+1/2}(P_{i+1}-P_i)-K_{\text{в}\,i-1/2}(P_i-P_{i-1})+\frac{\mu_{\text{в}}}{\mu_{\text{н}}}B\frac{h^2(1-l)^2}{\tau}\overline{S}_{\text{в}_i}.$$

или

$$A_iS_{\text{в}_{i-1}}-C_iS_{\text{в}_i}+B_iS_{\text{в}_{i+1}}=-F_i. \tag{19}$$

В (19) коэффициенты определяются следующими равенствами:

$$A_i=\frac{\mu_{\text{в}}}{\mu_{\text{н}}}Bh^2(1-l)l'\frac{2-\varsigma_1}{2},\ B_i=-\frac{\mu_{\text{в}}}{\mu_{\text{н}}}Bh^2(1-l)l'\frac{2-\varsigma_2}{2},$$

$$C_i=\frac{\mu_{\text{в}}}{\mu_{\text{н}}}Bh^2(1-l)l'\frac{2-\varsigma_1}{2}-\frac{\mu_{\text{в}}}{\mu_{\text{н}}}Bh^2(1-l)l'\frac{2-\varsigma_2}{2}-\frac{\mu_{\text{в}}}{\mu_{\text{н}}}B\frac{h^2(1-l)^2}{\tau},$$

$$F_i=K_{\text{в}\,i-1/2}(P_i-P_{i-1})-K_{\text{в}\,i+1/2}(P_{i+1}-P_i)-\frac{\mu_{\text{в}}}{\mu_{\text{н}}}B\frac{h^2(1-l)^2}{\tau}\overline{S}_{\text{в}_i},$$

где $\varsigma_{i_1}\in\left[\varsigma_{i-1/2},\ \varsigma_{i-0}\right]$, $\varsigma_{i_2}=\left[\varsigma_{i+0},\ \varsigma_{i+1/2}\right]$ и принято $\varsigma_{i_1}=\xi-\frac{h_2}{4}$, $\varsigma_{i_2}=\xi+\frac{h_2}{4}$.

Используя условия, при которых закачивается газ (10)

$$\frac{-3P_0^2+4P_1^2-P_2^2}{2hl_{\text{к}}}=-\alpha_{\text{н}}q_{\text{г}}$$

и при $i=0$ находим на границе прогоночные коэффициенты

$$\alpha_1=\frac{C_1-4B_1}{A_1-3B_1},\ \beta_1=-\frac{B_1}{A_1-3B_1}\left(2hl_{\text{к}}\alpha_{\text{н}}q_{\text{г}}+\frac{F_1}{B_1}\right).$$

Граница раздела определяется следующей формулой

$$l_{\text{к}}=\sqrt{\overline{l}_{\text{к}}^2-\frac{\tau}{h}\left(3P_{\xi_i}-4P_{\xi_{i-1}}+P_{\xi_{i-2}}\right)}.$$

Исходя из условий (13), получаем

$$D\frac{3P_l^2-4P_{l-1}^2+P_{l-2}^2}{2hl_{\text{к}}}=\frac{1}{2h(1-l_{\text{к}})}\left[\frac{K_{\text{г}}}{\mu_{\text{г}}}+\frac{K_{\text{н}}}{\mu_{\text{н}}}+\frac{K_{\text{в}}}{\mu_{\text{в}}}\right](4P_2-3P_1+P_3).$$

После некоторых преобразований будем иметь

$$\frac{(1-l_{\text{к}})D}{l_{\text{к}}\left(\frac{K_{\text{г}}}{\mu_{\text{г}}}+\frac{K_{\text{н}}}{\mu_{\text{н}}}+\frac{K_{\text{в}}}{\mu_{\text{в}}}\right)}\left[3-\alpha_l(4-\alpha_{l-1})\right]P_l^2+$$

$$+\frac{(1-l_{\text{к}})D}{l_{\text{к}}\left(\frac{K_{\text{г}}}{\mu_{\text{г}}}+\frac{K_{\text{н}}}{\mu_{\text{н}}}+\frac{K_{\text{в}}}{\mu_{\text{в}}}\right)}\left[\beta_{l-1}-\beta_l(4-\alpha_{l-1})\right]=-\left[3-(4-\alpha_3)\alpha_2\right]P_1+(4-\alpha_3)\beta_2-\beta_3.$$

Из условия (11) получаем $P_l = P_1 = P$, откуда относительно неизвестных давлений – квадратное уравнение

$$D_1 P^2 + D_2 P + D_3 = 0,$$

где

$$D_1 = \frac{(1-l_к)D}{l_к\left(\frac{K_г}{\mu_г}+\frac{K_н}{\mu_н}+\frac{K_в}{\mu_в}\right)}\left[3-\alpha_l\left(4-\alpha_{l-1}\right)\right],\ D_2 = 3-\left(4-\alpha_3\right)\alpha_2,$$

$$D_3 = \frac{(1-l_к)D}{l_к\left(\frac{K_г}{\mu_г}+\frac{K_н}{\mu_н}+\frac{K_в}{\mu_в}\right)}\left[\beta_{l-1}-\beta_l\left(4-\alpha_{l-1}\right)\right]-\left(4-\alpha_3\right)\beta_2-\beta_3.$$

Решая квадратное уравнение, находим в этой точке давления

$$P_{1,2} = \frac{-D_2 \pm \sqrt{D_2^2 - 4D_1D_3}}{2D_1}.$$

Полученная нелинейная система уравнений решена методом прогонки в каждом шаге с применением метода простой итерации. При этом организуются внутренняя и внешняя итерации. Во внутренней за критерий точности при определении давлений берем

$$\max_i \left|P_i^{(S)} - P_i^{(S-1)}\right| < \varepsilon_1.$$

При определении нефтенасыщенности за критерий точности берем

$$\max_i \left|S_{н_i}^{(\sigma_1)} - S_{н_i}^{(\sigma_1-1)}\right| < \varepsilon_2.$$

При определении водонасыщенности за критерий точности берем

$$\max_i \left|S_{в_i}^{(\sigma_2)} - S_{в_i}^{(\sigma_2-1)}\right| < \varepsilon_3.$$

За критерий точности при определении продвижения границы раздела берем

$$\left|l_к^{\sigma_3} - \overline{l}_к^{(\sigma_3-1)}\right| < \varepsilon_4,$$

где σ_1, σ_2, σ_3 — число итерационных процессов по давлению, насыщенности и продвижению границы раздела, а ε_i $(i = \overline{1,4})$, некоторые постоянные величины.

Здесь для определения фазовой проницаемости в одномерной и двухмерной постановках используются зависимости приведенные в работе [12].

Для определения количества извлеченной воды, нефти и газа, можно использовать обобщенный закон Дарси

$$Q_н = -A\frac{кK_н}{\mu_н}\frac{P_н}{L}gradP,\ Q_г = -A\frac{кK_г}{\mu_г}\frac{P_н}{L}gradP,\ Q_в = -A\frac{кK_в}{\mu_в}\frac{P_н}{L}gradP.$$

Умножая объемные расходы фаз на соответствующие плотности и суммируя их, получаем массовый расход

$$Q_m^м = \rho_н Q_н + \rho_г Q_г + \rho_в Q_в = -Aк\frac{P_н}{L}\left(\frac{K_н}{\mu_н}\rho_н + \frac{K_г}{\mu_г}\rho_г + \frac{K_в}{\mu_в}\rho_в\right)\frac{\partial P}{\partial x} =$$

$$= -Aк\frac{P_н}{L}\left(\frac{K_н}{\mu_н}\rho_н + \frac{K_в}{\mu_в}\rho_в + \frac{K_г}{\mu_г}\frac{P_н P}{RTZ}\right)\frac{\partial P}{\partial x} = -Aк\frac{P_н}{L}\frac{\rho_н}{\mu_н}\left(K_н + \frac{\mu_н}{\rho_н}\frac{P_н}{\mu_г RTZ}K_г P + \frac{\mu_н}{\rho_н}\frac{\rho_в}{\mu_в}K_в\right)\frac{\partial P}{\partial x}.$$

Отсюда

$$Q_m^м = -Aк\frac{\rho_н P_н}{\mu_н L}KgradP.$$

Исходя из общего количества извлеченной жидкости, определяем отдельно нефть, газ и воду соответственно для внутренних галерей:

$$q_{\text{н}} = \frac{K_{\text{н}}\left(S_{\text{н}_i}, S_{\text{в}_i}\right)}{2h}\left[-6P_i + 4\left(P_{i+1} + P_{i-1}\right) - \left(P_{i+2} + P_{i-2}\right)\right],$$

$$q_{\text{г}} = \frac{A_{\text{гн}} P_i K_{\text{г}}\left(S_{\text{н}_i}, S_{\text{в}_i}\right)}{2h}\left[-6P_i + 4\left(P_{i+1} + P_{i-1}\right) - \left(P_{i+2} + P_{i-2}\right)\right],$$

$$q_{\text{в}} = \frac{A_{\text{вн}} K_{\text{в}}\left(S_{\text{н}_i}, S_{\text{в}_i}\right)}{2h}\left[-6P_i + 4\left(P_{i+1} + P_{i-1}\right) - \left(P_{i+2} + P_{i-2}\right)\right].$$

Когда галерея находится на левой границе, количества извлеченной жидкости для каждой фазы определяем следующими формулами

$$q_{\text{н}} = \frac{K_{\text{н}}\left(S_{\text{н}_i}, S_{\text{в}_i}\right)}{2h}\left(-3P_0 + 4P_1 - P_2\right),\ q_{\text{г}} = \frac{A_{\text{гн}} P_i K_{\text{г}}\left(S_{\text{н}_i}, S_{\text{в}_i}\right)}{2h}\left(-3P_0 + 4P_1 - P_2\right),$$

$$q_{\text{в}} = \frac{A_{\text{вн}} K_{\text{в}}\left(S_{\text{н}_i}, S_{\text{в}_i}\right)}{2h}\left(-3P_0 + 4P_1 - P_2\right).$$

Аналогичными формулами определяются количества извлеченных жидкостей для каждой фазы, когда галерея скважин находится на правой границе области определения.

Таким образом, получены математическая модель и вычислительный алгоритм процесса совместной фильтрации нефти, газа и воды в пласте, при помощи которых можно изучать данный процесс при упругом расширении газовой зоны, а также при воздействии на нефтяной пласт газом.

3. Обсуждение результатов

На основе описанного алгоритма было разработано программное обеспечение для анализа динамического состояния и управления объектом с учетом совместного движения трехфазной среды при различных условиях функционирования пластовой системы. Программное обеспечение разработано на языке программирования Delphi для ОС Windows и включает в себя управляющий модуль для ввода данных, модуль произведения расчетов и модуль анализа и интерпретации результатов расчетов.

Вычислительные эксперименты проводили при следующих входных значениях программы: протяженность пласта $L = 10$ км; мощность $B = 6$ м; ширина $H = 700$ м; вязкость нефти $\mu_{\text{н}} = 4$ сп; количество добываемой жидкости $q_m = 1000$ т/сутки; $m = 0,2$; $K = 0,1$ дарси; $\mu_{\text{г}} = 0,2$ сП; $P_{\text{н}} = 200$ ат; $\rho_{\text{н}} = 0,85$ г/см^3; $R = 8,31$ Дж/(моль$\cdot$К); $T = 273$ К; $Z = 40$; $N = 50$; $EPS = 0,0001$.

Результаты проведенных вычислительных экспериментов приведены в работах [13-15] и на рис. 2–7.

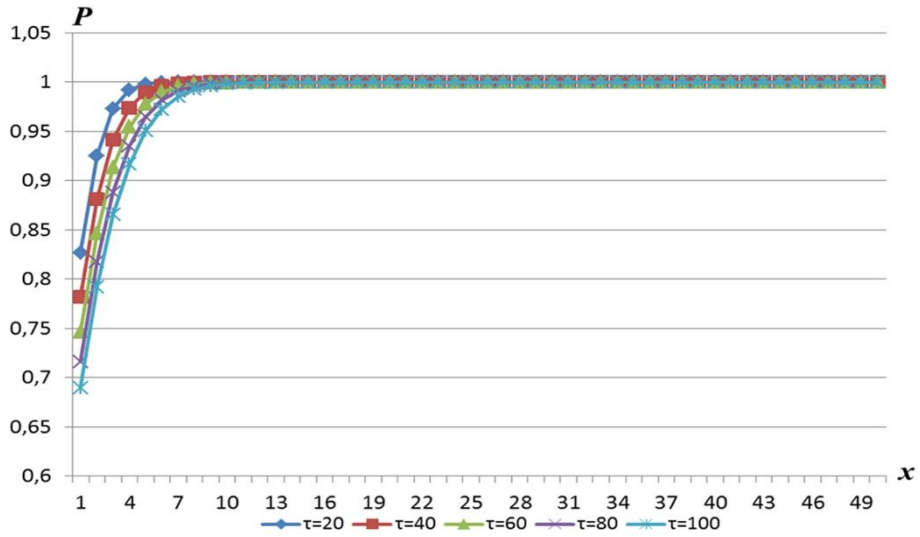


Рис. 2. Динамика перераспределения давления в пласте со временем (время в сутках)

В ходе экспериментов было принято, что система пластов разрабатывается одной галереей, расположенной в начале и в средине координат. Пласт разрабатывается при следующих краевых условиях: на левой границе $K\dfrac{\partial P}{\partial x}\Big|_{x=0} = q_{\text{ж}}$ и на правой — $P\big|_{x=1} = P_{\text{н}}$, т.е. поддерживается начальное пластовое давление.

Как следует из рис. 2, с течением времени работы скважины давление внутри области фильтрации пропорционально снижается за счет отбора флюидов на границе. Такое снижение значения давления зависит от коэффициента фильтрации и пористости среды.

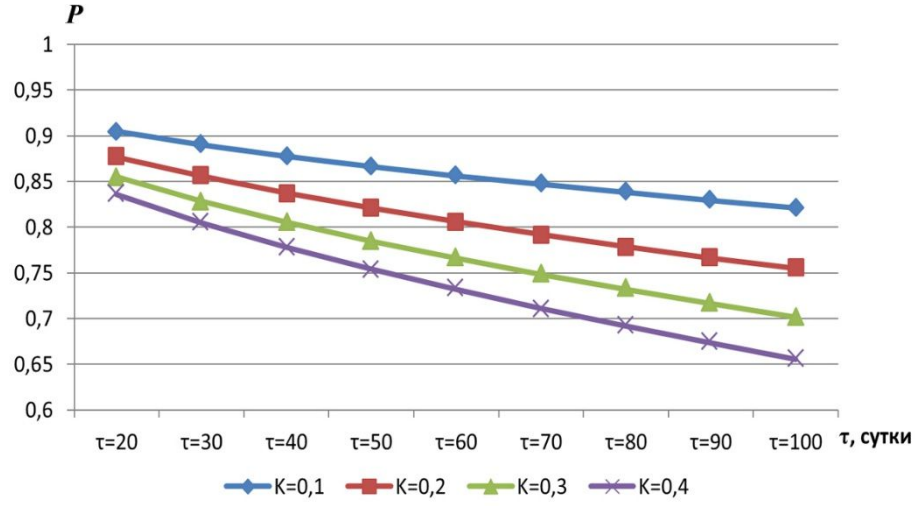


Рис. 3. Динамика перераспределения давления в пласте при различных значениях коэффициента фильтрации

Проведенные численные расчеты показали, что существенными параметрами, влияющими на технологию разработки добычи углеводородов из пластовых систем, являются коэффициент фильтрации и структура пористых пород (рис. 3).

Когда отбор флюидов происходит внутри области фильтрации (рис. 4–5) наблюдается симметричный перепад давления относительно местонахождения скважин.

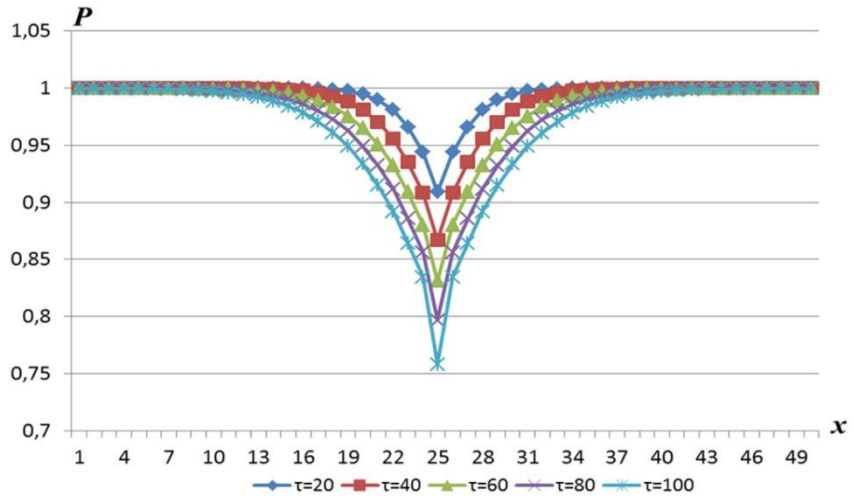


Рис. 4. Перераспределение давления в пласте (в сутках) при работе одной скважин

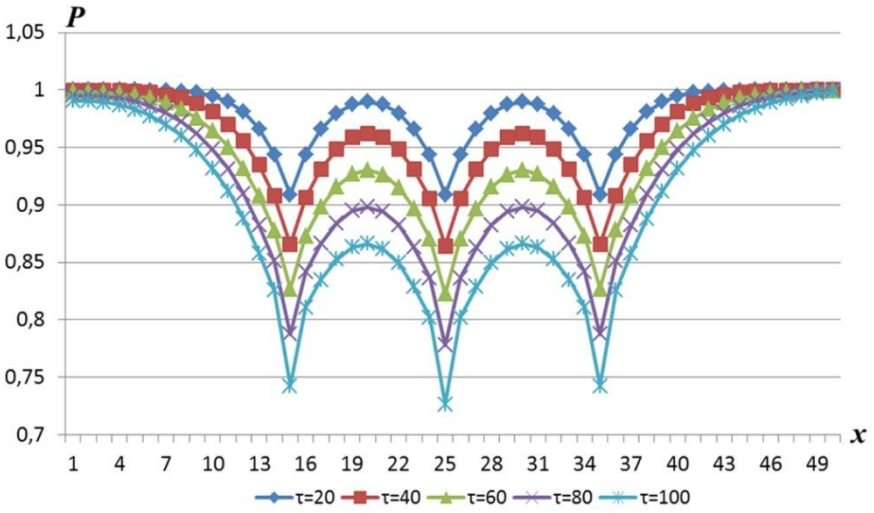


Рис. 5. Перераспределение давления в пласте (в сутках) при работе трех скважин

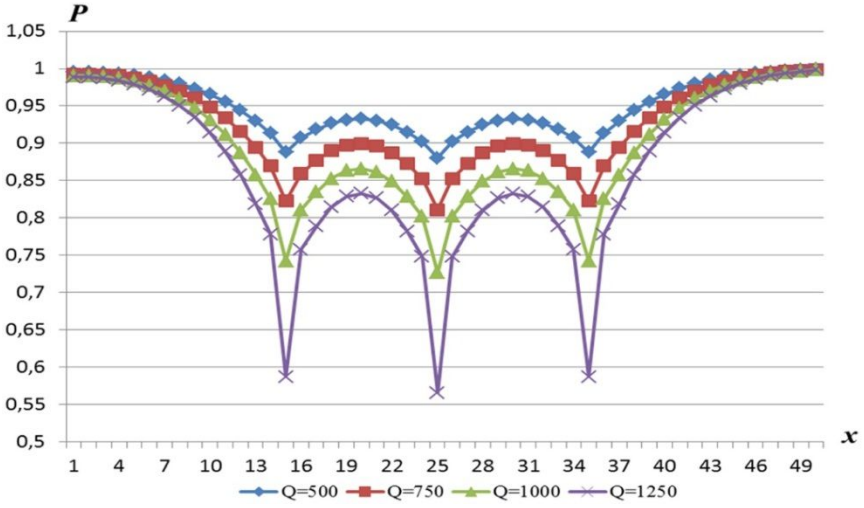


Рис. 6. Перераспределение давления в пласте при различных значениях дебитов в скважинах ($\tau = 100$ сут.)

Вычислительные эксперименты были проведены при различных значениях дебитов скважин. Как следует из кривых на рис. 6, давление в фильтрационной области пропорционально уменьшается в зависимости от роста дебитов скважин.

Анализ полученных результатов проведенных численных экспериментов показывает, что при одинаковой интенсивности добычи падение давления на скважине быстрее при фильтрации нефти, чем при фильтрации с газом, а при рассмотрении в условиях наличия газа в составе нефти увеличивается текучесть смеси. Скорость падения давления на галерее при больших вязкостях нефти всё быстрее по времени, а при небольших вязкостях нефти — сначала быстрее, достигая некоторого небольшого значения, затем начинает падать.

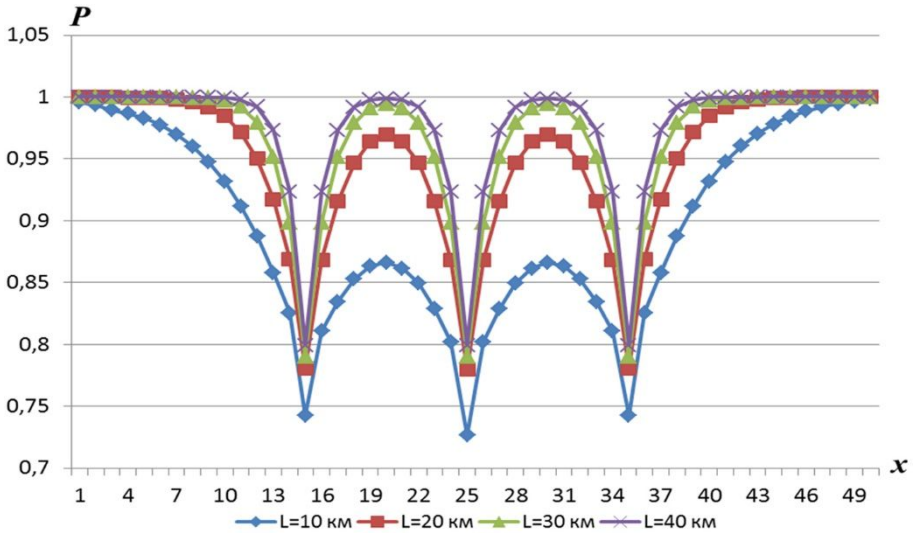


Рис. 7. Перераспределение давления в пласте при различных значениях длины пласта ($\tau = 100$ сут.)

В целом, можно отметить, что динамика перераспределения давления пласта существенно зависит от мощности пласта. С увеличение мощности пласта давление в скважине и соседних точках уменьшается. А время эксплуатации продуктивного пласта (рис. 7) существенно зависит от его длины, мощности, числа скважин и их дебитов.

Заключение

В качестве заключения можно отметить следующее. В настоящей работе были рассмотрены сложные динамические процессы, происходящие в пластовых условиях при двухстороннем вытеснении нефти газом и водой. Были рассмотрены модель и численный алгоритм решения задачи трехфазной фильтрации нефти, газа и воды в пористой среде при воздействии на продуктивный пласт объемом газа и продвижения жидкости. Для обоснования адекватности разработанных модели и алгоритма, согласно гидродинамическим законам взаимосвязных систем, были проведены вычислительные эксперименты на ЭВМ.

Анализ результатов вычислительных экспериментов при широких изменениях фильтрационных параметров для решения различных тестовых задач, показывает адекватность построенных математических моделей, сходимость и устойчивость построенных вычислительных алгоритмов.

Результаты подтверждают пригодность алгоритма и программы для расчетов полей давлений и насыщенностей, а также показателей разработки месторождений в системе нефть – газ, нефть – вода, нефть – газ – вода.

Разработанные математическая модель, вычислительный алгоритм и программное средство могут быть использованы для анализа функционирования, оперативного управления и прогнозирования разработки нефтегазовых месторождений при различных условиях воздействия на пласт и принятия конкретных практических рекомендаций.

Литература

1. Monteiro P.J., Rycroft Ch.H., Barenblatt G.I. A mathematical model of fluid and gas flow in nanoporous media / P.J. Monteiro, Ch.H. Rycroft, G.I. Barenblatt // Proceedings of the National Academy of Sciences of the United States of America. — 2012. — Vol. 109, No. 50. — P. 20309–20313. DOI: 10.1073/pnas.1219009109
2. Barenblatt G.I., Patzek T.W., Silin D.B. The mathematical model of nonequilibrium effects in water-oil displacement / G.I. Barenblatt, T.W. Patzek, D.B. Silin // Society of Petroleum Engineers. — 2003. — Vol. 8, No. 4. — P. 409–416. DOI: 10.2118/87329-pa
3. Chraïbi M., Zaleski S., Franco F. Modeling the solution gas drive process in heavy oils / M. Chraïbi, S. Zaleski, F. Franco // Записки Горного института. — 2008. — № 174. — С. 36–40.
4. Шелухин В.В. Задача капиллярного вытеснения для одной модели трехфазной фильтрации / В.В. Шелухин // Прикладная механика и техническая физика. — 2003. — Т. 44, № 6. — С. 95–106.
5. Atkinson C., Isangulov R. A mathematical model of an oil and gas field development process / C. Atkinson, R. Isangulov // European Journal of Applied Mathematics. — Vol. 21, No. 3. — P. 205–227. DOI: 10.1017/s095679251000001x
6. Ахметзянов А.В., Ибрагимов И.И., Ярошенко Е.А. Интегрированные гидродинамические модели при разработке нефтяных месторождений / А.В. Ахметзянов, И.И. Ибрагимов, Е.А. Ярошенко // Управление большими системами. — 2010. — № 29. — С. 167–183.
7. Ахмед–Заки Д.Ж. Об одной задаче двухфазной фильтрации смеси в пористой среде с учетом теплового воздействия // Научные труды НИПИ Нефтегаз ГНКАР. — 2010. — № 3. — С. 29–33.
8. Давлетбаев А.Я. Фильтрация жидкости в пористой среде со скважинами с вертикальной трещиной гидроразрыва пласта / А.Я. Давлетбаев // Инженерно-физический журнал. — 2012. — Т. 85, № 6. — С. 919–924.
9. Демьянов А.Ю., Динариев О.Ю., Иванов Е.Н. Моделирование переноса воды с мелкодисперсной газовой фазой в пористых средах / А.Ю. Демьянов, О.Ю. Динариев, Е.Н. Иванов // Инженерно-физический журнал. — 2012. — Т. 85, № 6. — С. 1145–1154.
10. Василев Ю.Н. Автоматизированная система управления разработкой газовых месторождений / Ю.Н. Василев. — Москва: Недра, 1987. — 141 с.
11. Шалимов Б.В. О фильтрации трехфазной жидкости (модель Баклеря—Леверетта) / Б.В. Шалимов // Механика жидкости и газа. — 1972. — № 1. — С. 39–44.

12. Ravshanov N., Abilkasimov B., Kurbonov N. The Model and Numerical Algorith, to Research the Filtration processes in porous media taking into account the phase transitions of multicomponent mixtures / N. Ravshanov, B. Abilkasimov, N. Kurbonov // International Multidisciplinary Journal European researcher. — 2012. — No. 1. — P. 5–11.
13. Равшанов Н., Курбонов Н. Моделирование процесса фильтрации трехфазной смеси «нефть-газ-вода» в пористых средах / Н. Равшанов, Н. Курбонов // Технология материалов. — Москва: ИНГН, 2014. — № 3. — С. 3–13.
14. Курбонов Н.М. Алгоритм оптимальной добычи газа из пластовых систем / Н.М. Курбонов // Отраслевые аспекты технических наук. — Москва: ИНГН, 2013. — № 10. — С. 15–19.
15. Равшанов Н., Курбонов Н.М. Моделирование процесса фильтрации жидкостей и газа в пористых средах / Н. Равшанов, Н.М. Курбонов // Информатика: проблемы, методология, технологии : сборник трудов XIV международной конференции. — Воронеж, 2014. — Т. 1. — С. 243–247.

Равшанов Нормахмад, д.т.н., заведующий лабораторией «Моделирование сложных систем» Центра разработки программных продуктов и аппаратно-программных комплексов, Ташкентский университет информационных технологий (Ташкент, Узбекистан), ravshanzade-09@mail.ru.

Курбонов Нозим Мухаммадрашитович, младший научный сотрудник лаборатории «Моделирование сложных систем» Центра разработки программных продуктов и аппаратно-программных комплексов, Ташкентский университет информационных технологий (Ташкент, Узбекистан), nozim_kurbonov@mail.ru.

Поступила в редакцию 13 января 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 2, pp. 89–106

DOI: 10.14529/cmse150207

COMPUTER MODELING OF PROCESSES OF FLUID FILTRATION IN POROUS MEDIA

N. Ravshanov, Tashkent University of Information Technologies (Tashkent, Uzbekistan) ravshanzade-09@mail.ru,

N. Kurbonov, Tashkent University of Information Technologies (Tashkent, Uzbekistan) nozim_kurbonov@mail.ru

To intensify the development of oil and gas fields, to enhance their technical and economic parameters in order to maximize product extraction from old oil and gas deposits, it is necessary to conduct comprehensive research using mathematical tools. As an example of such a mathematical tool there is a triad - "mathematical models, numerical algorithms and software" related to solving of problems and to conduct computational experiments with different input technological

parameters of the object of study. This paper considers mathematical model, conservative numerical algorithms and software for computational experiments based on the basic laws of hydrodynamics. Created mathematical and software serves the purpose of research, forecasting and decision-making on designing and development of oil and gas fields. Also the paper presents the results of computational experiments with unsteady filtration of fluids in porous media.

Keywords: computer modeling, numerical method, computational experiment, porous media, fluid, software.

References

1. Monteiro P.J., Rycroft Ch.H., Barenblatt G.I. A mathematical model of fluid and gas flow in nanoporous media // Proceedings of the National Academy of Sciences of the United States of America. 2012. Vol. 109, No. 50. P. 20309–20313. DOI: 10.1073/pnas.1219009109
2. Barenblatt G.I., Patzek T.W., Silin D.B. The mathematical model of nonequilibrium effects in water–oil displacement // Society of Petroleum Engineers. 2003. Vol. 8, No. 4. P. 409–416. DOI: 10.2118/87329-pa
3. Chraïbi M., Zaleski S., Franco F. Modeling the solution gas drive process in heavy oils // Zapiski Gornogo instituta [Notes of the Mining Institute]. 2008. No. 174. P. 36–40.
4. Sheluhin V.V. Zadacha kapilljarnogo vytesnenija dlja odnoj modeli trehfaznoj fil'tracii prikladnaja mehanika i tehnicheskaja fizika [The problem of capillary displacement for one model of three-phase filtering] // Prikladnaja mehanika i tehnicheskaja fizika [Applied Mechanics and Technical Physics]. 2003. Vol. 44, No. 6. P. 95–106.
5. Atkinson C., Isangulov R. A mathematical model of an oil and gas field development process // European Journal of Applied Mathematics. Vol. 21, No. 3. P. 205–227. DOI: 10.1017/s095679251000001x
6. Ahmetzjanov A.V., Ibragimov I.I., Jaroshenko E.A. Integrirovannye gidrodinamicheskie modeli pri razrabotke neftjanyh mestorozhdenij upravlenie bol'shimi sistemami [Integrated hydrodynamical models of oil field development processes] // Upravlenie bol'shimi sistemami [Large-scale Systems Control]. 2010. No. 29. P. 167–183.
7. Ahmed–Zaki D.Zh. Ob odnoj zadache dvuhfaznoj fil'tracii smesi v poristoj srede s uchetom teplovogo vozdejstvija [About one problem of biphase filtration of mix in porous media with thermal influence] // Nauchnye trudy NIPI Neftegaz GNKAR [Matters of Research Project Institute Neftogaz]. 2010. No. 3. P. 29–33.
8. Davletbaev A.Ja. Fil'tracija zhidkosti v poristoj srede so skvazhinami s vertikal'noj treshhinoj gidrorazryva plasta [Fluid filtration in a porous medium with wells with vertical hydraulic fracture of the formation] // Inzhenerno-fizicheskij zhurnal [Journal of Engineering Physics and Thermophysics]. 2012. Vol. 85, No 6. P. 919–924.
9. Dem'janov A.Ju., Dinariev O.Ju., Ivanov E.N. Modelirovanie perenosa vody s melkodispersnojgazovoj fazoj v poristyh sredah [Simulation of water with finely dispersed gas phase in porous media] // Inzhenerno-fizicheskij zhurnal [Journal of Engineering Physics and Thermophysics]. 2012. Vol. 85, No 6. P. 1145–1154.
10. Vasilev Ju.N. Avtomatizirovannaja sistema upravlenija razrabotkoj gazovyh mestorozhdengij [Automated system for managing the development of gas fields]. 1987. 141 p.
11. Shalimov B.V. O fil'tracii trehfaznoj zhidkosti (model' Baklerja—Leveretta) [About three-phase fluid filtration (model Buckler—Leverett)] // Mehanika zhidkosti i gaza [Fluid and gas mechanics]. 1972. No. 1. P. 39–44.

12. Ravshanov N., Abilkasimov B., Kurbonov N. The Model and Numerical Algorith, to Research the Filtration processes in porous media taking into account the phase transitions of multicomponent mixtures // Journal European researcher. 2012. No. 1. P. 5–11.
13. Ravshanov N., Kurbonov N. Modelirovanie processa fil'tracii trehfaznoj smesi «neft'-gaz-voda» v poristyh sredah [Modeling of process of a filtration of the three-phase mix "oil-gas-water" in porous spheres] // Tehnologija materialov [Technology of materials]. 2014. No. 3. P. 3–13.
14. Kurbonov N.M. Algoritm optimal'noj dobychi gaza iz plastovyh sistem [The algorithm for optimal gas production from stratal systems] // Otraslevye aspekty tehnicheskih nauk [Sectoral aspects of technical sciences]. 2013. No. 10. P. 15–19.
15. Ravshanov N., Kurbonov N.M. Modelirovanie processa fil'tracii zhidkostej i gaza v poristyh sredah [Modelling of the process liquid and gas filtration in porous media] // Informatika: problemy, metodologija, tehnologii : sbornik trudov XIV mezhdunarodnoj konferencii [Proceedings of the conference Computer science: problems, methodology, technology]. 2014. Vol. 1. P. 243–247.

Received January 13, 2015.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ», серия «Вычислительная математика и информатика» основан в 2012 году.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год (февраль, май, август и ноябрь).

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `http://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются.**
2. Адрес редакции научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, факультет ВМИ, кафедра ИТ, ответственному секретарю Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**