Децентрализованные алгоритмы управления ресурсами распределенных вычислительных и GRID-систем*

М.Г. Курносов
Институт физики полупроводников им. А.В. Ржанова СО РАН
e-mail: mkurnosov@isp.nsc.ru

А.А. Пазников
Сибирский государственный университет телекоммуникаций и информатики

Предлагаются алгоритмы децентрализованной диспетчеризации параллельных программ в пространственно-распределённых вычислительных и GRID-системах. Исследуется эффективность механизмов миграции задач между очередям вычислительных подсистем. Проводится сравнение различных структур логических связей между диспетчерами подсистем. Представлены результаты обслуживания потоков реальных MPI-программ на мультикластерной системе средствами разработанного пакета GBroker и диспетчером GridWay.

1. Введение

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

К значимым проблемам организации функционирования пространственно-распределенных ВС относится диспетчеризация параллельных программ. Для каждой программы, из поступивших в ВС, требуется определить вычислительные ресурсы (подсистемы) для ее выполнения. В средствах диспетчеризации необходимо учитывать возможность изменения состава и загрузки ВС с течением времени.

Централизованные средства диспетчеризации программ имеют существенный недостаток: отказ управляющего узла ВС может привести к неработоспособности всей системы. Кроме того, в случае применения таких средств в большемасштабных ВС возрастают временные затраты на поиск требуемых ресурсов. Поэтому актуальной является задача разработки децентрализованных моделей, алгоритмов и программного обеспечения для диспетчеризации параллельных задач в распределенных ВС.

При децентрализованной диспетчеризации в системе должен функционировать коллектив диспетчеров, осуществляющий выбор необходимых ресурсов для реализации программ. Это позволяет достичь и живучести большемасштабных ВС, то есть способности систем продолжать работу при отказах отдельных компонентов и подсистем.

Существует несколько пакетов централизованной диспетчеризации параллельных

*Работа выполнена при поддержке РФФИ (гранты № 10-07-00157, 11-07-00105) и Совета по грантам Президента РФ для поддержки ведущих научных школ (грант НШ-5176.2010.9).

программ в пространственно-распределенных ВС: GridWay, AppLeS, GrADS, Nimrod/G, Condor-G, WMS и др. GridWay [3] - наиболее распространённый из них; в нем преследуется цель минимизации времени обслуживания задач; поддерживается миграция задач между подсистемами; имеется возможность решения задач, представленных в виде ациклических орграфов. Пакеты AppLeS [4] и GrADS [5] предоставляют инструментарий для решения задач в GRID-системах. В AppLeS [4] диспетчеризация выполняется на уровне самого приложения, что требует от пользователя особых знаний и снижает универсальность применения данного пакета. GrADS [5] поддерживает миграцию задач и допускает указание зависимостей между задачами. Nimrod/G [6] на основе экономических моделей обеспечивает равновесие между условными поставщиками и потребителями ресурсов. Nimrod/G ориентирован на решение задач с изменяемыми параметрами (parameter sweep application). В Condor-G [7] для поиска ресурсов используются механизмы ClassAd и Matchmaking; компонент DAGMan используется для решения задач, представленных ориентированными ациклическими графами.

В работе предлагаются децентрализованные алгоритмы и программное обеспечение диспетчеризации параллельных программ в пространственно-распределённых вычислительных и GRID-системах.

2. Децентрализованные алгоритмы диспетчеризации

Пусть имеется пространственно-распределенная ВС, состоящая из H подсистем; N – суммарное количество элементарных машин (ЭМ) в подсистемах. Для подсистемы $i \in S = \{1, 2, \ldots, H\}$ введем обозначения: n_i – количество ЭМ, входящих в состав подсистемы, c_i – число свободных в данный момент процессоров, q_i – число задач в очереди в состоянии ожидания, p_i – число задач в состоянии выполнения. Пусть также $b_{ij} = b(i, j, m)$ – пропускная способность канала связи между подсистемами $i, j \in S$ при передаче сообщений размером m байт ($[b(i, j, m)]$ = байт/с).

В каждой подсистеме присутствует диспетчер, который поддерживает 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. Наличие дуги $(i, j) \in E$ в графе означает, что диспетчер i может отправлять ресурсные запросы (задачи из своей очереди) диспетчеру j. Множество всех вершин j, смежных вершине i, образуют её локальную окрестность $L(i) = j \in S : (i, j) \in E$.

Пользователь направляет задачу и запрос на выделение ресурсов одному из диспетчеров распределенной ВС. Диспетчер (в соответствии с реализованными в нем алгоритмами) выполняет поиск (суб)оптимальной подсистемы j^* (или подсистем $j_1^*, j_2^*, \ldots, j_m^*$) из своей локальной окрестности для решения задачи пользователя.

Считаем, что задача характеризуется рангом r (количеством параллельных ветвей) и размерами $z_1, z_2, \ldots, z_k$ исполняемых файлов и входных данных, которые находятся на подсистемах $h_1, h_2, \ldots, h_k$, $h_i \in S$ ($[z_i]$=байт).

Авторами предложены четыре алгоритма диспетчеризации параллельных программ в пространственно-распределенных вычислительных системах: простейший алгоритм (SS – Simple Scheduling), алгоритм на основе назначения задачи в очередь нескольких подсистем (RS – Replication-based Scheduling), алгоритм на основе миграциии задач из очереди (MS – Migration-based Scheduling) и алгоритм, основанный на комбинации двух подходов (RMS – Replication and Migration-based Scheduling).

Рассматривается функционирование ВС в режиме обслуживания потока задач. Обозначим события, которые могут при этом происходить в системе: *job_ submit*, *try_ migrate*, *job_ run*. Событие *job_ submit* возникает при поступлении задачи на вход диспетчера, *job_ run* - в момент начала выполнения параллельной программы, *try_ migrate* - при инициализации механизма поиска новой подсистемы для задачи в очереди. Рассмотрим обработчики событий, необходимые для реализации в диспетчере предложенных алгоритмов. В SS и MS используется простейший вариант процедуры обработки события *job_ submit*.

Алгоритм ***job_ submit***
1 Получить информацию о системах локальной окрестности $L(s) \cup s$.
2 Вычислить значения целевой функции $f(i)$ для всех подсистем.
3 Отправить задачу на подсистему d с наименьшим значением целевой функции.

В результате её выполнения задача ставится в очередь подсистемы $d \in L(s) \cup s$.

Для сравнения времени обслуживания задач на подсистемах исплользуется подход, описанный в [8]. Целевая функция при этом выглядит следующим образом:

$$f(i) = \begin{cases} \frac{t_t}{t_{max}}, & \text{если } c_i \geq r \text{ и } q_i = 0, \\ \frac{t_t}{t_{max}} + \frac{c_i^{-1}}{c_{max}^{-1}} + \frac{\lambda_i}{\lambda_{max}}, & \text{иначе.} \end{cases} \quad (1)$$

где $\lambda_i = q_i/n_i$, t_i - оценка времени доставки файлов задачи до подсистемы i, $t_i = \sum_{i=1}^{k} z_i/b(i, h_i, z_i)$.

Алгоритмы диспетчеризации RS и RMS предполагают на третьем шаге алгоритма *job_ submit* одновременную постановку задачи в очередь m подсистем с наименьшими значениями целевой функции. Как только на некоторой подсистеме программа переходит в состояние выполнения (событие *job_ run*), на остальных m–1 подсистемах производится удаление задачи из очереди. Описанная схема позволяет сократить среднее время ожидания задачи в очереди, а, следовательно, и среднее время обслуживания.

Событие *try_ migrate* (алгоритмы MS, RMS) наступает с интервалом времени Δ для каждой задачи в состоянии ожидания. В результате выполняется поиск новой подсистемы. Периодический поиск подсистем позволяет учитывать динамически изменяющуюся загрузку ресурсов пространственно-распределенных ВС.

Алгоритм ***try_ migrate***
1 Получить информацию о системах локальной окрестности $L(s) \cup s$.
2 Вычислить значения целевой функции для всех подсистем.
3 Если для одной из подсистем значение целевой функции превосходит текущее более чем на χ, удалить задачу из текущей очереди и отправить на новую подсистему.

Алгоритмы SS, MS, RS и RMS реализованы и включены в состав пакета GBroker децентрализованной диспетчеризации параллельных программ в пространственно-распределенных мультикластерных ВС.

3. Моделирование и результаты экспериментов

В Центре параллельных вычислительных технологий ГОУ ВПО “Сибирский государственный университет телекоммуникаций и информатики” (ЦПВТ ГОУ ВПО

“СибГУТИ”) и Лаборатории вычислительных систем Института физики полупроводников им. А.В. Ржанова СО РАН (ИФП СО РАН) создан и развивается программный пакет GBroker [2] децентрализованной диспетчеризации параллельных программ в пространственно-распределенных ВС. Пакет интегрирован с сервисами GlobusToolkit.

В пакет (рис. 1) входят диспетчер gbroker, клиентское приложение gclient, системы мониторинга dcsmon – состояния вычислительных ресурсов и netmon – производительности каналов связи между подсистемами.

Созданные алгоритмы диспетчеризации исследованы на мультикластерной ВС, со-

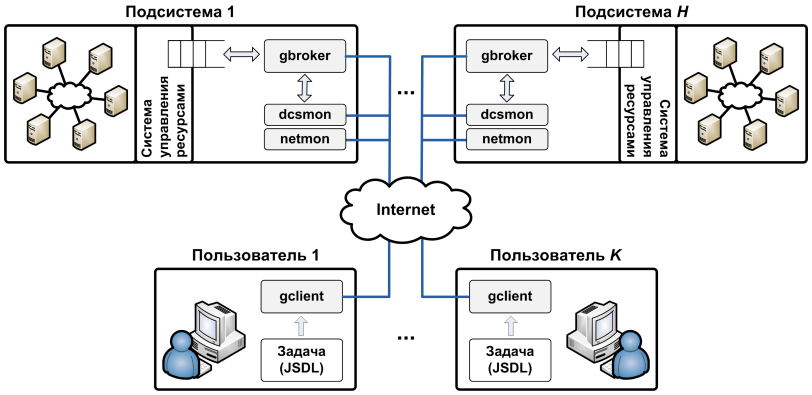

Рис. 1. Функциональная структура пакета GBroker

зданной ЦПВТ ГОУ ВПО “СибГУТИ” совместно с Лабораторией вычислительных систем ИФП СО РАН. В экспериментах участвовали 6 сегментов системы: кластер Xeon80 (10 узлов, 80 процессорных ядер), кластер Xeon16 (2 узла, 8 процессорных ядер), кластер Xeon32 (1 узел, 8 процессорных ядер), кластер L400l (9 узлов, 18 процессорных ядер), кластер L400a1 (4 узла, 8 процессорных ядер), кластер L400a2 (4 узла, 8 процессорных ядер). Сети связи между сегментами – Gigabit Ethernet и Fast Ethernet.

В качестве тестовых задач использовались MPI-программы, входящие в состав пакета SPEC MPI 2007: WRF3, POP2, LAMMPS, RAxML, Tachyon. Входные данные находились на сегменте Xeon80. В ходе моделирования генерировались простейшие потоки параллельных программ интенсивности λ ($[\lambda] = \text{с}^{-1}$). Тестовые задачи выбирались случайным образом. Ранг r параллельной программы выбирался из набора $\{1, 2, 4, 8\}$.

В качестве показателей эффективности алгоритмов диспетчеризации использовались *пропускная способность B системы*, и *среднее время T обслуживания задачи*.

На рис. 2 приведены графики показателей эффективности диспетчеризации для алгоритмов SS и RS. RS может быть использован для высокоприоритетных задач, при обслуживании потоков малой интенсивности и для программ с небольшими размерами входных данных.

На рис. 3 представлено сравнение экспериментальных данных, полученных для алгоритмов MS, RMS и SS. Интервал поиска подсистемы равнялся $\Delta = 30$ с, критерий миграции $\chi = 0,2$. Наименьшее среднее время обслуживания задач было показано для алгоритмов SS и MS, наибольшая пропускная способность системы - для алгоритма с миграцией. Использование алгоритма RMS также в случаях, аналогичным RS.

В ходе исследования структур логических связей на все подсистемы одновременно поступали одинаковые потоки. В качестве алгоритма диспетчеризации использовался MS. Из графиков (рис. 4) видно, что высокой пропускной способностью, помимо пол-

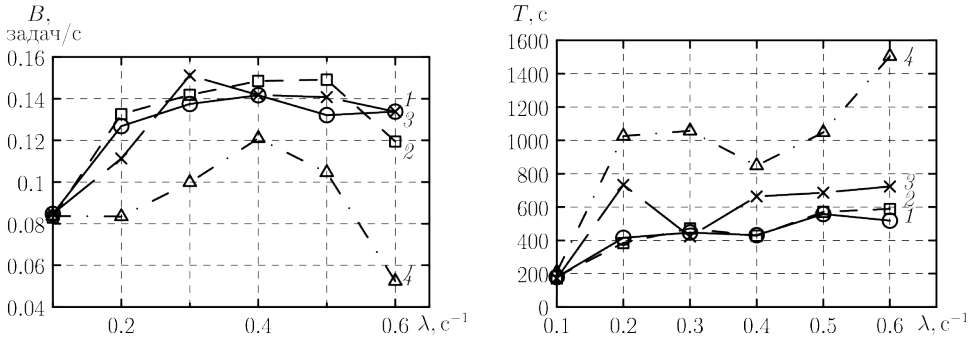

Рис. 2. Результаты моделирования алгоритмов SS, RS
1 - *MS*; 2 - *RS*, $m = 3$; 3 - *RMS*, $m = 3$; 4 - *RMS*, $m = 6$

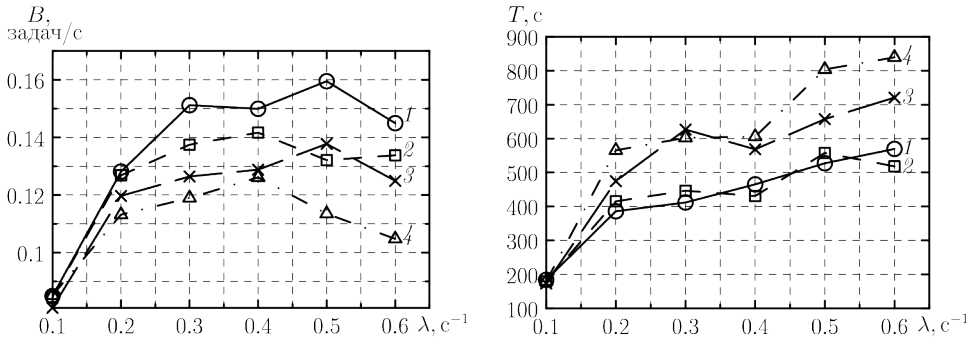

Рис. 3. Результаты моделирования алгоритмов MS, RMS
1 - *MS*; 2 - *SS*; 3 - *RMS*, $m = 2$; 4 - *RMS*, $m = 3$

носвязной структуры, обладают системы на основе $2D$-тора, решётки и D_2-графа.

Выполнено сравнение эффективности обслуживания потоков задач централизо-

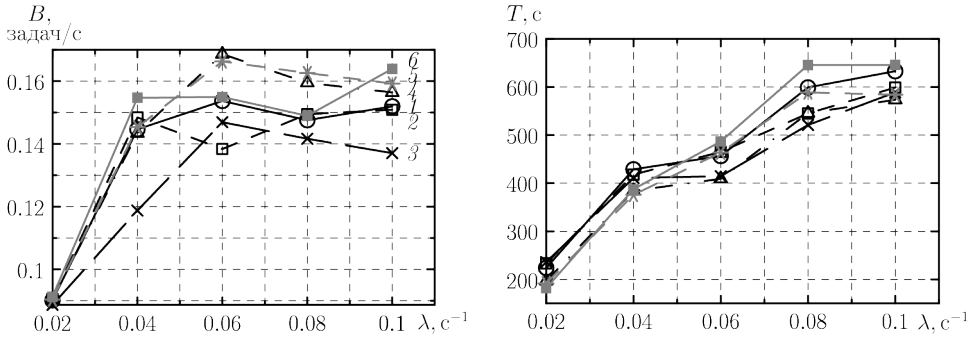

Рис. 4. Сравнение различных структур логических связей коллектива диспетчеров gbroker
1 - полный граф; 2 - звезда; 3 - кольцо; 4 - решетка; 5 - тор; 6 - D_2-граф

ванным диспетчером GridWay и средствами пакета GBroker. Для диспетчера GBroker моделировалось централизованное и распределенное обслуживание задач. В качестве алгоритма использовался MS. Видно (рис. 5), что при распределенном обслуживании потоков задач пропускная способность системы начинает превосходить GridWay.

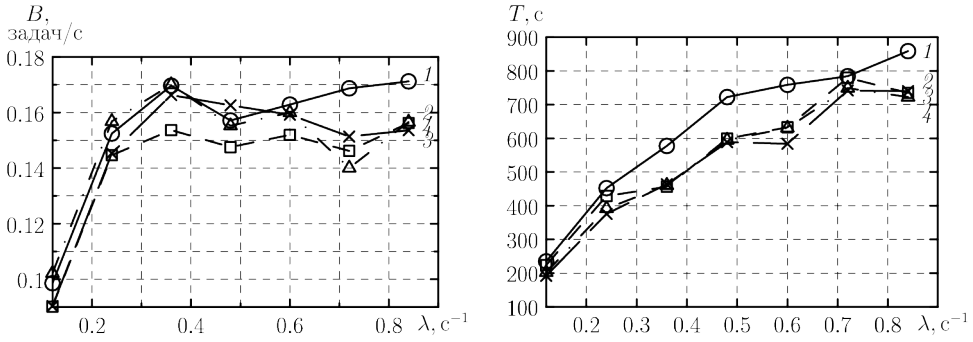

Рис. 5. Сравнение диспетчеров GBroker и GridWay
1 - GBroker; 2 - GBroker, полный граф; 3 - GBroker, тор; 4 - GridWay

4. Заключение

Результаты исследования созданных алгоритмов децентрализованной диспетчеризации показали их эффективность. В сравнении с диспетчером GridWay, разработанные средства обеспечивают более высокую пропускную способность при распределенном обслуживании потоков задач. В качестве логических структур коллектива диспетчеров целесообразно использовать неполносвязные графы, обладающие малым (средним) диаметром. При формировании локальных окрестностей из небольшого количества подсистем предложенные алгоритмы позволяют эффективно обслуживать потоки задач независимо от числа подсистем пространственно-распределенной ВС.

Список литературы

[1] Хорошевский В.Г. Распределенные вычислительные системы с программируемой структурой // Вестник СибГУТИ. 2010. N 2(10). С. 3–41.

[2] Курносов М.Г., Пазников А.А. Децентрализованное обслуживание потоков параллельных задач в пространственно-распределенных вычислительных системах // Вестник СибГУТИ. 2010. N 2(10). С. 79–86.

[3] Huedo E., Montero R., Llorente I. A Framework for Adaptive Execution on Grids // Software – Practice and Experience (SPE). 2004. V. 34. P. 631-651.

[4] Berman F., Wolski R., Casanova H. Adaptive Computing on the Grid Using AppLeS // In IEEE Trans. on Parallel and Distributed Systems (TPDS). 2003. V. 14, No. 4. P. 369–382.

[5] Cooper K., Dasgupta A., Kennedy A., Koelbel C. 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.

[6] Buyya R., Abramson D., Giddy J. Nimrod/G: An architecture for a resource management and scheduling system in a global computational Grid // Proceedings of the 4th International Conference on High Performance Computing in Asia-Pacific Region. 2000. P. 283–289.

[7] Frey J., Tannenbaum T., Livny M., Foster I. Condor-G: A Computation Management Agent for Multi-Institutional Grids // Cluster Computing. 2001. V. 5. P. 237–246.

[8] AL-Khateeb A., Abdullah R., Rashid N. Job Type Approach for Deciding Job Scheduling in Grid Computing Systems // Journal of Computer Science. 2009. V. 5. P. 745–750.