УДК 004.272

М.Г. Курносов, А.А. Пазников

ДЕЦЕНТРАЛИЗОВАННЫЕ АЛГОРИТМЫ ДИСПЕТЧЕРИЗАЦИИ ПРОСТРАНСТВЕННО-РАСПРЕДЕЛЁННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ[1]

Рассмотрена задача управления ресурсами пространственно-распределённых вычислительных систем. Предложены алгоритмы децентрализованной диспетчеризации: локально-оптимальный (ДЛО), алгоритмы на основе репликации (ДР) и миграции (ДМ) задач и с применением комбинированного подхода (ДРМ). Проведено исследование алгоритмов на мультикластерной ВС. Выполнено сравнение эффективности пакета GBroker децентрализованной диспетчеризации параллельных программ с централизованным диспетчером GridWay.

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

В настоящее время при решении сложных задач науки и техники широко используются пространственно-распределённые вычислительные системы (ВС). В архитектурном плане такие ВС представляют собой макроколлективы рассредоточенных вычислительных средств (подсистем), взаимодействующих между собой через локальные и глобальные сети связи (включая сеть Internet) [1]. К пространственно-распределённым относятся мультикластерные вычислительные и GRID-системы.

К значимым проблемам организации функционирования пространственно-распределённых ВС относится диспетчеризация. Для каждой параллельной задачи, из поступивших в ВС, требуется определить подсистемы для её решения. При этом важно учитывать изменения состава и загрузки ВС с течением времени.

В пространственно-распределённых 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).

Централизованные средства диспетчеризации задач в пространственно-распределённых 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 большемасштабных ВС возрастают временные затраты на поиск требуемых ресурсов.

Основными программными пакетами организации централизованной диспетчеризации параллельных задач в пространственно-распределённых вычислительных и GRID-системах являются: GridWay [2], AppLeS [3], GrADS [4], Nimrod/G [5],

[1] Работа выполнена при поддержке РФФИ (гранты № 11-07-00105, 10-07-00157), Совета по грантам Президента РФ для поддержки ведущих научных школ (грант НШ-5176.2010.9) и в рамках госконтракта № 07.514.11.4015 с Минобрнауки РФ.

Condor-G [6]. Пакет GridWay входит в состав пакета Globus Toolkit и является наиболее распространённым GRID-диспетчером. В нём преследуется цель минимизации времени обслуживания задач и поддерживается механизм их миграции между подсистемами. В AppLeS диспетчеризация выполняется на уровне самого приложения, что требует от пользователя знания конфигурации системы – это снижает универсальность указанного пакета. Пакет GrADS, как и GridWay, поддерживает миграцию задач и допускает указание зависимостей по данным между задачами. 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 распределённой ВС функционирует коллектив диспетчеров, которые поддерживают распределённую очередь задач и совместно принимают решение о выборе ресурсов. Это позволяет достичь живучести ВС – способности систем продолжать работу при отказах отдельных подсистем.

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

В статье предложено три новых алгоритма, основанных на репликации задач (ДР), миграции задач (ДМ) и применении комбинированного подхода (ДРМ). Подходы, реализованные в этих алгоритмах, позволяют качественнее учитывать переменный характер загрузки ресурсов и производительность каналов связи (по сравнению с локально-оптимальным алгоритмом (ДЛО) [9]). Также исследовано влияние выбора структуры локальных окрестностей диспетчеров на эффективность диспетчеризации.

1. Децентрализованная диспетчеризация параллельных задач в пространственно-распределённых ВС

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). Введем 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 сообщения размером m байт между подсистемами $i, j \in S$ ($[t(i, j, m)] = \text{с}$). Считается, что все подсистемы объединены сетью связи.

На каждой подсистеме присутствует локальная СУР и децентрализованный диспетчер. 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 диспетчеру j. Множество вершин j, смежных вершине i, образуют её локальную окрестность $L(i) = \{j \in S \mid (i, j) \in E\}$.

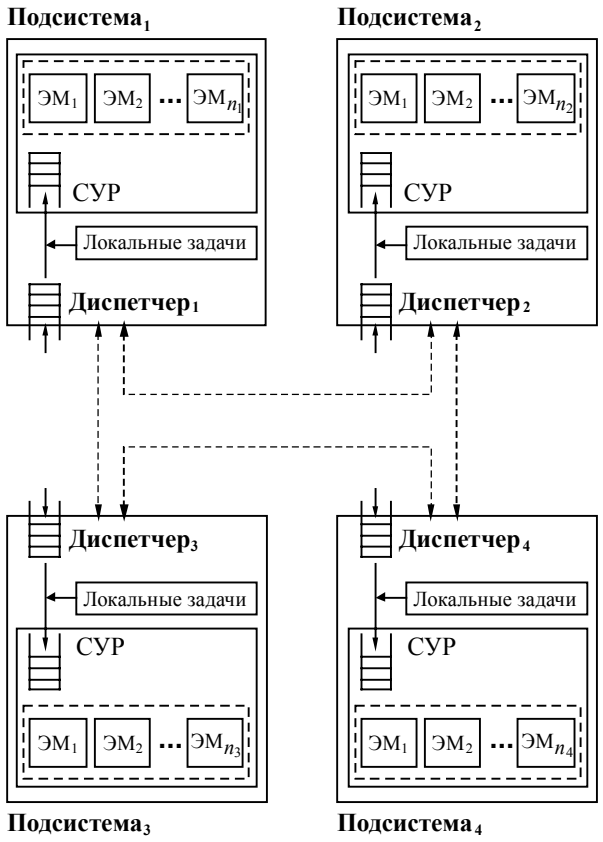

Рис. 1. Пример локальных окрестностей диспетчеров:
$H = 4$, $L(1) = \{2, 3\}$, $L(2) = \{1, 4\}$, $L(3) = \{1, 4\}$, $L(4) = \{2, 3\}$

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 $h_1, h_2, \ldots, h_k$, 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.

2. Децентрализованные алгоритмы диспетчеризации задач

Предложено четыре децентрализованных алгоритма диспетчеризации параллельных задач в пространственно-распределённых ВС. 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 этого строится множество допустимых подсистем $S(i) = \{j \mid n_j \geq r, j \in L(i) \cup \{i\}\}$, имеющих количество ЭМ не ниже требуемого.

2.1. Алгоритм локально-оптимальной диспетчеризации (ДЛО)

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.

2.2. Алгоритм диспетчеризации на основе репликации задач (ДР)

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 Δ 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'.

2.3. Алгоритм диспетчеризации на основе миграции задач (ДМ)

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 Δ 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 (ДРМ)

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. Программный пакет GBroker децентрализованной диспетчеризации задач

Все предложенные алгоритмы были реализованы в пакете GBroker [9] децентрализованной диспетчеризации параллельных задач в пространственно-распределённых ВС. Пакет разрабатывается Центром параллельных вычислительных технологий ФГОБУ ВПО «Сибирский государственный университет телекоммуникаций и информатики» (ЦПВТ ФГОБУ ВПО «СибГУТИ»), совместно с Лабораторией вычислительных систем Института физики полупроводников им. А.В. Ржанова СО РАН (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. Экспериментальное исследование алгоритмов

Исследование созданных алгоритмов проводилось на пространственно-распределённой мультикластерной ВС, созданной ЦПВТ ФГОБУ ВПО «СибГУТИ» совместно с лабораторией вычислительных систем ИФП СО РАН (рис. 2). 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 расчета графических сцен. Входные данные для тестовых задач размещались на сегменте Xeon80; на эту же подсистему доставлялись результаты выполнения программ.

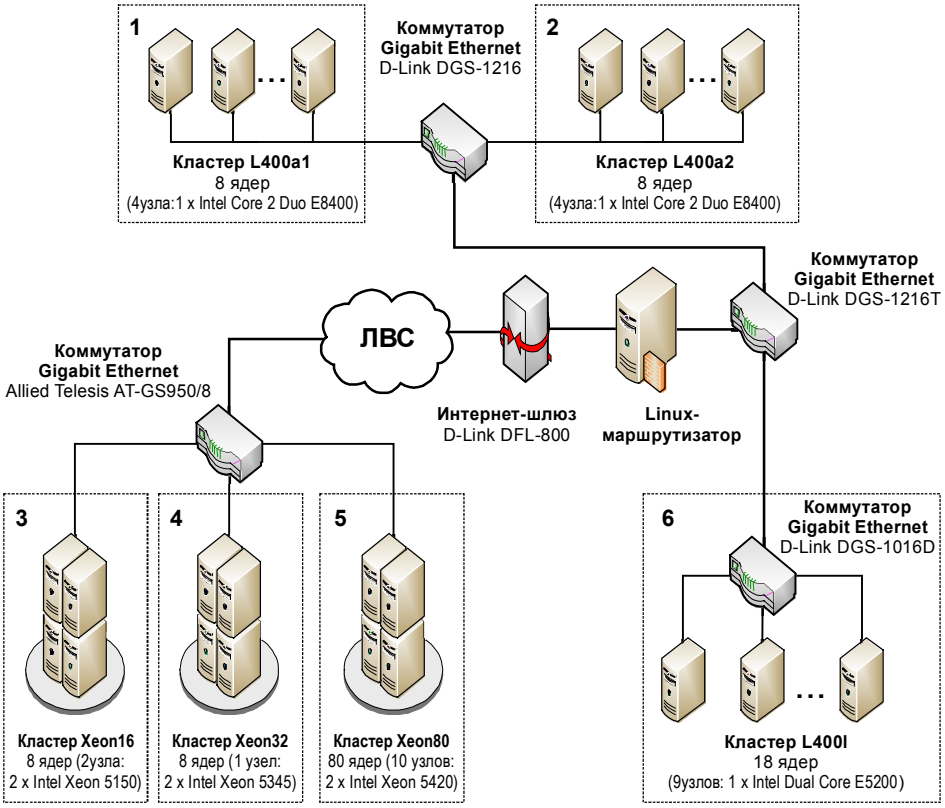

Рис. 2. Тестовая конфигурация мультикластерной ВС ($H = 6$, $N = 130$)

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 $k \in \{1, 2, \ldots, M\}$ 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:

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

4.1. Сравнительный анализ алгоритмов

На рис. 3 и 4 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.

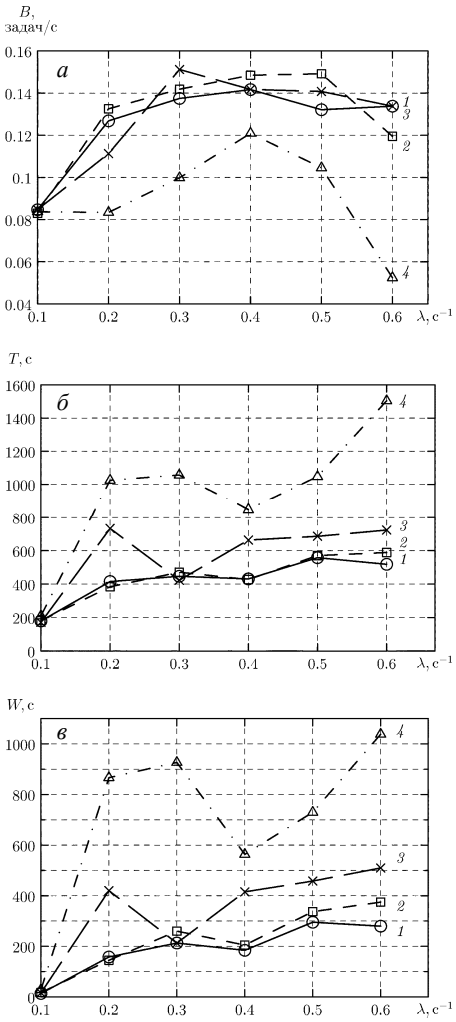

Рис. 3. Сравнение эффективности алгоритмов ДЛО и ДР: *1* – Алгоритм ДЛО; *2* – Алгоритм ДР, $m = 2$; *3* – Алгоритм ДР, $m = 3$; *4* – Алгоритм ДР, $m = 6$

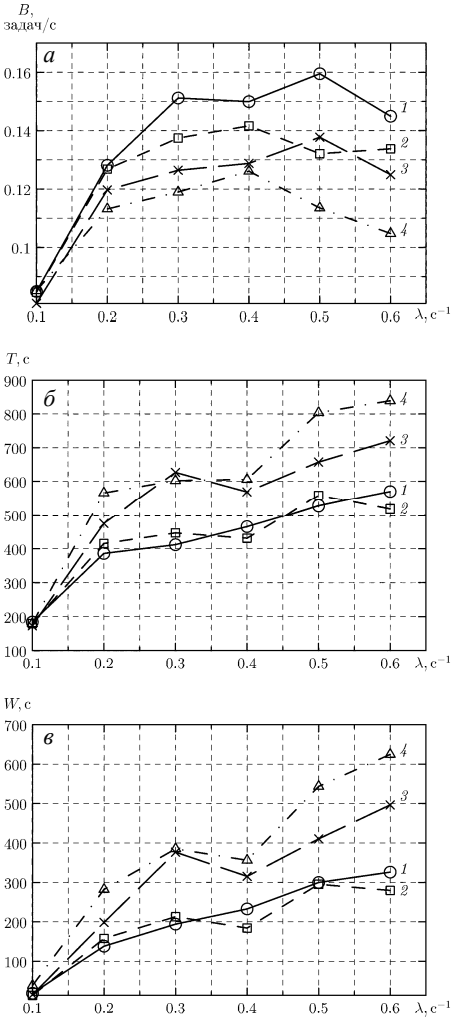

Рис. 4. Сравнение эффективности алгоритмов ДЛО, ДМ и ДРМ: *1* – Алгоритм ДМ; *2* – Алгоритм ДЛО; *3* – Алгоритм ДРМ, $m = 2$; *4* – Алгоритм ДРМ, $m = 3$

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. 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.

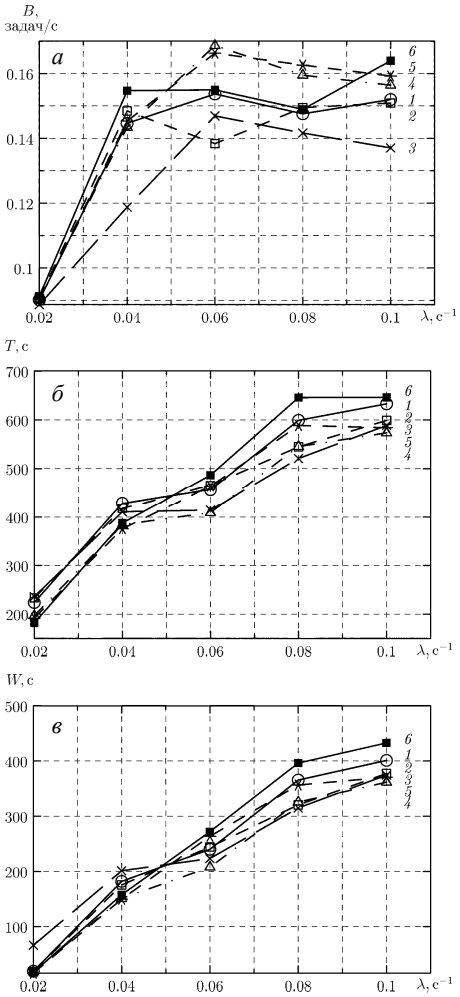

Рис. 5. Влияние структур локальных окрестностей диспетчеров на эффективность диспетчеризации: *1* – полный граф; *2* – звезда; *3* – кольцо; *4* – решетка; *5* – 2*D*-тор; *6* – D_2-граф

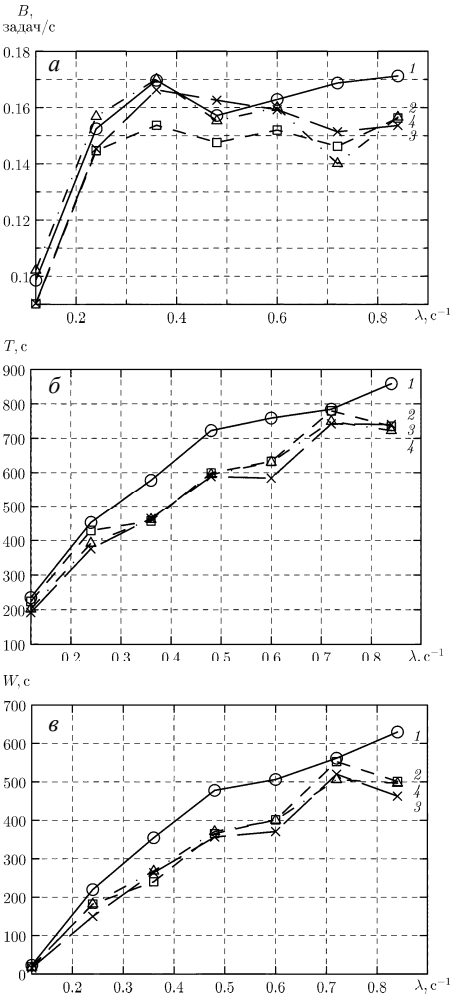

Рис. 6. Сравнение эффективности диспетчеров GBroker и GridWay: *1* – GBroker; *2* – GBroker, полный граф; *3* – GBroker, 2*D*-тор; *4* – GridWay

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.

4.2. Выбор структуры локальных окрестностей диспетчеров

Выполнено исследование влияния структуры локальных окрестностей диспетчеров на эффективность диспетчеризации. В эксперименте на вход всех диспетче-

ров одновременно поступали потоки из $M = 50$ задач. Рассматривались локальные окрестности в виде полных графов, колец, решёток, звёзд, 2D-торов и D_2-графов [1].

На рис. 5 показано влияние структуры локальных окрестностей диспетчеров на эффективность обслуживания потоков задач алгоритмом ДМ. Высокие значения пропускной способности, помимо полносвязной структуры, были получены для конфигураций на основе 2D-тора, решётки и D_2-графа, при этом для двух последних были достигнуты наибольшие значения. Использование неполносвязных структур при формировании локальных окрестностей диспетчеров не приводит к значительному снижению показателей эффективности диспетчеризации. Такие структуры могут быть образованы при отсутствии прямых линий связи между отдельными подсистемами, например в случае отказов некоторых подсистем. В качестве структур локальных окрестностей могут быть рекомендованы торы или D_n-графы, которые обладают малым (средним) диаметром.

4.3. 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 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 2D-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 с GridWay и незначительно возрастают при централизованном обслуживании.

Заключение

По сравнению с централизованным подходом, предложенные алгоритмы децентрализованной диспетчеризации существенно снижают сложность поиска ресурсов и обеспечивает живучесть пространственно-распределённых ВС. По сравнению с централизованным диспетчером GridWay была достигнута более высокая пропускная способность (с использованием алгоритма ДМ), при сопоставимых значениях среднего времени обслуживания задачи.

Механизм миграции, реализованный в алгоритмах ДМ, ДРМ, может использоваться для повышения пропускной способности системы (по отношению к ДЛО). При репликации задач на несколько подсистем (алгоритмы ДР, ДРМ) возрастают накладные расходы на доставку данных, что приводит к увеличению времени обслуживания задач.

Созданный пакет GBroker является свободно распространяемым. Для организации децентрализованной диспетчеризации задач в пространственно-распределённой ВС достаточно установить пакет GBroker на всех подсистемах и выполнить конфигурацию диспетчеров в соответствии с выработанными рекомендациями. При формировании локальных окрестностей диспетчеров рекомендуется использовать структуры малого диаметра.

ЛИТЕРАТУРА

1. *Хорошевский В.Г.* Распределённые вычислительные системы с программируемой структурой // Вестник СибГУТИ. 2010. № 2 (10). С. 3–41.
2. *Huedo E.*, *Montero R.*, *Llorente I.* A framework for adaptive execution on grids // Software – Practice and Experience (SPE). 2004. V. 34 P. 631–651.
3. *Berman F.*, *Wolski R.*, *Casanova H.* Adaptive computing on the grid using AppLeS // IEEE Trans. on Parallel and Distributed Systems. 2003. V. 14. No. 4. P. 369–382.
4. *Cooper K.*, *Dasgupta A.*, *Kennedy K.* New grid scheduling and rescheduling methods in the GrADS project // In Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04). 2004. P. 199–206.
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 International Conference on High Performance Computing in Asia-Pacific Region. 2000. P. 283–289.
6. *Frey J.*, *Tannenbaum T.*, *Livny M.*, *et al.* Condor-G: A computation management agent for multi-institutional grids // Cluster Computing. 2001. V. 5. P. 237–246.
7. *Корнеев В.В.* Архитектура вычислительных систем с программируемой структурой. Новосибирск: Наука, 1985. 164 с.
8. *Монахов О.Г.*, *Монахова Э.А.* Параллельные системы с распределённой памятью: управление ресурсами и заданиями. – 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 // Вестник Томского государственного университета. Управление, вычислительная техника и информатика. 2011. № 3 (16). С. 78–85.

Курносов Михаил Георгиевич
Пазников Алексей Александрович
Сибирский государственный университет
телекоммуникаций и информатики,
E-mail: mkurnosov@gmail.com, apaznikov@gmail.com

Поступила в редакцию 3 июня 2011 г.

Kurnosov Mikhail G., *Paznikov Alexey A.* (Siberian State University of Telecommunications and Information Sciences). **Decentralized scheduling algorithms of geographically-distributed computer systems.**

Keywords: task scheduling, meta-scheduling, geographically-distributed computer systems, GRID-systems.

Locally optimal algorithm and algorithms based on job migration, job replication and both migration and replication of decentralized scheduling of parallel programs in geographically-distributed computer systems are proposed in this paper. Software of parallel jobs decentralized scheduling is considered. Job migration and job replication are indented to help to consider dynamically changed structure and resource workload. Computational complexity of proposed algorithms doesn't depend on number of subsystems, because the search implements within the scheduler's local neighborhood. This provides algorithms applicability in large-scale geographically-distributed CS.

Algorithms have been realized and included into GBroker software suite of parallel programs decentralized scheduling in geographically-distributed multicluster and GRID-systems. Modeling of developed algorithms and software tools on the active multicluster system has shown high effectiveness of job migration. Investigation of local neighborhood structures has shown that the using of non-fully connected structures of small (mean) diameter doesn't result in significant decrease of the system performance. An experimental comparison of developed packet GBroker with centralized scheduler GridWay has shown that mean service time of job flows with centralized and decentralized scheduling is comparable. Using of the algorithm with migration has allowed to exceed the bandwidth of this centralized scheduling system.