

МОДЕЛИРОВАНИЕ АЛГОРИТМОВ ДЕЦЕНТРАЛИЗОВАННОГО ОБСЛУЖИВАНИЯ ПОТОКОВ ПАРАЛЛЕЛЬНЫХ ЗАДАЧ В GRID-СИСТЕМАХ

М. Г. Курносов, А. А. Пазников

Сибирский государственный университет телекоммуникаций и информатики,
630102, Новосибирск, Россия

УДК 004.272

Предложены децентрализованные алгоритмы диспетчеризации в пространственно распределенных вычислительных системах параллельных задач с целью минимизации времени их обслуживания. Описана функциональная структура разработанного программного пакета GBroker децентрализованного управления ресурсами мультикластерных вычислительных и GRID-систем. Представлены результаты моделирования созданных алгоритмов, обслуживающих потоки параллельных задач на действующей пространственно распределенной мультикластерной вычислительной системе.

Ключевые слова: диспетчеризация параллельных программ, пространственно распределенные вычислительные системы, GRID-системы.

Decentralized scheduling algorithms in geographically-distributed computer systems are proposed in this paper. The main goal is to minimize the job service time. Functional structure of the developed software suite of the decentralized resource management is shown. Experimental results with the developed algorithms at the service of parallel jobs streams are presented. Experiments were conducted on the geographically-distributed multicluster computer system.

Key words: parallel programs scheduling, geographically-distributed computer systems, GRID-systems.

Введение. При решении сложных задач науки и техники широкое применение получили пространственно распределенные вычислительные системы (ВС) — макроколлективы рассредоточенных вычислительных средств (подсистем), взаимодействующих через локальные и глобальные сети связи (включая сеть Internet) [1]. К числу таких систем относятся пространственно распределенные мультикластерные вычислительные и GRID-системы.

Важной проблемой организации функционирования пространственно распределенных ВС является диспетчеризация. Для решения каждой поступившей в ВС параллельной задачи требуется определить подсистемы. Необходимо учитывать изменения состава и загрузки ВС с течением времени.

В пространственно распределенных вычислительных и GRID-системах каждая подсистема функционирует под управлением локальной системы управления ресурсами (СУР) (TORQUE, Altair PBS Pro, SLURM и др.), которая поддерживает очереди пользовательских задач и выделяет для них вычислительные ресурсы (процессорные ядра).

Работа выполнена при финансовой поддержке РФФИ (коды проектов 11-07-00105, 10-07-00157), Министерства образования и науки РФ (госконтракт № 07.514.11.4015) и Совета по грантам Президента РФ для государственной поддержки ведущих научных школ РФ (грант № НШ-2175.2012.9).

Централизованные средства диспетчеризации задач в пространственно распределенных ВС подразумевают наличие в системе центрального диспетчера, поддерживающего глобальную очередь задач. Отказ такого диспетчера может привести к неработоспособности всей системы. Кроме того, в случае применения таких средств в больших масштабах ВС возрастают временные затраты на поиск необходимых ресурсов.

Основными программными пакетами организации централизованной диспетчеризации параллельных задач в пространственно распределенных вычислительных и GRID-системах являются GridWay [2], AppLeS [3], GrADS [4], Nimrod/G [5], Condor-G [6]. Пакет GridWay входит в состав пакета Globus Toolkit и является наиболее распространенным GRID-диспетчером. В этом пакете минимизируется время обслуживания задач и поддерживается механизм их миграции между подсистемами. В AppLeS диспетчеризация выполняется на уровне самого приложения. Это приводит к уменьшению области применимости указанного пакета. Пакет GrADS, как и GridWay, поддерживает миграцию задач и допускает указание зависимостей по данным между задачами. Пакет Nimrod/G на основе экономических моделей обеспечивает равновесие между условными поставщиками (подсистемами) и потребителями вычислительных ресурсов (задачами). В Condor-G зависимости между задачами задаются в виде ориентированного ациклического графа.

При децентрализованной диспетчеризации в распределенной ВС функционирует коллектив диспетчеров, которые поддерживают распределенную очередь задач и совместно принимают решение о выборе ресурсов. Это позволяет повысить живучесть — способность ВС продолжать работу при отказах отдельных подсистем.

Для распределенных вычислительных систем с программируемой структурой созданы эффективные децентрализованные алгоритмы управления ресурсами [7, 8]. Однако область применимости этих методов для пространственно распределенных вычислительных и GRID-систем ограничена. В алгоритмах не учитываются производительность каналов связи между ресурсами и возможность образования очередей задач на подсистемах.

В данной работе предлагаются децентрализованные алгоритмы и программное обеспечение диспетчеризации параллельных задач в пространственно распределенных вычислительных и GRID-системах. В алгоритмах учитываются изменения состава и загрузки ресурсов распределенных ВС в процессе их функционирования.

1. Децентрализованная диспетчеризация параллельных задач в пространственно распределенных ВС. Пусть имеется пространственно распределенная 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). Введем следующие обозначения: n_i — количество ЭМ, входящих в состав подсистемы $i \in S = 1, 2, \dots, H$; 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 сообщения размером m байт между подсистемами $i \in S, j \in S$ ($[t(i, j, m)] = c$). Считается, что все подсистемы объединены сетью связи.

На каждой подсистеме присутствуют локальная СУР и децентрализованный диспетчер, который поддерживает свою очередь параллельных задач и осуществляет поиск вычислительных ресурсов для их выполнения.

Коллектив диспетчеров представлен в виде ориентированного графа $G(S, E)$, в котором вершинам соответствуют диспетчеры, а ребрам — логические связи между ними (рис. 1). Наличие дуги $(i, j) \in E$ в графе означает, что диспетчер i может отправлять задачи диспет-

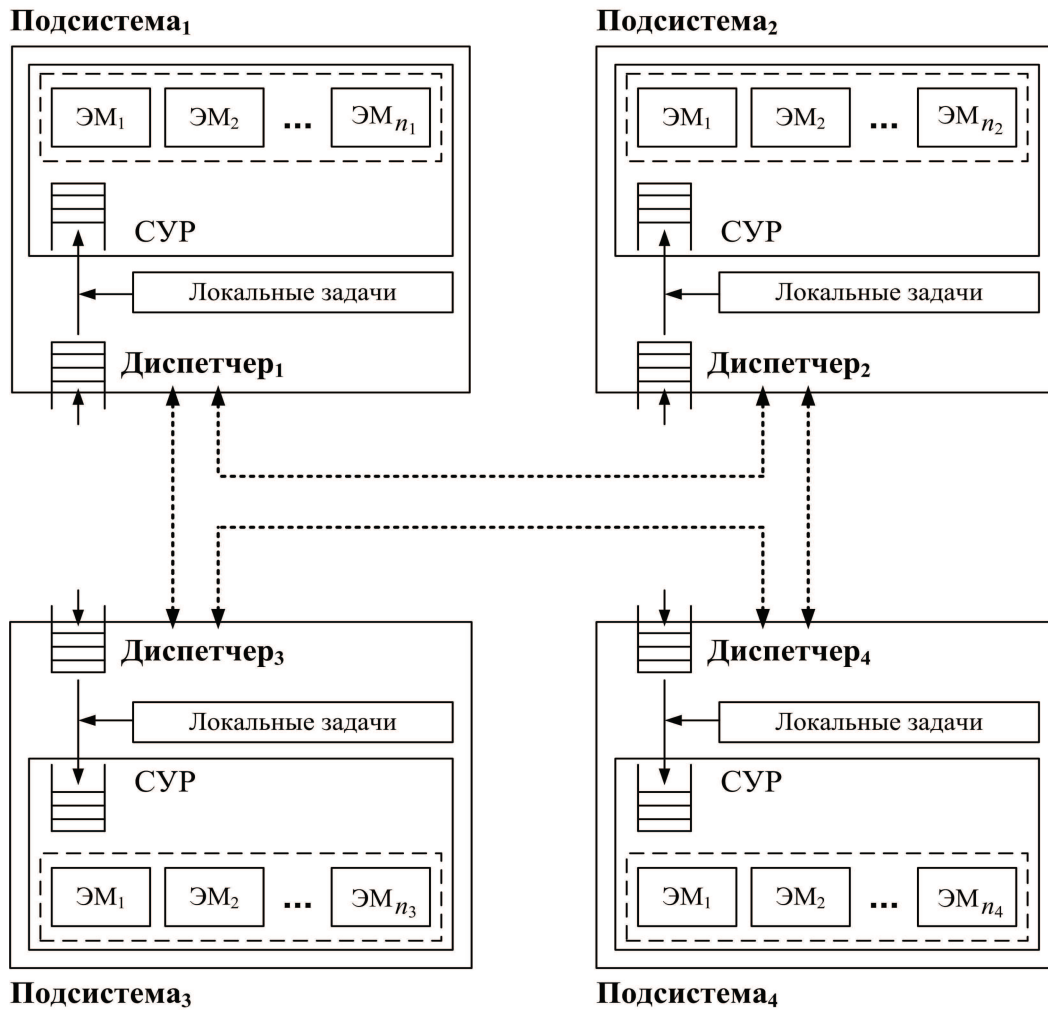


Рис. 1. Пример локальных окрестностей диспетчеров
 $H = 4, L(1) = \{2, 3\}, L(2) = \{1, 4\}, L(3) = \{1, 4\}, L(4) = \{2, 3\}$

черу j . Множество всех вершин j , смежных вершине i , образуют ее локальную окрестность $L(i) = \{j \in S | (i, j) \in E\}$.

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, \dots, z_k исполняемого и входных файлов программы в байтах, а также номера h_1, h_2, \dots, h_k подсистем, на которых размещены соответствующие файлы ($h_l \in S$). Диспетчер (в соответствии с реализованным в нем алгоритмом) выполняет поиск (суб)оптимальной подсистемы $j^* \in L(i) \cup \{i\}$ (или подсистем $j_1^*, j_2^*, \dots, j_m^*$) из его локальной окрестности.

1.1. Алгоритм локально-оптимальной диспетчеризации (ДЛО). Алгоритм осуществляет выбор локально-оптимальной подсистемы.

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

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

Здесь $t_j = \sum_{l=1}^k t(h_l, j, z_l)$ — 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, приходящее 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.

1.2. *Алгоритм диспетчеризации на основе репликации задач (ДР).* В основе данного алгоритма лежит назначение задачи одновременно на несколько подсистем.

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

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

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

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

1.3. *Алгоритм диспетчеризации на основе миграции задач (ДМ).* В данном алгоритме реализован периодический поиск новых подсистем для задач из очереди диспетчера.

Шаг 1. В 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 Δ запускает процедуру ДЛЮ поиска подсистемы j' для задачи в очереди диспетчера j^* .

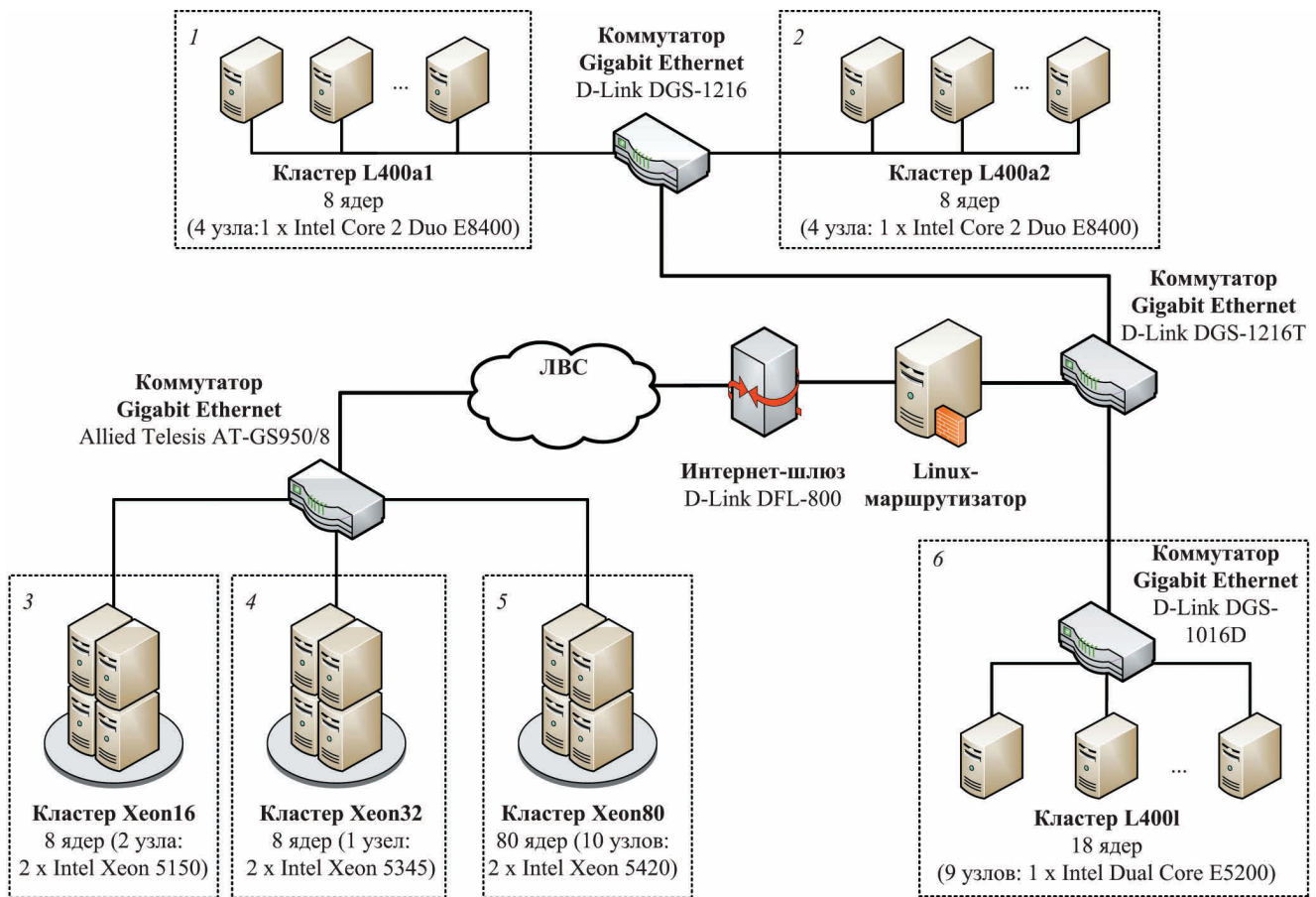
Шаг 4. Если для найденной подсистемы выполняется условие $F(j^*) - F(j') > \varepsilon$, то выполняется миграция задачи в очередь подсистемы j' (задача удаляется из очереди диспетчера j^*).

1.4. *Алгоритм диспетчеризации на основе репликации и миграции задач (ДРМ).* Алгоритм основан на комбинации двух подходов — назначении на несколько подсистем и миграции из очереди.

Следует отметить, что вычислительная сложность поиска подсистем предложенными алгоритмами не зависит от количества H подсистем, так как поиск осуществляется только в локальных окрестностях диспетчеров. Это обеспечивает применимость алгоритмов в большемасштабных пространственно распределенных ВС.

2. Программный пакет GBroker децентрализованной диспетчеризации задач.

В Центре параллельных вычислительных технологий Сибирского государственного университета телекоммуникаций и информатики (ЦПВТ СибГУТИ) и лаборатории вычислительных систем Института физики полупроводников им. А. В. Ржанова СО РАН (ИФП СО РАН) создан и развивается программный пакет GBroker [9] децентрализованной диспетчеризации параллельных задач в пространственно распределенных ВС. Пакет, разработанный на языке ANSI C для операционной системы GNU/Linux, включает диспетчер GBroker, клиентское приложение 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 — модуль мониторинга

Рис. 2. Тестовая конфигурация мультикластерной ВС ($H = 6, N = 130$):

1-6 — номера подсистем

вычислительных ресурсов подсистем локальной окрестности диспетчера. NetMon — модуль мониторинга производительности каналов связи между подсистемами.

Администратор устанавливает пакет на всех подсистемах, настраивает локальные окрестности диспетчеров и модулей DCSMon и NetMon. Пользователь формирует задачу, состоящую из параллельной программы и ресурсного запроса на языке Job submission description language (JSDL), и отправляет ее средствами GClient любому из диспетчеров GBroker распределенной ВС. Передача файлов в системе обеспечивается службой GridFTP.

3. Экспериментальное исследование алгоритмов. Исследование созданных алгоритмов проводилось на пространственно распределенной мультикластерной ВС, созданной ЦПВТ СибГУТИ совместно с лабораторией вычислительных систем ИФП СО РАН (рис. 2). На каждом сегменте установлены операционная система GNU/Linux, локальная система управления ресурсами TORQUE 2.3.7, пакет Globus Toolkit 5.0, децентрализованный диспетчер GBroker, системы мониторинга DCSMon и NetMon. 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; parallel ocean program (POP2) — пакет моделирования процессов в океане; LAMMPS — пакет решения задач молекулярной динамики; RAxML — пакет моделирования задач биоинформатики; Tachyon — пакет расчета графических сцен. Входные данные для тестовых

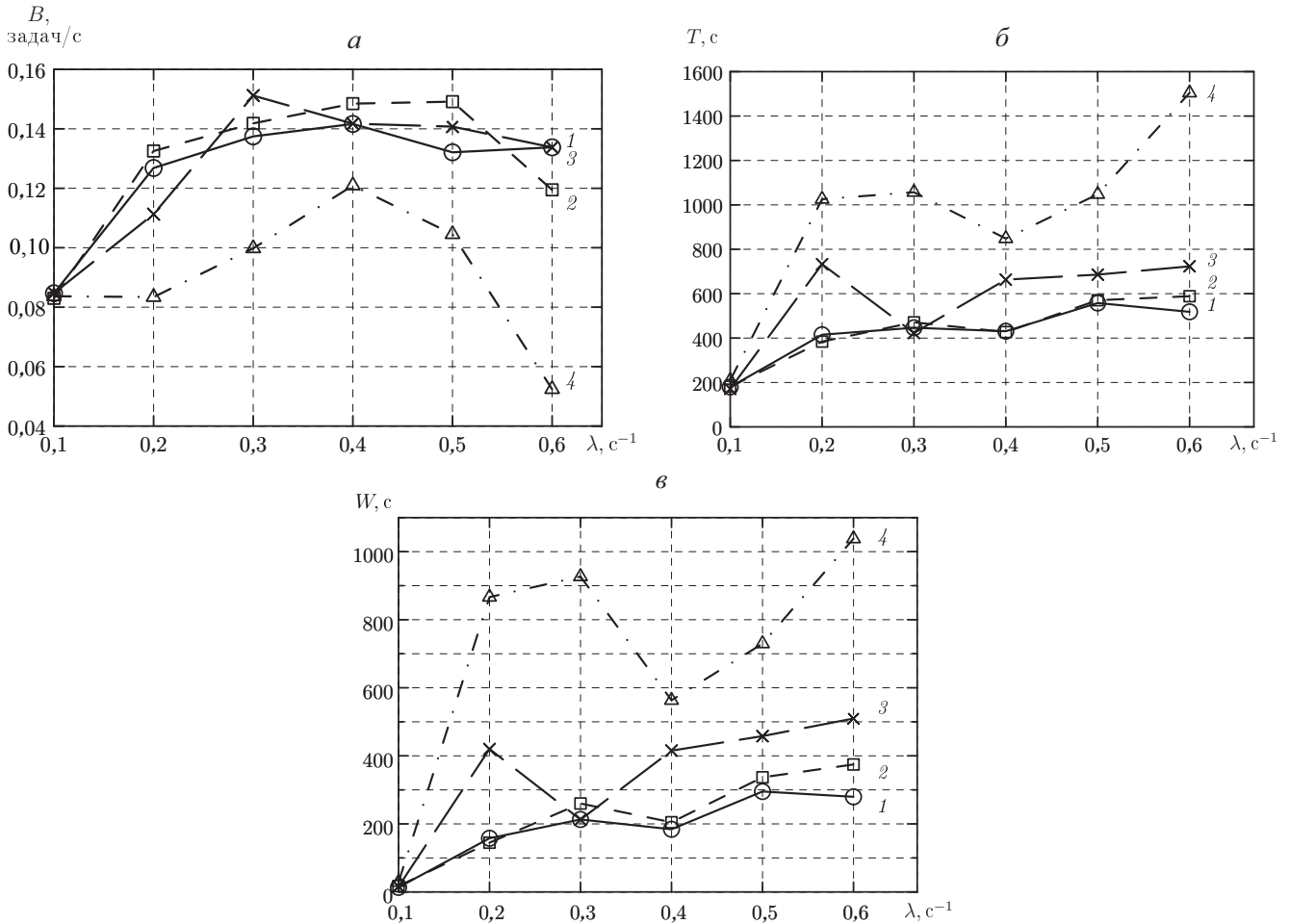


Рис. 3. Эффективность алгоритмов ДЛЮ и ДР:

1 — алгоритм ДЛЮ; 2–4 — алгоритм ДР (2 — $m = 2$; 3 — $m = 3$; 4 — $m = 6$);
 а — пропускная способность системы; б — среднее время обслуживания задачи;
 в — среднее время пребывания задачи в очереди

задач размещались на сегменте Xeon80. На ту же подсистему доставлялись результаты выполнения программ.

Генерировались простейшие потоки с различной интенсивностью λ 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 $k \in 1, 2, \dots, M$ на вход диспетчера, t'_k — время начала решения задачи k , t''_k — время завершения решения задачи k . Пусть τ — 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:

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

3.1. Сравнительный анализ алгоритмов. На рис. 3, 4 приведены результаты сравнения эффективности алгоритмов ДЛЮ, ДР, ДМ и ДРМ при обслуживании потока из $M = 200$

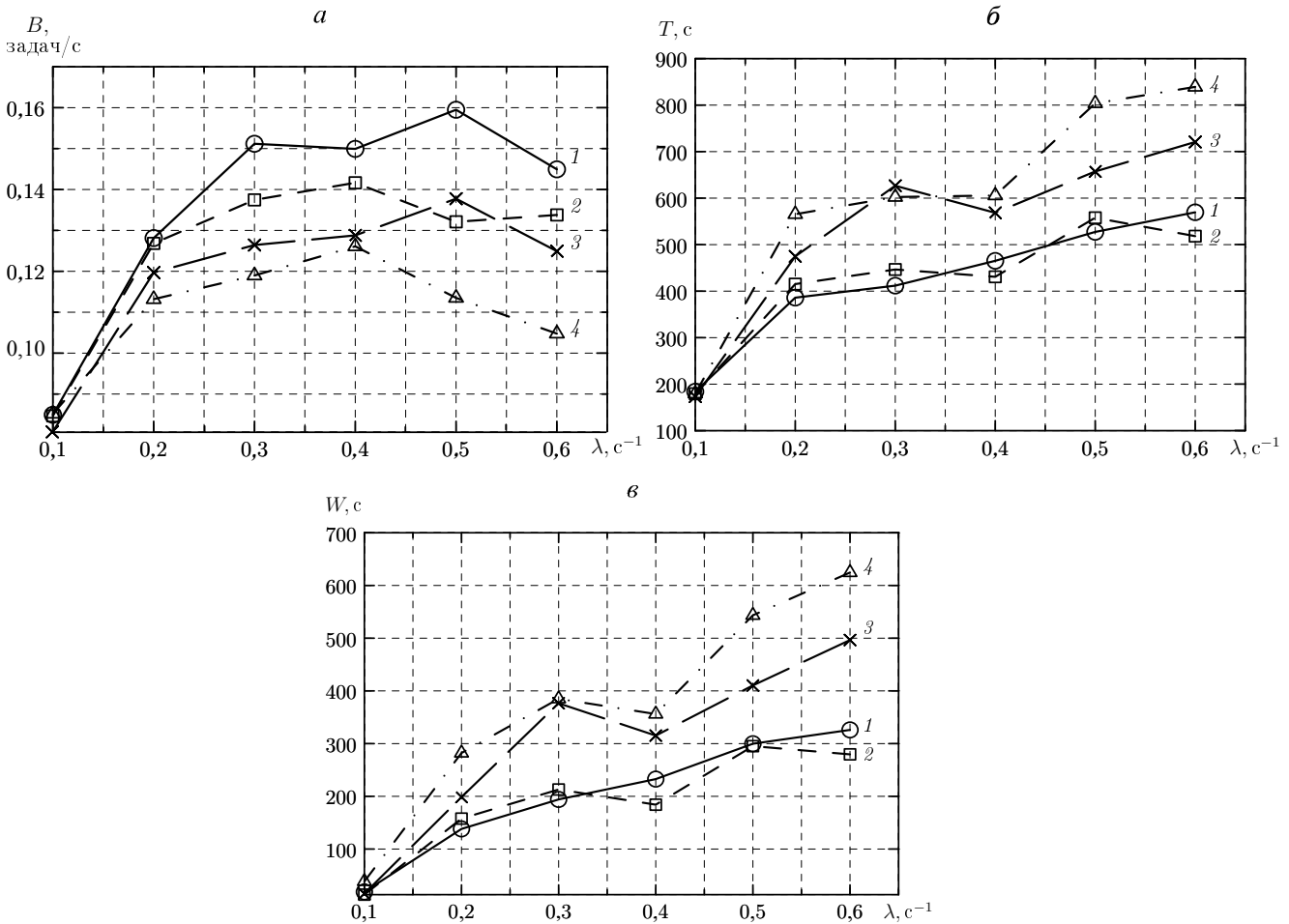


Рис. 4. Эффективность алгоритмов ДЛО, ДМ и ДРМ:

1 — алгоритм ДМ; 2 — алгоритм ДЛО; 3, 4 — алгоритм ДРМ (3 — $m = 2$, 4 — $m = 3$);

а — пропускная способность системы; б — среднее время обслуживания задачи;

в — среднее время пребывания задачи в очереди

задач. Поток задач поступал на подсистему Хеон80, локальные окрестности диспетчеров имели структуру полного графа.

При использовании алгоритма ДР для $m \in \{2, 3\}$ пропускная способность системы выше, чем при использовании алгоритма ДЛО. При больших значениях m деградация показателей обусловлена 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 (см. рис. 4), при этом большая 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.

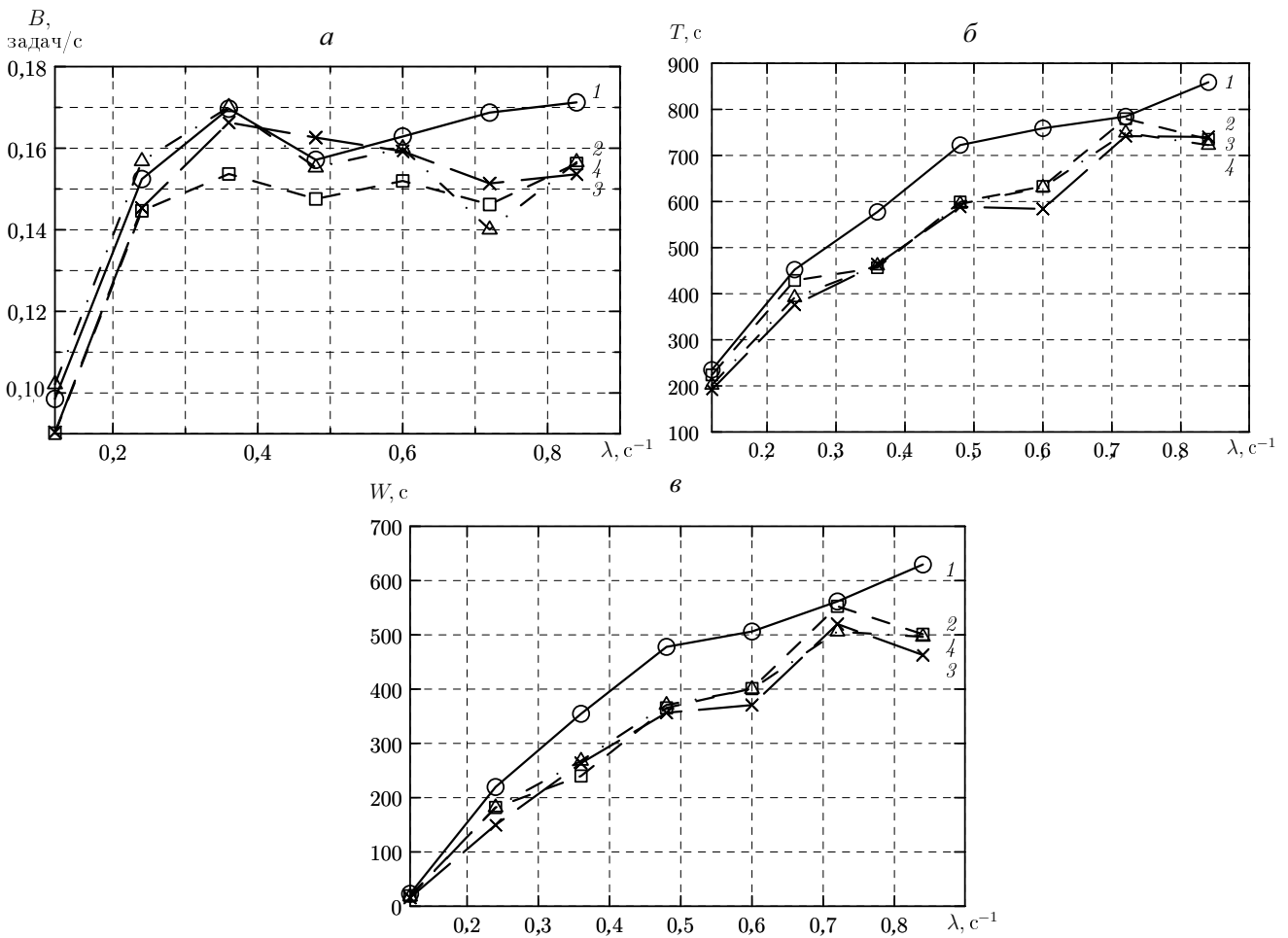


Рис. 5. Эффективность диспетчеров GBroker и GridWay:
 1 — GBroker; 2 — GBroker (полный граф); 3 — GBroker (2D-тор); 4 — GridWay;
 а — пропускная способность системы; б — среднее время обслуживания задачи;
 в — среднее время пребывания задачи в очереди

3.2. Сравнительный анализ диспетчеров GBroker и GridWay. Выполнено сравнение эффективности обслуживания потоков задач централизованным диспетчером GridWay и созданным децентрализованным пакетом GBroker.

Для диспетчера GBroker проведено два эксперимента. В ходе первого эксперимента на подсистему Xeон80 поступал поток из $M = 300$ задач. Во втором эксперименте моделировалось распределенное обслуживание задач: одинаковые потоки из $M = 50$ задач одновременно поступали в очереди диспетчеров всех шести подсистем. При этом в качестве структур логических связей диспетчеров использовались 2D-тор и полный граф, а в качестве алгоритма диспетчеризации — ДМ. Пакет GridWay установлен на сегменте Xeон80 и настроен в соответствии с рекомендациями разработчиков.

На рис. 5 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 соответствующим параметрам в пакете GridWay и незначительно возрастают при централизованном обслуживании.

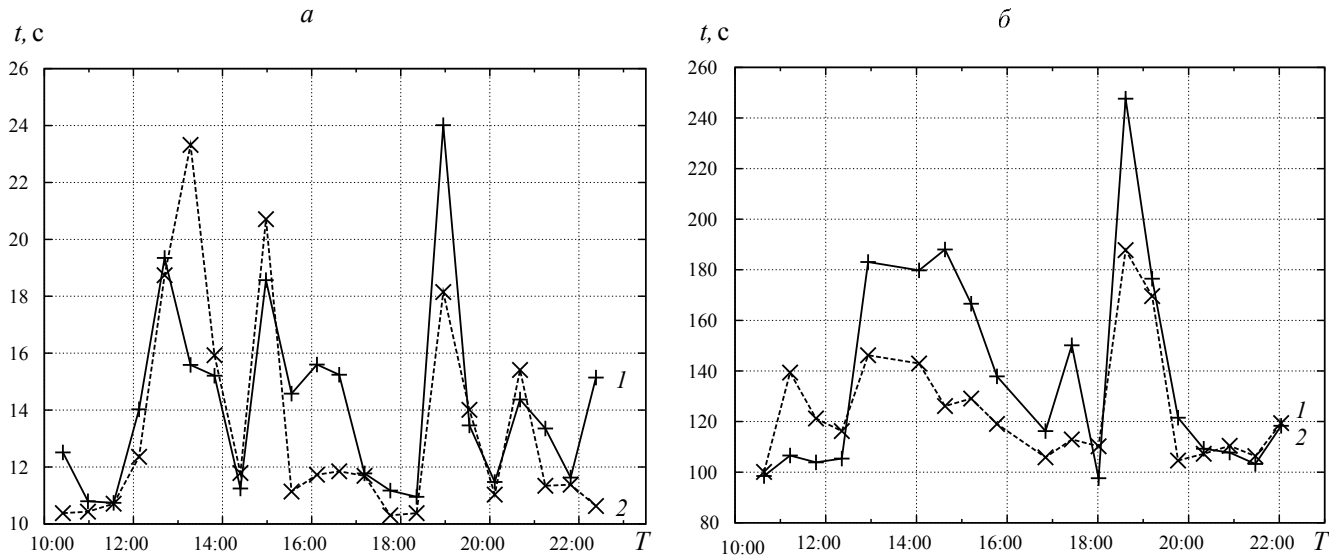


Рис. 6. Прогноз времени t передачи файлов между подсистемами Xeon80 и L4001 (Gigabit Ethernet):
 $a - m = 100$ Мбайт, $b - m = 1000$ Мбайт, 1 — NetMon; 2 — GridFTP

3.3. *Моделирование средств мониторинга производительности каналов связи.* Модуль NetMon прогнозирования производительности каналов связи установлен на всех подсистемах и выполняет периодический сбор информации о времени передачи файлов различных размеров между подсистемами распределенной ВС. Администратор системы выполняет конфигурацию службы NetMon, задавая таблицу тестовых значений. Периодически проводится измерение времени передачи тестовых файлов средствами службы GridFTP из пакета Globus Toolkit. Прогноз времени передачи файла размером m байт составляется на основе таблицы, полученной путем кусочно-линейной интерполяции.

Выполнена оценка погрешности прогноза времени доставки данных. В эксперименте с интервалом 300 с в течение 12 ч выполнялись измерения при передаче файлов размерами 100 и 1000 Мбайт по протоколу GridFTP между различными подсистемами. На рис. 6 видно, что относительное отклонение прогнозируемого времени передачи файлов от измеренного в среднем не превышает 35 %, что, по мнению авторов, приемлемо в GRID-системах.

Заключение. Централизованные системы диспетчеризации задач в большомасштабных распределенных вычислительных системах характеризуются вычислительной сложностью поиска требуемых ресурсов. Децентрализованная диспетчеризация существенно проще централизованной и позволяет повысить живучесть распределенных ВС.

Результаты исследования созданных алгоритмов и программных средств диспетчеризации параллельных задач на мультикластерной вычислительной системе показали, что и при децентрализованной, и при централизованной диспетчеризации средние времена обслуживания потоков задач сопоставимы. Время диспетчеризации достаточно мало по сравнению со временем выполнения задач. Инструментарий децентрализованной диспетчеризации — один из необходимых компонентов пространственно распределенных вычислительных и GRID-систем.

Созданный пакет GBroker является свободно распространяемым. Для организации децентрализованной диспетчеризации задач в пространственно распределенной ВС достаточно установить пакет GBroker на всех подсистемах и выполнить конфигурацию диспетчеров в соответствии с предлагаемыми рекомендациями.

Список литературы

1. ХОРОШЕВСКИЙ В. Г. Распределенные вычислительные системы с программируемой структурой // Вестн. СибГУТИ. 2010. № 2. С. 3–41.
2. HUEDO E., MONTERO R., LLORENTE I. A framework for adaptive execution on grids // Software-Practice Experience. 2004. V. 34. P. 631–651.
3. BERMAN F., WOLSKI R., CASANOVA H. Adaptive computing on the grid using AppLeS // IEEE Trans. Parallel Distributed Systems. 2003. V. 14, N 4. P. 369–382.
4. COOPER K., DASGUPTA A., KENNEDY A., KOELBEL C. ET AL. New grid scheduling and rescheduling methods in the GrADS project // Intern. J. Parallel Programming. 2005. V. 33, N 2/3. P. 209–229.
5. BUYYA R., ABRAMSON D., GIDDY J. Nimrod/G: An architecture for a resource management and scheduling system in a global computational grid // Proc. of the 4th Intern. conf. on high performance computing in Asia-Pacific region, Beijing (China), 14–17 May 2000. Beijing: IEEE CS Press, 2000. P. 283–289.
6. FREY J., TANNENBAUM T., LIVNY M., FOSTER I. Condor-G: A computation management agent for multi-institutional grids // Cluster Comput. 2001. V. 5. P. 237–246.
7. КОРНЕЕВ В. В. Архитектура вычислительных систем с программируемой структурой. Новосибирск: Наука. Сиб. отд-ние, 1985. 164 с.
8. МОНАХОВ О. Г. Параллельные системы с распределенной памятью: управление ресурсами и заданиями / О. Г. Монахов, Э. А. Монахова. Новосибирск: ИВМиМГ СО РАН, 2001. 168 с.
9. КУРНОСОВ М. Г., ПАЗНИКОВ А. А. Инструментарий децентрализованного обслуживания потоков параллельных MPI-задач в пространственно распределенных мультикластерных вычислительных системах // Вестн. ТГУ. 2011. № 3. С. 78–85.

*Курносков Михаил Георгиевич — канд. техн. наук,
доц. Сибирского государственного университета телекоммуникаций и информатики;
тел.: +7 (383) 269-82-86; e-mail: tkurnosov@gmail.com;*

*Пазников Алексей Александрович — асп. Сибирского государственного
университета телекоммуникаций и информатики; e-mail: a paznikov@gmail.com*

Дата поступления — 23.01.12 г.