УДК 004.382.2-027.21

# Масштабируемый инструментарий параллельного мультипрограммирования пространственно-распределённых вычислительных систем[1]

В.Г. Хорошевский, М.Г. Курносов, С.Н. Мамойленко, К.В. Павский, А.В. Ефимов, А.А. Пазников, Е.Н. Перышкова

В работе представлен масштабируемый инструментарий (модели, алгоритмы и программное обеспечение) организации мультипрограммного функционирования распределенных вычислительных систем (ВС). Основное внимание уделено подсистемам диспетчеризации и формирования расписаний решения параллельных задач на пространственно-распределённых ВС. Представлены результаты моделирования работы алгоритмов этих подсистем на ресурсах пространственно-распределённой мультикластерной ВС.

*Ключевые слова*: параллельное мультипрограммирование, управление ресурсами, диспетчеризация задач, составление расписаний, распределённые вычислительные системы.

## 1. Введение

Технологии и программное обеспечение распределённых и высокопроизводительных вычислительных систем (ВС) относят к критически важным для Российской Федерации [1]. Широкое применение получили пространственно-распределённые вычислительные системы – макроколлективы рассредоточенных вычислительных средств (подсистем), взаимодействующих между собой через локальные и/или глобальные сети связи (включая сеть Internet) [2, 3]. К таким системам относятся GRID-системы и мультикластерные ВС.

Актуальной является проблема оптимизации использования ресурсов пространственно-распределённых ВС при решении на них параллельных задач. Известно, что повышение эффективности использования ресурсов ВС связано с использованием технологии параллельного мультипрограммирования [2, 3], – ресурсы ВС распределяются между несколькими одновременно решаемыми задачами.

К значимым проблемам организации функционирования пространственно-распределённых ВС в мультизадачных режимах относятся диспетчеризация задач (определение, на какой подсистеме пространственно-распределённой ВС будет решаться задача) и формирование расписаний их решения на выделенных ресурсах (подсистемах).

Большинство используемых на практике средств диспетчеризации задач в пространственно-распределённых ВС — централизованные. Такие средства имеют существенный недостаток – отказ управляющего узла ВС может привести к неработоспособности всей системы. Кроме того, в случае применения таких средств в большемасштабных ВС, возрастают

[1] Работа выполнена при поддержке РФФИ (гранты № 09-07-00095, 11-07-00109, 11-07-00105), Совета по грантам Президента РФ для поддержки ведущих научных школ (грант НШ-5176.2010.9) и в рамках государственного контракта № 07.514.11.4015 с Минобрнауки РФ.

временны́е затраты на поиск требуемых ресурсов. Поэтому необходимо разрабатывать децентрализованные модели, алгоритмы и программное обеспечение для диспетчеризации параллельных задач в пространственно-распределённых ВС.

Повысить эффективность эксплуатации ресурсов подсистем пространственно-распределённых ВС и снизить время нахождения задач в локальных очередях, в частности, возможно, если задачи будут допускать решение на различных конфигурациях вычислительных ресурсов (в рамках одной подсистемы). В этом случае задачи называют масштабируемыми или «пластичными» (moldable). Исследования пользовательских запросов показывают, что свойством масштабируемости обладают более 80 % задач [4].

Системы пакетной обработки, используемые на практике, допускают наличие в очередях масштабируемых задач и позволяют пользователям указывать количество требуемых для их решения элементарных машин (ЭМ) в виде диапазонов допустимых значений. При этом алгоритмы планирования свойство масштабируемости зачастую не принимают во внимание. Поэтому необходимо разрабатывать методы, алгоритмы и программные средства формирования расписаний решения масштабируемых задач на распределённых ВС.

В данной работе представлены оригинальные алгоритмы и программы децентрализованной диспетчеризации параллельных задач (раздел 2) и составления расписаний решения масштабируемых задач на распределённых ВС (раздел 3). В разделе 4 описаны программные компоненты подсистемы параллельного мультипрограммирования пространственно-распределённой мультькикластерной ВС Центра параллельных вычислительных технологий (ЦПВТ) Федерального государственного образовательного бюджетного учреждения высшего профессионального образования «Сибирский государственный университет телекоммуникаций и информатики» (ФГОБУ ВПО «СибГУТИ») и Лабораторией вычислительных систем Института физики полупроводников им. А.В. Ржанова Сибирского отделения РАН (ИФП СО РАН). Результаты моделирования работы предложенных алгоритмов и программного обеспечения представлены в разделе 5.

## 2. Диспетчеризация задач в пространственно-распределённых вычислительных системах

### 2.1. Постановка задачи

Пусть имеется пространственно-распределённая ВС, состоящая из $H$ подсистем; $N$ – суммарное количество элементарных машин (ЭМ) в подсистемах. Под ЭМ понимается единица вычислительного ресурса, предназначенного для выполнения ветви параллельной программы (например, процессорное ядро). Введём обозначения:

– $n_i$ – количество ЭМ, входящих в состав подсистемы $i \in S = \{1, 2, \ldots, H\}$;
– $c_i$ – число свободных ЭМ в подсистеме $i$;
– $q_i$ – число задач в очереди подсистемы $i$;
– $s_i$ – число задач, выполняющихся на ЭМ подсистемы $i$;
– $t_{ij} = t(i, j, m)$ – время передачи (в секундах) сообщения размером $m$ байт между подсистемами $i, j \in S$.

Считается, что все подсистемы объединены сетью связи.

На каждой подсистеме присутствует локальная система управления ресурсами (СУР) и децентрализованный диспетчер. Коллектив диспетчеров представлен в виде ориентированного графа $G(S, E)$, в котором вершинам соответствуют диспетчеры, а рёбрам – логические связи между ними (рис. 1). Наличие дуги $(i, j) \in E$ в графе означает, что диспетчер $i$ может отправлять задачи диспетчеру $j$. Множество вершин $j$, смежных вершине $i$, образуют её локальную окрестность $L(i) = \{j \in S \mid (i, j) \in E\}$.

Пользователь направляет задачу диспетчеру $i$. Задача содержит программу, входные файлы и ресурсный запрос, в котором указываются ранг $r$ программы (количество параллельных ветвей), размеры $z_1, z_2, \ldots, z_k$ исполняемого и входных файлов программы

([$z_l$] = байт), а также номера $h_1$, $h_2$, …, $h_k$ подсистем, на которых размещены соответствующие файлы ($h_l \in S$). Диспетчер (в соответствии с реализованным в нём алгоритмом) выполняет поиск (суб)оптимальной подсистемы $j^* \in L(i) \cup \{i\}$ (или подсистем $j_1^*, j_2^*, \ldots, j_m^*$) из его локальной окрестности.

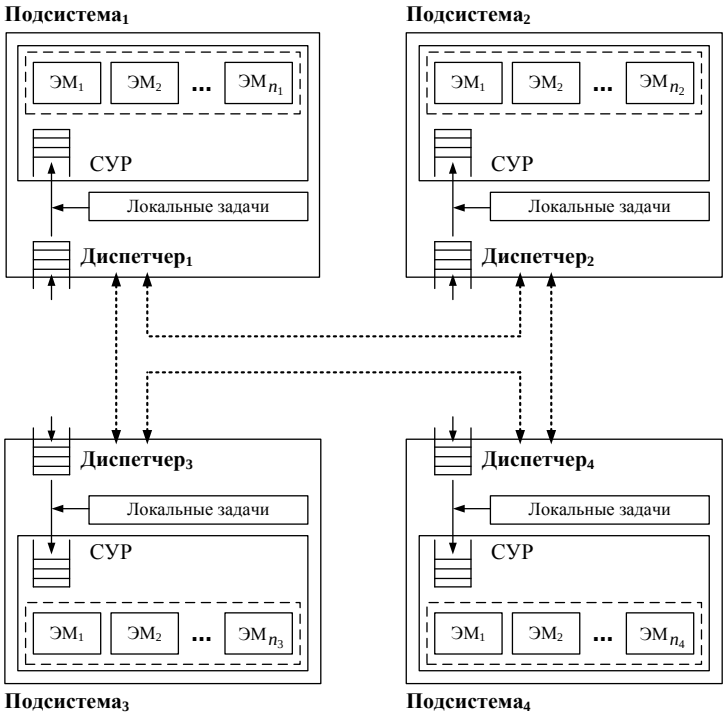



Рис. 1. Пример локальных окрестностей диспетчеров:
$H = 4$, $L(1) = \{2, 3\}$, $L(2) = \{1, 4\}$, $L(3) = \{1, 4\}$, $L(4) = \{2, 3\}$

Предложено четыре децентрализованных алгоритма диспетчеризации параллельных задач в пространственно-распределённых ВС (локально-оптимальный (ДЛО), на основе репликации (ДР), на основе миграции задач (ДМ), на основе репликации и миграции задач (ДРМ)). Каждый алгоритм описывает функционирование диспетчера $i$ при поступлении задачи в его очередь. На начальном этапе работы все алгоритмы предполагают обращение диспетчера $i$ к системе мониторинга ресурсов ВС и получение текущих значений параметров $t_{ij}$, $c_j$, $s_j$, $q_j$ и $n_j$ ($j \in L(i) \cup \{i\}$). После чего строится множество допустимых подсистем $S(i) = \{j \mid n_j \geq r, j \in L(i) \cup \{i\}\}$, имеющих количество ЭМ не ниже требуемого.

### 2.2. Алгоритм локально-оптимальной диспетчеризации

Алгоритм осуществляет выбор локально-оптимальной подсистемы.

*Шаг 1.* Из окрестности $S(i)$ диспетчера $i$ выбирается подсистема $j^*$ с минимальным значением $F(j)$, $j \in S(i)$.

$$j^* = \arg\min_{j \in S(i)} F(j), \qquad F(j) = \begin{cases} \dfrac{t_j}{t_{\max}} + \dfrac{c_{\max}}{c_j} + \dfrac{w_j}{w_{\max}}, & \text{если } c_j < r \text{ или } q_j > 0, \\ \dfrac{t_j}{t_{\max}}, & \text{иначе,} \end{cases}$$

где

– $t_j = \sum_{l=1}^{k} t(h_l, j, z_l)$ – время доставки файлов задачи до подсистемы $j$;

– $t_{\max} = \max_{j \in S(i)} \{t_j\}$;
– $c_{\max} = \max_{j \in S(i)} \{c_j\}$;
– $w_j = q_j / n_j$ – количество задач в очереди, приходящееся на одну ЭМ подсистемы $j$;
– $w_{\max} = \max_{j \in S(i)} \{w_j\}$.

*Шаг 2.* Задача направляется в очередь локальной СУР подсистемы $j^*$, после чего осуществляется доставка файлов задачи на эту подсистему.

Ранжирование подсистем по значению функции $F(j)$ позволяет учесть время доставки файлов задачи до подсистем, а также их относительную загруженность.

### 2.3. Алгоритм диспетчеризации на основе репликации задач

*Шаг 1.* Из окрестности $S(i)$ выбирается $m$ подсистем $j_1^*, j_2^*, \ldots, j_m^*$ в порядке неубывания значений функции $F(j)$.

*Шаг 2.* Задача одновременно направляется в очереди локальных СУР подсистем $j_1^*, j_2^*, \ldots, j_m^*$, после чего осуществляется доставка файлов задачи до этих подсистем.

*Шаг 3.* С интервалом времени $\Delta$ диспетчер $i$ проверяет состояние задачи на подсистемах $j_1^*, j_2^*, \ldots, j_m^*$ и определяет подсистему $j'$, на которой задача запущена на выполнение раньше других подсистем.

*Шаг 4.* Задача удаляется из очередей локальных СУР подсистем, отличных от $j'$.

Периодический поиск подсистем для задач в очереди позволяет учесть динамически изменяющуюся загрузку ресурсов пространственно-распределённых вычислительных и GRID-систем.

### 2.4. Алгоритм диспетчеризации на основе миграции задач

Шаг 1. Из окрестности $S(i)$ диспетчера $i$ выбирается подсистема $j^*$ с минимальным значением $F(j)$, $j \in S(i)$.

Шаг 2. Задача направляется в очередь локальной СУР подсистемы $j^*$, после чего осуществляется доставка файлов задачи на эту подсистему.

Шаг 3. Диспетчер $i$ с интервалом времени $\Delta$ запускает процедуру ДЛО поиска подсистемы $j'$ для задачи в очереди диспетчера $j^*$.

Шаг 4. Если для найденной подсистемы выполняется условие $F(j^*) - F(j') > \varepsilon$, то выполняется миграция задачи в очередь подсистемы $j'$ (задача удаляется из очереди диспетчера $j^*$).

### 2.4. Алгоритм диспетчеризации на основе репликации и миграции задач

Алгоритм основан на комбинации двух подходов – назначения на несколько подсистем и миграции из очереди.

Важно отметить, что вычислительная сложность поиска подсистем предложенными алгоритмами не зависит от количества $H$ подсистем, так как поиск осуществляется только в пределах локальных окрестностей диспетчеров. Это обеспечивает применимость алгоритмов в большемасштабных пространственно-распределённых ВС.

# 3. Формирование расписаний решения масштабируемых задач на распределённых вычислительных системах

## 3.1. Постановка задачи

Имеются распределённая ВС, состоящая из $N$ элементарных машин, и набор из $L$ масштабируемых задач. Каждая задача $i \in \{1, 2, \ldots, L\}$ характеризуется вектором $p_i = (p_i^1, p_i^2, ..., p_i^{q_i})$ и величиной штрафа $c_i$ за задержку её решения в единицу времени, $c_i > 0$. Элементы вектора $p_i^k = (r_i^k, t_i^k, w_i^k)$, $k \in \{1, 2, ..., q_i\}$ задают допустимые значения ранга $r_i^k$ задачи и времени $t_i^k$ её решения и определяют приоритет $w_i^k$ выбора этих значений ($0 < r_i^k \le N$, $t_i^k > 0$, $w_i^k > 0$).

«Удовлетворённость» пользователя выбором для решения задачи значений с приоритетом $w_i^k$ оценивается как $w_i^k \Big/ \max\limits_{k=1,q_i} w_i^k$. Допускается существование в векторе $p_i$ нескольких элементов с одинаковым приоритетом. Считается, что в наборе присутствуют задачи с различным количеством $q_i$ допустимых значений параметров и возможно взаимодействие ЭМ с любой другой машиной ВС. Зависимости между величинами $r_i^k$, $t_i^k$, $w_i^k$, $c_i$ отсутствуют.

Необходимо для каждой задачи $i$ выбрать: $k_i$ – номер элемента вектора $P_i$, время $s_i$ начала решения задачи, а также выделить подсистему $J_i = j$ элементарных машин ВС, где $0 < k_i \le q_i$, $s_i \ge 0$, $j \in \{1, 2, ..., N\}$. В результате должен быть сформирован вектор $S = \langle (k_1, s_1, J_1), (k_2, s_2, J_2), ..., (k_L, s_L, J_L) \rangle$, называемый расписанием решения задач на ВС. Характеристиками расписания являются: время $T(S)$ решения всех задач набора и суммарный штраф $F(S)$ за задержку их решения. Кроме этого, качество сформированного расписания оценивается степенью загрузки ресурсов ВС при решении задач.

Требуется найти расписание $S^*$ решения задач на вычислительной системе такое, чтобы суммарное время $T(S^*)$ решения задач и штраф $F(S^*)$ за задержку их решения были минимальными:

$$\min_{S \in \Omega} T(S), \quad T(S) = \max_{i=1,L} \{s_i + t_i^{k_i}\}, \tag{1}$$

$$\min_{S \in \Omega} F(S), \quad F(S) = \sum_{i=1}^{L} s_i c_i, \tag{2}$$

при ограничениях:

$$\forall i \in \{1, 2, ..., L\}, |J_i| = r_i^k \text{ и } \forall j_1 \in J_i, \forall j_2 \in J_i \setminus \{j_1\} \Rightarrow j_1 \ne j_2, \tag{3}$$

$$\forall t \ge 0, \bigcap_{i \in \Xi(t)} J_i = 0, \sum_{i \in \Xi(t)} r_i^{k_i} \le N, \tag{4}$$

$$L^{-1} \sum_{i=1}^{L} \frac{w_i^{k_i}}{\max\limits_{k=1,q_i} w_i^k} \ge e, \tag{5}$$

где $\Omega$ – область допустимых расписаний $S$, $\Xi(t) = \{i \in \{1, 2, ..., N\} \mid s_i \le t \le s_i + t_i^{k_i}\}$ – множество номеров задач, решаемых в момент времени $t$; $e$ – минимально допустимая средняя «удовлетворённость» пользователей. Ограничение (3) определяет, что каждой задаче с учётом выбранных значений параметров должно быть выделено столько ЭМ, сколько требуется для её

решения. Ограничение (4) гарантирует, что в каждый момент времени задачи решаются на непересекающихся подсистемах ЭМ и их суммарный ранг не превосходит количества элементарных машин ВС.

Задача (1) – (5) относится к многокритериальной оптимизации [5]. Для решения задачи используем метод приоритетов [5], задавая функции (1) приоритет выше, чем функции (2).

### 3.2. Формирование укрупнённых задач

На первом этапе задачи набора разбиваются на подмножества [6] таким образом, чтобы выполнялись ограничения (3) – (5) и достигался субминимум функции (1). Подмножества задач будем называть укрупнёнными задачами.

Формально постановка задачи первого этапа приобретает следующий вид. Решением является тройка $S_1 = (K, M, \Im)$, где

– $K = (k_1, k_2, ..., k_L)$ – вектор выбранных для задач значений $k_i$;

– $\Im = \{\Im_m\}$ – множество укрупнённых задач $\Im_m \subseteq \{1, 2, ..., L\}$ $m = \overline{1, M}$;

– $M = |\Im|$ – количество укрупнённых задач.

Требуется найти решение $S_1^*$, доставляющее минимум функции $T'$:

$$\min_{S_1 \in \Omega_1} T'(S_1),\ T'(S_1) = \sum_{m=1}^{M} T_m,\ T_m = \max_{i \in \Im_m} t_i^{k_i}, \tag{6}$$

при ограничениях:

$$\bigcup_{m=1}^{M} \Im_m = \Im,\ \bigcap_{m=1}^{M} \Im_m = \varnothing,\ \sum_{m=1}^{M} |\Im_m| = L, \tag{7}$$

$$R_m \le N,\ R_m = \sum_{i \in \Im_m} r_i^{k_i}, \tag{8}$$

где $\Omega_1$ – область допустимых решений $S_1$, $R_m$ – суммарный ранг задач каждого подмножества $\Im_m$, Ограничение (7) требует, чтобы все задачи набора были распределены по укрупнённым задачам. Ограничение (8) задает требование, чтобы суммарный ранг укрупнённых задач не превосходил количество ЭМ вычислительной системы.

Задача (6) – (8) относится к задачам ортогональной упаковки объектов в контейнеры [7]. При этом задача набора кодируется прямоугольником с шириной, равной $r_i^k$, и высотой $t_i^k$. Повороты и пересечения прямоугольников не допускаются. Известно, что эта задача является NP-сложной [8].

### 3.3. Генетический алгоритм формирования укрупнённых задач

Генетический алгоритм предусматривает последовательность жизненных циклов популяции. Каждый цикл состоит из следующих операций: выбора пар особей, их размножения, мутации и селекции наиболее приспособленных особей (формирования новой популяции). Процесс повторяется до тех пор, пока на протяжении заданного количества популяций не будет появляться особь с лучшим значением функции приспособленности. Итоговое разбиение задач набора формируется из особи, имеющей наилучшее значение функции приспособленности.

В терминах генетических алгоритмов будем считать:

– *ген* – укрупнённая задача (пакет);

– *особь или хромосома* – допустимое разбиение задач набора при фиксированных значениях $k_i$;

– *популяция* – несколько особей с различными значениями $k_i$;

– *функция приспособленности особи* – значение функции (6) для соответствующего разбиения задач набора.

Размер популяции задаётся параметром алгоритма и остаётся постоянным в процессе всей эволюции.

### 3.3.1. Последовательная версия алгоритма

Начальная популяция формируются следующим образом. Первая особь выбирает для каждой задачи такое $k_i$, при котором значения параметров имеют наивысший приоритет $w_i^k$. Вторая особь – минимум запрашиваемых ресурсов $r_i t_i$. Третья – максимум «удельной площади» $w_i^k / r_i^k t_i^k$. Остальные особи выбирают $k_i$ случайным образом. Далее каждая особь формирует укрупнённые задачи (гены), используя алгоритм FFDH [9].

На каждом жизненном цикле из популяции выбираются пары особей. Количество пар задается как половина от количества особей в популяции. Для формирования пары из популяции случайным образом выбирается одна особь. Парой для неё выбирается особь, наиболее удалённая от неё по значению функции приспособленности и не состоящая в других парах. Далее над каждой выбранной парой с заданной вероятностью либо выполняется оператор кроссинговера, либо производится мутация родительских особей.

В качестве оператора кроссинговера предлагается алгоритм, в основу которого положен метод перетасовки генов [10]:

*Шаг 1.* Гены родительских особей помещаются в общий список и сортируются по степени загрузки ($(R_m T_m)^{-1} \sum_{i \in \Im_m} r_i^k t_i^k$).

*Шаг 2.* Из полученной последовательности выбираются гены, множества задач в которых не пересекаются. Выбранные гены помещаются в новую особь. Оставшиеся гены расформировываются.

*Шаг 3.* Для расформированных задач случайным образом выбирается один из параметров. Далее эти задачи распределяются по генам с использованием алгоритма FFDH (с учётом уже имеющихся генов).

Мутация выполняется над одной или двумя родительскими особями. Оператор мутации с равной вероятностью либо случайным образом изменяет значения параметров задач, гарантируя выполнение ограничения (5), либо выбирает, для заданной доли задач особи, значения параметров, имеющие максимальный приоритет.

В результате от каждой пары создаются одна или две изменённые особи «потомков», которые могут не выжить, если для них по каким-либо причинам не выполняются ограничения (3) – (5). В новую популяцию попадают родительские и «выжившие потомки», имеющие наилучшие показатели функции приспособленности.

### 3.3.2. Параллельная версия алгоритма

Очевидно, что время работы последовательной версии генетического алгоритма зависит от количества задач в наборе. Для больших наборов множество задач можно разбить на части, для каждой из которых независимо выполнить последовательную версию генетического алгоритма. Термин «большой набор» следует определять экспериментальным путём для конкретной реализации последовательной версии и аппаратурных возможностей вычислителей, на которых этот алгоритм будет выполняться.

## 3.4. Определение последовательности решения укрупнённых задач

На втором этапе определяется последовательность решения укрупнённых задач (пакетов), позволяющая получить минимум функции (2). Состав укрупнённых задач не изменяется.

Пусть $S_2 = m_1, ..., m_l, ..., m_M$ – некоторая последовательность решения пакетов, где $m_l$ – номер пакета, который будет решаться в первую очередь. Штраф за решение задач при выбранной последовательности решения пакетов определяется как

$$F'(S_2) = \sum_{l=2}^{M} C_{ml} \sum_{r=1}^{l-1} T_{m_r}, \quad (9)$$

где $C_{ml} = \sum_{i \in \Im_m} c_i$ – штраф за задержку решения задач, входящих в $m$-й пакет.

Требуется найти такую последовательность $S_2^*$, чтобы достичь минимума функции (9).

Запишем функцию (10) в следующем виде

$$F'(S_2) = C*T - f(S_2),$$

где

$$C = \sum_{l=1}^{M} C_{ml}, \; T = \sum_{l=1}^{M} T_{ml},$$

$$f(S_2) = \sum_{l=1}^{M} T_{ml} \sum_{r=1}^{l} C_{m_r}. \quad (10)$$

Очевидно, что минимум (9) достигается при максимуме (10).

Известно утверждение [6, 11], что для того чтобы последовательность решения пакетов задач обеспечивала минимум функции (10), необходимо и достаточно, чтобы выполнялись неравенства

$$\frac{T_{m_1}}{C_{m_1}} \le \frac{T_{m_2}}{C_{m_2}} \le ... \le \frac{T_{m_l}}{C_{m_l}} \le ... \le \frac{T_{m_M}}{C_{m_M}}. \quad (11)$$

### 3.5. Формирование итогового расписания решения задач

Итоговое расписание $S^* = \langle (k_i, s_i, J_i) \rangle$ формируется исходя из $S_1^*$ и $S_2^*$ следующим образом. Для $k_i$ используются значения из $S_1^*$. Время начала решения $s_i$ определяется временем начала решения укрупнённой задачи $s_r^*$, в которую она была помещена. Время $s_r^*$ определяется найденной последовательностью решения укрупнённых задач: $s_r^* = \sum_{l=1}^{r-1} T_{m_l}, s_1^* = 0$. Итак, $\forall m \in \{1, 2, ..., M\}$, $\forall i \in \Im_m, s_i = s_m^*$. Машины ВС распределяются между задачами каждой укрупнённой задачи в соответствии с требованиями.

## 4. Масштабируемый инструментарий параллельного мультипрограммирования распределённых вычислительных систем

Инструментарий параллельного мультипрограммирования (рис. 2) входит в состав системного программного обеспечения пространственно-распределённой мультикластерной ВС. Инструментарий – это модели, методы и программное обеспечение организации функционирования распределённых ВС при решении множества задач, представленных параллельными программами.

Стандартные компоненты системного программного обеспечения представлены:

- сетевой операционной системой GNU/Linux (дистрибутив CentOS 5.5, версия ядра 2.6.18);
- средствами разработки, отладки и анализа последовательных и параллельных программ:
  - компиляторы: GCC, Oracle, Intel;

- математические библиотеки: GNU Scientic Library, AMD Core Math Library, Intel Math Kernel Library;
- библиотеки передачи сообщений между ветвями параллельных программ:
  - распределённые приложения – MPI: MPICH2, OpenMPI, IntelMPI;
  - параллельные программы – OpenMP: gcc, sun, intel;
  - средствами отладки и анализа программ: gdb, vampire, gprof и т.д.

– программным обеспечением организации взаимодействия пространственно-распределённых кластерных ВС и диспетчеризации пользовательских заданий: Globus Toolkit, GridWay.

Инструментарий параллельного мультипрограммирования включает:

– средства самоконтроля и самодиагностики;
– подсистему организации функционирования ВС в мультипрограммных режимах, включающую средства вложения параллельных программ и реализации эффективных групповых обменов между ветвями параллельных программ;
– средства организации распределённой очереди задач и диспетчер пользовательских запросов;
– подсистему анализа параллельных программ и алгоритмов планирования ресурсов распределённых ВС;
– средства мониторинга и организации удалённого доступа к ресурсам ВС.

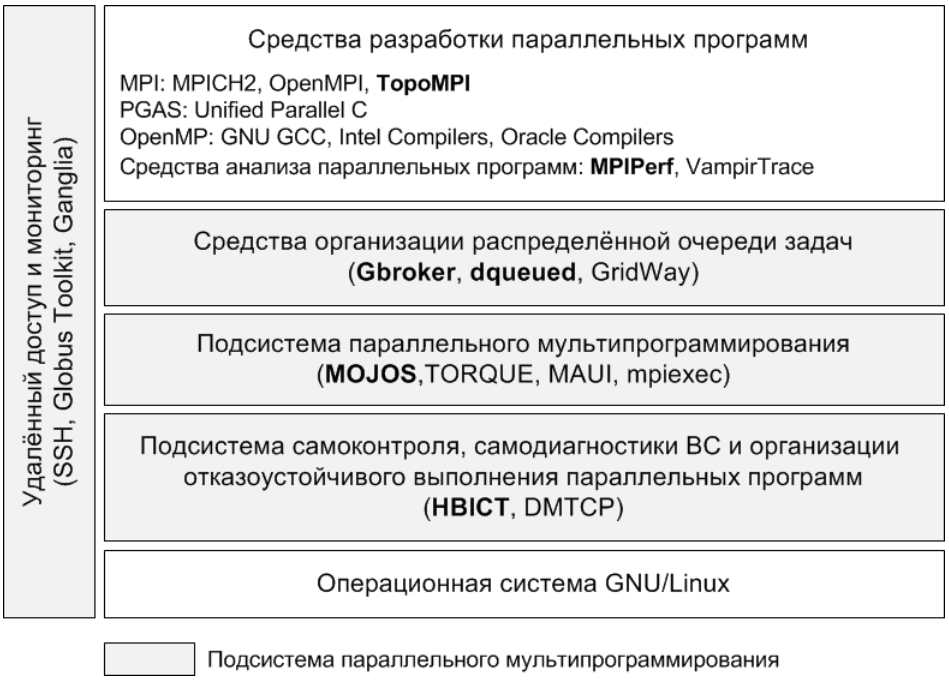



Рис. 2. Программное обеспечение пространственно-распределённой мультикластерной вычислительной системы

### 4.1. Программный пакет GBroker

Программный пакет GBroker [12] обеспечивает децентрализованную диспетчеризацию параллельных MPI-программ в пространственно-распределённых ВС. Пакет разработан на языке ANSI C для операционной системы GNU/Linux. Пакет разрабатывается ЦПВТ ФГОБУ ВПО «СибГУТИ» совместно с Лабораторией вычислительных систем ИФП СО РАН. Он включает в себя диспетчер GBroker, модуль интерфейса GClient и системы мониторинга NetMon и DCSMon. Модуль GBroker реализует алгоритмы децентрализованной диспетчеризации задач, взаимодействуя с локальной СУР через подсистему GRAM пакета Globus Toolkit. DCSMon отвечает за информацию о вычислительных ресурсах подсистем локальной

окрестности диспетчера. NetMon формирует сведения о производительности каналов связи межу подсистемами.

Все модули устанавливаются на подсистемах; администратор задаёт локальные окрестности диспетчеров и систем мониторинга. Задача состоит из параллельной программы и описания на языке Job Submission Description Language (JSDL). Пользователь может отправить задачу (командой gclient) любому из диспетчеров GBroker распределённой ВС. Передача файлов внутри системы осуществляется средствами GridFTP.

### 4.2. Программный пакет MOJOS

Программный пакет MOJOS (от англ. MOldable JOb Scheduling) предназначен для моделирования и отладки алгоритмов формирования расписаний решения масштабируемых задач на распределённых ВС. Пакет разработан на языке ANSI C для операционной системы GNU/Linux. В состав пакета входят средства:

- генерирования задач. Задачи генерируются на основе известных статистик поступления задач в распределённые вычислительные системы [13] или с использованием аналитических моделей потоков задач. Масштабирование задач производится с использованием модели Downey [14]. Приоритеты запросов задаются случайным образом (равновероятно);
- формирования расписаний решения масштабируемых задач. В пакете реализованы алгоритмы формирования расписаний решения задач, основанные на методах упаковки прямоугольных объектов в контейнеры и полубесконечную полосу. Алгоритмы учитывают свойство масштабируемости задач и приоритетность выбора конфигураций вычислительных ресурсов для решения задач;
- анализа сформированных расписаний решения задач. Используя эти средства, пользователь имеет возможность сравнить несколько сформированных расписаний и проанализировать работу соответствующих алгоритмов.

Дополнительно в пакете MOJOS предусмотрен ряд средств наглядного воспроизведения сформированных расписаний решения задач.

## 5. Моделирование

Созданный инструментарий параллельного мультипрограммирования исследован на пространственно-распределённой мультикластерной ВС, созданной ЦПВТ ФГОБУ ВПО «СибГУТИ» совместно с Лабораторией ВС ИФП СО РАН [15].

### 5.1. Моделирование алгоритмов диспетчеризации задач

Исследование алгоритмов диспетчеризации проводилось на пространственно-распределённой мультикластерной ВС (рис. 3). На каждом сегменте установлена операционная система GNU/Linux, локальная система управления ресурсами TORQUE 2.3.7, пакет Globus Toolkit 5.0 и компоненты пакета GBroker. На сегменте 5 (Xeon80) настроен диспетчер GridWay 5.6.1 для управления ресурсами всех подсистем.

В качестве тестовых задач использовались MPI-программы из пакета тестов SPEC MPI 2007:

- Weather Research and Forecasting (WRF) – пакет моделирования климатических процессов;
- The Parallel Ocean Program (POP2) – пакет моделирования процессов в океане;
- LAMMPS – пакет решения задач молекулярной динамики;
- RAxML – пакет моделирования задач биоинформатики;
- Tachyon – пакет расчёта графических сцен.

Входные данные для тестовых задач размещались на сегменте Xeon80; на эту же подсистему доставлялись результаты выполнения программ.

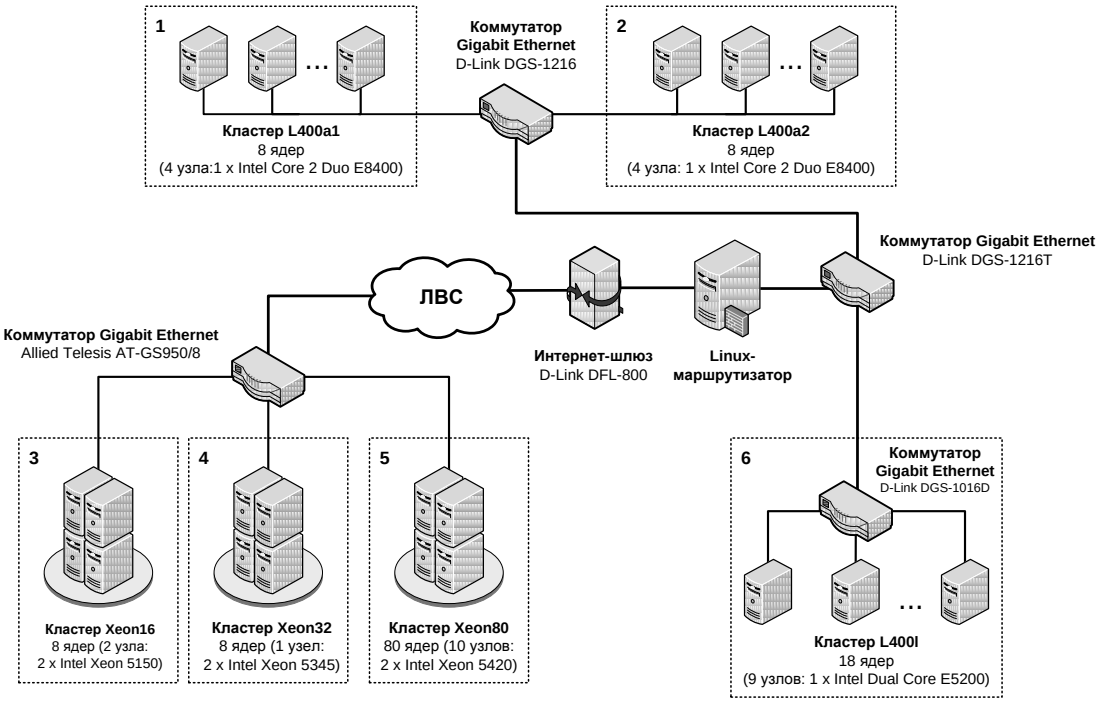



Рис. 3. Тестовая конфигурация мультикластерной ВС ($H = 6$, $N = 130$)

Генерировались простейшие потоки с различной интенсивностью $\lambda$ поступления задач, которые псевдослучайно с равномерным распределением выбирались из тестового набора. Каждый поток формировался из $M$ задач. Ранг $r$ каждой задачи выбирался из множества $\{1, 2, 4, 8\}$ псевдослучайно с равномерным распределением.

Обозначим через $t_k$ время поступления задачи $k \in \{1, 2, \ldots, M\}$ на вход диспетчера, $t'_k$ – время начала решения задачи $k$, $t''_k$ – время завершения решения задачи $k$. Пусть $\tau$ – суммарное время обслуживания потока из $M$ задач.

Для оценки эффективности алгоритмов диспетчеризации использовались следующие показатели: пропускная способность $B$ системы, среднее время $T$ обслуживания задачи и среднее время $W$ пребывания задачи в очереди:

$$B = \frac{M}{\tau}, \quad T = \frac{1}{M}\sum_{k=1}^{M}(t''_k - t_k), \quad W = \frac{1}{M}\sum_{k=1}^{M}(t'_k - t_k).$$

На рис. 4 и 5 приведены результаты сравнения эффективности алгоритмов ДЛО, ДР, ДМ и ДРМ при обслуживании потока из $M = 200$ задач. Поток задач поступал на подсистему Xeon80, локальные окрестности диспетчеров имели структуру полного графа.

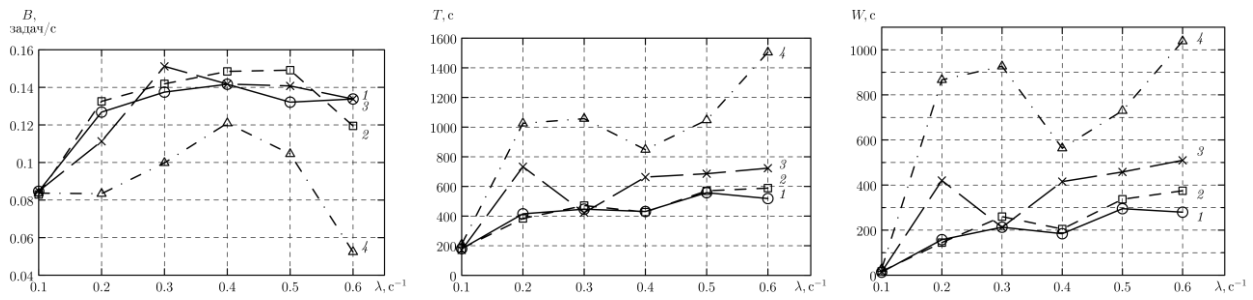
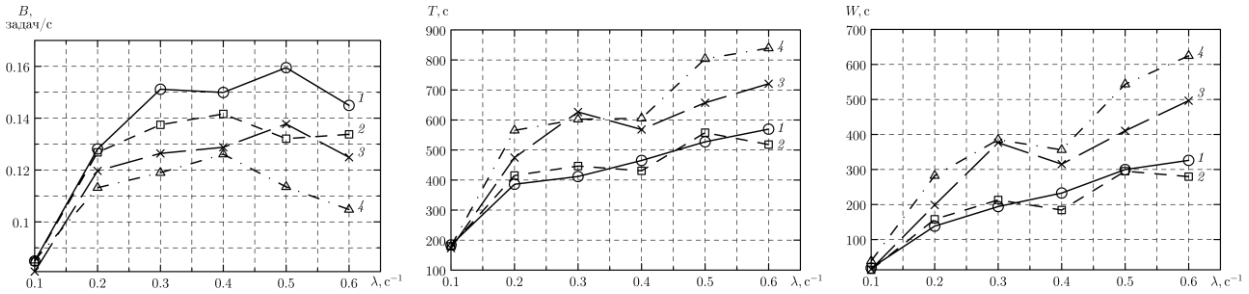



Рис. 4. Сравнение эффективности алгоритмов ДЛО и ДР:
*1* – Алгоритм ДЛО; *2* – Алгоритм ДР, $m = 2$; *3* – Алгоритм ДР, $m = 3$; *4* – Алгоритм ДР, $m = 6$




*А* *б* *В*

Рис. 5. Сравнение эффективности алгоритмов ДЛО, ДМ и ДРМ:
*1* – Алгоритм ДМ; *2* – Алгоритм ДЛО; *3* – Алгоритм ДРМ, $m = 2$; *4* – Алгоритм ДРМ, $m = 3$

Пропускная способность системы при использовании алгоритма ДР для $m \in \{2, 3\}$ выше пропускной способности при использовании алгоритма ДЛО. Деградация показателей при больших значениях $m$ связана с увеличением загрузки каналов связи при передаче входных файлов одной задачи на несколько сегментов.

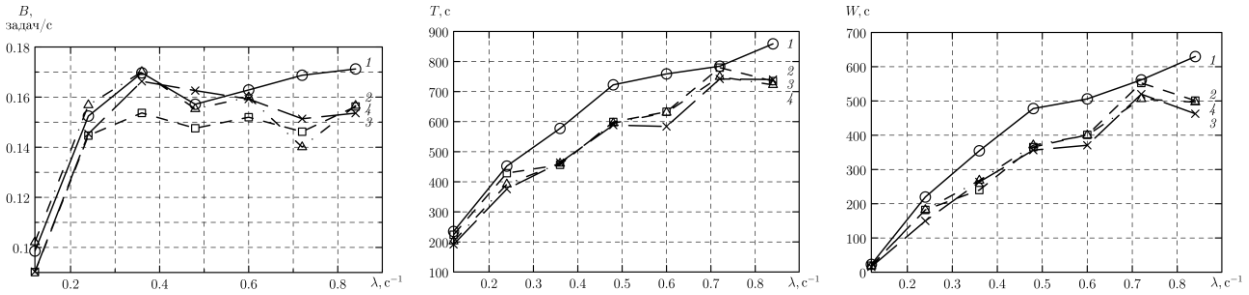
В алгоритмах ДМ и ДРМ интервал поиска подсистемы $\Delta = 30$ с, условие миграции $\varepsilon = 0.2$. Наименьшие среднее время обслуживания задач и среднее время ожидания в очереди достигнуты при использовании алгоритмов ДЛО и ДМ (рис. 5), при этом бóльшая пропускная способность системы получена для ДМ. Алгоритмы ДР и ДРМ рекомендуется применять в случае малой интенсивности потоков задач или небольших размеров входных данных.

При большой интенсивности потока задач значительно возрастает время доставки входных данных вследствие повышения загрузки каналов связи и сетевой файловой системы на сегментах. Это приводит к снижению пропускной способности системы и увеличению времени обслуживания задач (ожидания в очереди) для всех алгоритмов диспетчеризации.

Выполнено сравнение эффективности обслуживания потоков задач централизованным диспетчером GridWay и созданным децентрализованным пакетом GBroker.

Для диспетчера GBroker проведено два эксперимента. В ходе первого на подсистему Xeon80 поступал поток из $M = 300$ задач. Во втором эксперименте моделировалось распределённое обслуживание задач: одинаковые потоки из $M = 50$ задач одновременно поступали в очереди диспетчеров всех 6 подсистем. При этом в качестве структур логических связей диспетчеров использовались 2*D*-тор и полный граф, а в качестве алгоритма диспетчеризации – ДМ. Пакет GridWay установлен на сегменте Xeon80 и настроен в соответствии с рекомендациями разработчиков.

На рис. 6 видно, что пропускная способность диспетчера GBroker при обслуживании нескольких потоков превосходит пропускную способность пакета GridWay. Среднее время обслуживания и среднее время пребывания задач в очереди близки к GridWay и незначительно возрастают при централизованном обслуживании.




*а* *б* *В*

Рис. 6. Сравнение эффективности диспетчеров GBroker и GridWay:
*1* – GBroker; *2* – GBroker, полный граф; *3* – GBroker, 2*D*-тор; *4* – GridWay

### 5.2. Моделирование алгоритмов формирования расписаний решения масштабируемых задач

Наборы задач генерировались для систем с количеством ЭМ от $2^1$ до $2^{20}$ на основе модели рабочей загрузки [4]. Рассматривались наборы с количеством задач от $10^1$ до $10^5$. Приоритеты значениям параметров задач задавались путём их простой нумерации.

Под качеством получаемого решения понимаем: полученное значение функции *T(S)*, средняя удовлетворённость пользователей и уровень загрузки вычислительных ресурсов (коэффициент раскроя) при решении задач по сформированному расписанию.

Моделирование генетического алгоритма требует задания трёх параметров: количества жизненных циклов популяций без улучшения потомства (обозначим как STEPSGA), размера популяции (обозначим как PSGA), вероятности выполнения кроссинговера и мутации (обозначим как PGA).

Моделирование показало следующее. Для любой минимально допустимой средней «удовлетворённости» пользователей *e* лучшим значением для STEPSGA является 5. При дальнейшем увеличении значении STEPSGA время работы алгоритма растёт, а качество получаемого решения практически не изменяется (рис. 7). TFFD(S) – это время решения задач по расписанию, для формирования которого задачам назначены значения параметров с максимальным приоритетом и однократно выполнен алгоритм FFDH.

С увеличением значения параметра *e* уменьшается время работы алгоритма. Это связано с тем, что количество вариантов значений параметров с увеличением *e* сокращается (рис. 8). Наилучшим размером популяции PSGA является 32 особи для любого значения минимально допустимой средней «удовлетворённости» пользователей (рис. 9). Вероятность выполнения операторов кроссинговера и мутации существенно не влияет на качество получаемого решения (рис. 10).

Выполнение параллельной версии алгоритма целесообразно для наборов с количеством задач больше 1000. Ускорение алгоритма практически линейно. Ухудшение значения функции T(S) составило не более 10 %.

Моделирование второго этапа подтвердило на практике необходимость и достаточность (12) для обеспечения минимума функции штрафа за задержку решения задач набора (10).

## Заключение

В статье представлен масштабируемый инструментарий параллельного мультипрограммирования распределённых вычислительных систем.

Разработанные алгоритмы для планирования решения масштабируемых задач на ВС с учётом штрафов за задержку их решения и приоритетов выбора возможных конфигураций подсистем лягут в основу планировщика ресурсов распределённых ВС. Результаты исследования созданного инструментария диспетчеризации параллельных MPI-программ на мультикластерной ВС показали, что среднее время обслуживания задачи и при децентрализованной, и при централизованной диспетчеризации сопоставимы. Время диспетчеризации достаточно мало по сравнению со временем выполнения задач.

Инструментарий параллельного мультипрограммирования – один из необходимых компонентов обеспечения живучести, надёжности и эффективности использования ресурсов пространственно-распределённых вычислительных систем.

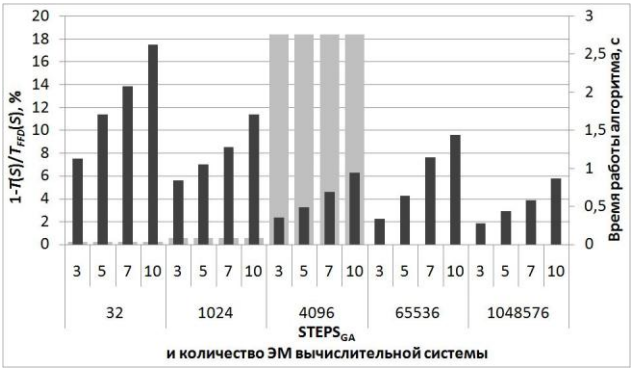



Рис. 7. Влияние параметра $STEPS_{GA}$ на качество работы последовательного ГА ($L$ = 1000, $PS_{GA}$ = 32, $P_{GA}$ = 0.8, $e$ = 0.95):
– улучшение значения функции $T(S)$, %;
– время работы алгоритма, $c$

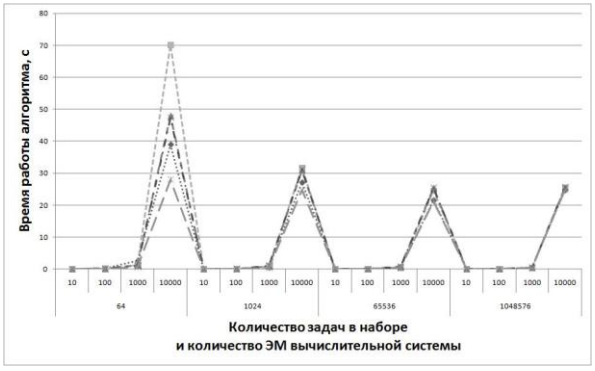



Рис. 8. Влияние $e$ на время работы последовательного ГА ($STEPS_{GA}$ = 5, $PS_{GA}$ = 32, $P_{GA}$ = 0.8):
– $e$ = 0,5; – $e$ = 0,7; – $e$ = 0,8; – $e$ = 0,9; – $e$ = 1


Рис. 9 Влияние размера популяции $PS_{GA}$ на качество работы ГА ($L$ = 10000, $STEPS_{GA}$ = 5, $P_{GA}$ = 0.8, $e$ = 0.95):
– улучшение значения функции $T(S)$, %;
– время работы алгоритма, $c$


Рис. 10. Влияние вероятности выполнения операторов кроссинговера и мутации на качество работы ГА ($L$ = 10000, $PS_{GA}$ = 32, $e$ = 0.95):
– улучшение значения функции $T(S)$, %;
– время работы алгоритма, $c$

## Литература

1. Указ Президента РФ «Об утверждении приоритетных направлений развития науки, технологии и техники в Российской Федерации и перечня критических технологий Российской Федерации» от 07.07.2011 № 899 // URL: `http://kremlin.ru/news/11861` (дата обращения: 19.11.2011).
2. Хорошевский В.Г. Распределённые вычислительные системы с программируемой структурой // Вестник СибГУТИ. – 2010. – № 2 (10). – С. 3-41.
3. Хорошевский В.Г., Курносов М.Г., Мамойленко С.Н. Пространственно-распределенная мультикластерная вычислительная система: архитектура и программное обеспечение // Вестник ТГУ. Управление, вычислительная техника и информатика. – 2011. – № 1(14). – С. 79-84.
4. Cirne W., Berman F. A model for moldable supercomputer jobs // 15th Intl. Parallel & Distributed Processing Symp. – 2001.
– URL: `http://www.lsd.dsc.ufpb.br/papers/moldability-model.pdf` (дата обращения: 24.11.2011).

5. Таха Х. Введение в исследование операций : 6-е изд. / Таха Хэмди А., пер. с англ. В.И. Тюпти, А.А. Минько. – М.: Вильямс, 2001. – 911 с.
6. Евреинов Э.В., Хорошевский В.Г. Однородные вычислительные системы. Новосибирск: Наука, 1978. – 319 с.
7. Бруно Дж. Л., Грэхем Р.Л., Коглер В.Г., Коффман Э.Г. мл., Сети Р., Ульман Дж.Д., Штиглиц К., Теория расписаний и вычислительные машины // Под ред. Б.А. Головкина, пер. с англ. В.М. Амочкина, М.: Изд-во «Наука», 1984. – 336 с.
8. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982. – 416 с.
9. Coffman E.G. Performance bounds for level-oriented two-dimensional packing algorithms / E.G. Coffman et al. // SIAM Journal on Computing. – 1980. – P. 808–826.
10. Rohlfshagen P., Bullinaria J.A. A genetic algorithm with exon shuffling crossover for hard bin packing problems // Proc. of the 9th annual conference on Genetic and evolutionary computation. ACM NewYork, 2007. pp. 1365-1371.
11. Smith W. Various optimizers for single-stage production // Naval res. Logist. Quart. 3. – 1956. – P. 59-66.
12. Курносов М.Г., Пазников А.А. Децентрализованное обслуживание потоков параллельных задач в пространственно-распределённых вычислительных системах // Вестник СибГУТИ. – 2010. – № 2 (10). – С. 79-86.
13. Parallel workloads archive // URL: http://www.cs.huji.ac.il/labs/parallel/workload/ (дата обращения: 24.11.2011).
14. Downey A.B. A Parallel Workload Model and Its Implications for Processor Allocation // 6th Intl. Symp. High Performance Distributed Comput., 1997.
15. Хорошевский В.Г., Курносов М.Г., Мамойленко С.Н., Поляков А.Ю. Архитектура и программное обеспечение пространственно-распределённых вычислительных систем // Вестник СибГУТИ. – 2010. – № 2 (10). – С. 112-122.

*Статья поступила в редакцию 19.11.2011*

**Хорошевский Виктор Гаврилович**
Член-корр. РАН, д.т.н., заведующий Лабораторией вычислительных систем Института физики полупроводников им. А.В. Ржанова СО РАН, директор Центра параллельных вычислительных технологий, заведующий Кафедрой вычислительных систем Сибирского государственного университета телекоммуникаций и информатики.
Тел.&факс: (383) 333-21-71, 269-82-75; e-mail: khor@isp.nsc.ru, khor@sibsutis.ru

**Курносов Михаил Георгиевич**
К.т.н., доцент Кафедры вычислительных систем Сибирского государственного университета телекоммуникаций и информатики, младший научный сотрудник Лаборатории вычислительных систем Института физики полупроводников им. А.В. Ржанова СО РАН.
Тел.&факс: (383) 330-56-26, 269-82-75; e-mail: mkurnosov@gmail.com

**Мамойленко Сергей Николаевич**
Доцент, к.т.н., заместитель заведующего Кафедрой вычислительных систем Сибирского государственного университета телекоммуникаций и информатики.
Тел.&факс: (383) 269-82-75, e-mail: msn@sibsutis.ru, msn@isp.nsc.ru

**Павский Кирилл Валерьевич**
К.т.н., научный сотрудник Лаборатории вычислительных систем Института физики полупроводников им. А.В. Ржанова СО РАН.

Тел.&факс: (383) 333-21-71; e-mail: pkv@isp.nsc.ru

**Ефимов Александр Владимирович**
Ст. преп. Кафедры вычислительных систем Сибирского государственного университета телекоммуникаций и информатики.
Тел.&факс: (383) 269-82-75, e-mail: efimov@cpct.sibsutis.ru

**Пазников Алексей Александрович**
Аспирант Кафедры вычислительных систем Сибирского государственного университета телекоммуникаций и информатики.
Тел.&факс: (383) 269-82-75; e-mail: apaznikov@gmail.com

**Перышкова Евгения Николаевна**
Аспирант Кафедры вычислительных систем Сибирского государственного университета телекоммуникаций и информатики.
Тел.&факс: (383) 269-82-75, e-mail: e_perishkova@csc.sibsutis.ru

**Scalable Toolkit for Parallel Multiprogramming of Spatially-Distributed Computing Systems**

**V.G. Khoroshevsky, M.G. Kurnosov, S.N. Mamoilenko, K.V. Pavsky, A.V. Efimov, A.A. Paznikov, E.N. Perishkova**

The algorithms of dispatching and scheduling of parallel tasks solving on spatially-distributed computing systems are offered. The description of scalable toolkit of parallel multiprogramming on spatially-distributed computing systems using the offered algorithms is presented. The results of the offered algorithms simulation on spatially-distributed multi-cluster computing system are presented.

*Keywords*: parallel multiprogramming, scheduling, dispatching, distributed systems.