Децентрализованная диспетчеризация параллельных программ в распределённых вычислительных системах

М.Г. Курносов, А.А. Пазников

Сибирский государственный университет телекоммуникаций и информатики
Новосибирск

Введение

На сегодняшний день для решения сложных задач науки и техники широкое распространение получили пространственно-распределенные вычислительные системы (ВС) (в частности, GRID-системы и мультикластерные ВС). Такие системы представляют собой макроколлективы пространственно рассредоточенных ВС (вычислительных подсистем), взаимодействующих между собой через локальные и глобальные сети связи (включая всемирную сеть Internet) [1].

При организации функционирования распределенных ВС возникает задача диспетчеризации параллельных программ, поступающих в систему. Для каждой программы необходимо определить, на каких вычислительных ресурсах каких подсистем она будет выполняться. В средствах диспетчеризации должны учитываться динамичность состава систем и переменная загрузка их вычислительных ресурсов.

Централизованные системы диспетчеризации имеют существенный недостаток: отказ управляющего узла может привести к неработоспособности всей ВС. Актуальной является задача разработки децентрализованных моделей, алгоритмов и системного программного обеспечения диспетчеризации параллельных задач в распределенных ВС.

При децентрализованной схеме диспетчеризации задач в системе функционирует коллектив диспетчеров, совместно принимающий решение о выделении ресурсов для программ. Это позволяет достичь живучести ВС, то есть ее способности продолжать работу при отказах отдельных подсистем.

На сегодняшний день существует несколько пакетов централизованной диспетчеризации параллельных программ в пространственно-распределенных системах: GridWay, CSF, Nimrod/G, Condor-G, GrADS, AppLeS, DIRAC, WMS и др. В GridWay [2, 3] преследуется цель минимизации времени обслуживания задачи; реализован механизм миграции задач между подсистемами. Среда CSF [4] предоставляет средства резервирования ресурсов, на основе алгоритма Round-Robin и созданных пользователем алгоритмов. В основу Nimrod/G [5] положена экономическая модель, целью которой является поиск равновесного состояния между поставщиками и потребителями ресурсов посредством механизма аукциона. В Condor-G [6] предполагается, что пользователь самостоятельно выбирает подсистему из списка доступных; при этом поддерживается создание пользовательских диспетчеров. Функционирование WMS [7] основывается на применении двух политик: в одном случае задача отправляется на подсистему как можно быстрее, в другом – задача ожидает в очереди до тех пор, пока ресурс не станет доступным. В DIRAC [8] на подсистемах установлены агенты, которые запрашивают задачи из глобальных очередей при освобождении ресурсов.

В данной работе рассматривается децентрализованный алгоритм диспетчеризации параллельных программ в распределенных ВС. Рассмотрим формальную постановку задачи.

Постановка задачи

Пусть пространственно-распределенная ВС состоит из H вычислительных подсистем и включает N элементарных машин (ЭМ). Через n_i обозначим количество ЭМ, входящих в состав подсистемы $i \in S = \{1,2,...,H\}$.

Известны значения показателей производительности каналов связи между подсистемами. Пусть $b_{pq} = b(p,q,m)$ – пропускная способность канала связи между подсистемами $p, q \in S$ при передаче сообщений размером m байт ([$b(p,q,m)$] = байт/с).

Считаем, что на каждой подсистеме присутствует очередь параллельных программ. Каждая программа характеризуется рангом r – количеством параллельных ветвей, ожидаемым временем t выполнения программы (walltime) и размером z файла параллельной программы и данных ($[z] = $ байт).

Для каждой поступающей в систему задачи требуется определить подсистему $s \in S$, доставляющую минимум времени её обслуживания.

Предложен децентрализованный алгоритм диспетчеризации параллельных программ.

Алгоритм децентрализованной диспетчеризации

Пусть на каждой из подсистем $i \in S$ функционирует диспетчер. Коллектив диспетчеров представлен в виде ориентированного графа $G = (V, E)$, в котором вершинам $V = \{1, 2, \ldots, H\}$ соответствуют диспетчеры, а ребрам – логические связи между ними. Наличие ребра $(i, j) \in E$ означает, что диспетчер i может отправлять ресурсные запросы диспетчеру j. Множество вершин, смежных из i, образует локальную окрестность v_i диспетчера i.

При поступлении программы в очередь подсистемы s, её диспетчер опрашивает диспетчеры $j \in v_s$ из своей локальной окрестности и получает от них информацию о текущем количестве задач в их очередях и оценки времени t_j, через которое программа может быть запущена на их ресурсах. У системы мониторинга запрашивается текущие значения пропускных способностей $b_{sj} = b(s, j, z)$ каналов связи до подсистем.

После чего выбирается подсистема d, обеспечивающая минимальное значение времени T обслуживания параллельной программы. Время обслуживания задачи включает времени доставки файлов параллельной программы и данных до выделенной подсистемы, время ожидания в очереди подсистемы и время выполнения.

$$d = \arg\min_{j \in v_s} \{T_{sj}\} \tag{1}$$

$$T_{sj} = z / b(s, j, z) + t_j + t\,, \tag{2}$$

Программный пакет децентрализованной диспетчеризации GBroker

В Центре параллельных вычислительных технологий ГОУ ВПО «Сибирского государственного университета телекоммуникаций и информатики» (ЦПВТ ГОУ ВПО “СибГУТИ”) ведется разработка программного пакета GBroker [9] децентрализованной диспетчеризации задач в пространственно-распределенных ВС. Пакет разрабатывается на языке ANSI C для операционной системы GNU/Linux.

В пакет (рис. 1) входит диспетчер gbroker, клиентское приложение gclient и средство мониторинга производительности каналов связи netmon.

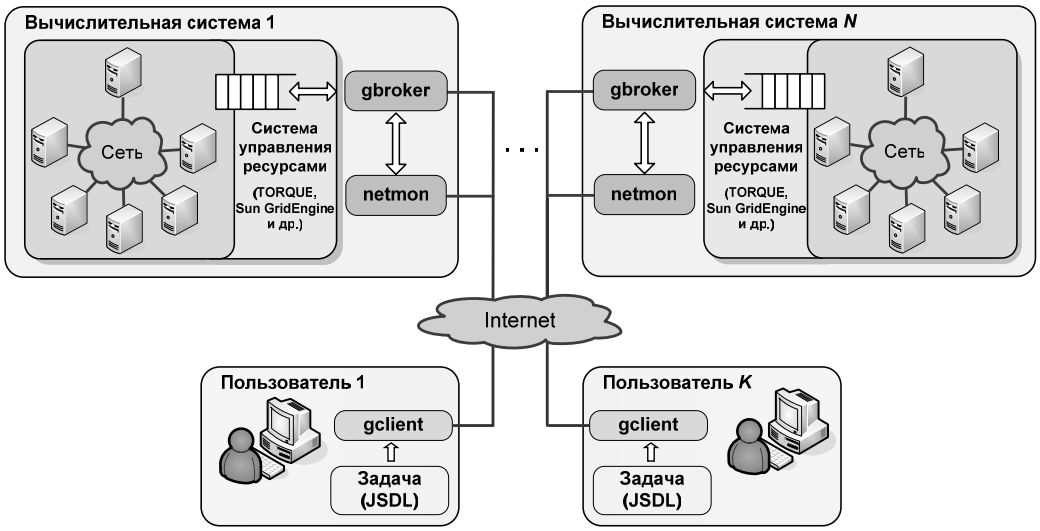

Рисунок 1. Функциональная структура пакета GBroker.

Модуль gbroker устанавливается на каждой из подсистем и обеспечивает на основе расширяемой архитектуры интерфейс с локальной системой управления ресурсами (TORQUE). Модуль netmon устанавливается вместе с gbroker на подсистемах. Взаимодействуя друг с другом, сервисы netmon собирают динамическую информацию производительности каналов связи между подсистемами. Модуль gclient реализует интерфейс между пользователем и системой.

Администратор настраивает локальные окрестности диспетчеров gbroker, указывая, какие диспетчеры с какими могут обмениваться задачами, и настраивает сервис netmon.

Пользователь формирует задание, состоящее из параллельной MPI-программы и паспорта на языке JSDL, и отправляет его средствами gclient любому из диспетчеров gbroker. Диспетчер в соответствии с описанным алгоритмом выбирает подсистему и выполняет на ней программу.

Результаты экспериментов

На мультикластерной вычислительной системе ЦПВТ ГОУ ВПО «СибГУТИ» проведено исследование разработанного алгоритма диспетчеризации и программного средства. Использовались кластеры следующих конфигураций:

– Xeon 16: 4 узла - 2 x Intel Xeon 5150, сети Gigabit/Fast Ethernet;

– Xeon 32: 4 узла - 2 x Intel Xeon E5345, сети Gigabit/Fast Ethernet;

– Xeon 80: 10 узлов - 2 x Intel Xeon E5420 сети Gigabit/Fast Ethernet.

В качестве тестовых задач использовались MPI-программы из пакета NPB (NAS Parallel Benchmarks), а также программы, реализующие параллельные версии различных численных методов.

Моделирование проводилось следующим образом. На выбранную подсистему поступал стационарный поток из M параллельных задач. Тестовые задачи выбирались случайным образом. Ранг r_i параллельной программы подчинен пуассоновскому закону со значениями математического ожидания $\{0{,}25r_{\max}, 0{,}5r_{\max}, 0{,}75r_{\max}, r_{\max}\}$, где $r_{\max} = 80$ – максимальное количество процессорных ядер среди вычислительных кластеров. Заявки на обслуживание поступали через интервалы времени t, экспоненциально распределенному с параметрами $\mu = \{0{,}1$ с; 0,2 с; 0,3 с; 0,4 с$\}$.

Пусть t_i^s – момент поступления i-й тестовой задачи на вход диспетчера, t_i^d – момент времени принятия решения о диспетчеризации, t_i^r – момент получения

пользователем результатов. Тогда $\tau = \max\limits_{i=1\ldots M} t_i^r - \min\limits_{i=1\ldots M} t_i^s$ – время обслуживания потока из M задач.

1) Среднее время обслуживания задачи $t^s = \frac{1}{M}\sum\limits_{i=1}^{M}(t_i^r - t_i^s)$;

2) Среднее время диспетчеризации $t^d = \frac{1}{M}\sum\limits_{i=1}^{M}(t_i^d - t_i^s)$;

3) Пропускная способность системы $B = M / \tau$.

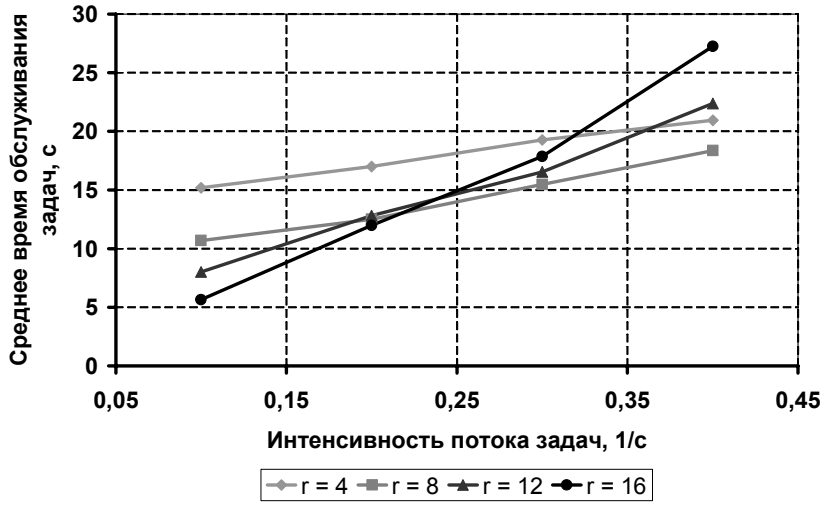

Рисунок 2. Зависимости среднего времени обслуживания задач от интенсивности λ потока задач.

Из графиков (рис. 2) видно, что среднее время обслуживания задач увеличивается с ростом интенсивности потока поступления. Это связано с образованием очередей на обслуживание в системе пакетной обработки заданий. Такое влияние интенсивности потоков особенно значительно при рангах параллельных программ, близких к размерам подсистем.

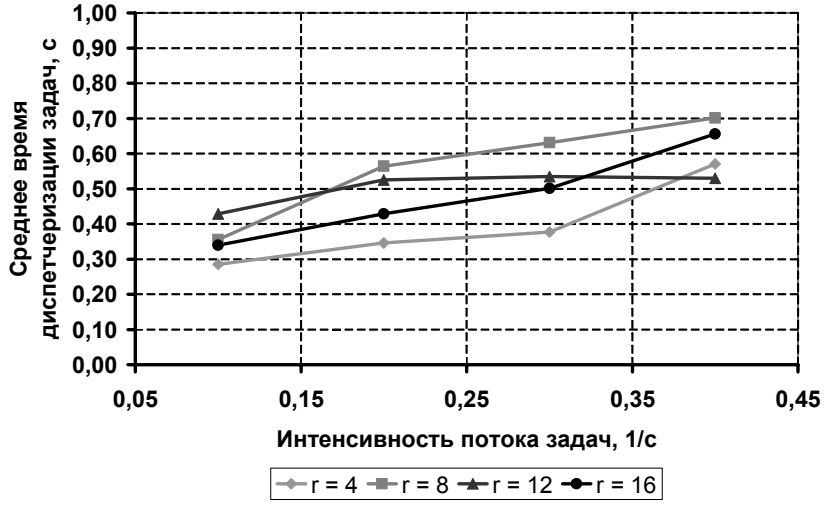

Рисунок 3. Зависимости среднего времени диспетчеризации задач от интенсивности λ потока задач.

По графикам (рис. 3) можно определить, что время диспетчеризации задач незначительно по сравнению с суммарным временем обслуживания и в среднем не превосходит одной секунды. Среднее время диспетчеризации задач не значительно изменяется как с ростом интенсивности потока поступления, так и с увеличением среднего ранга параллельных задач.

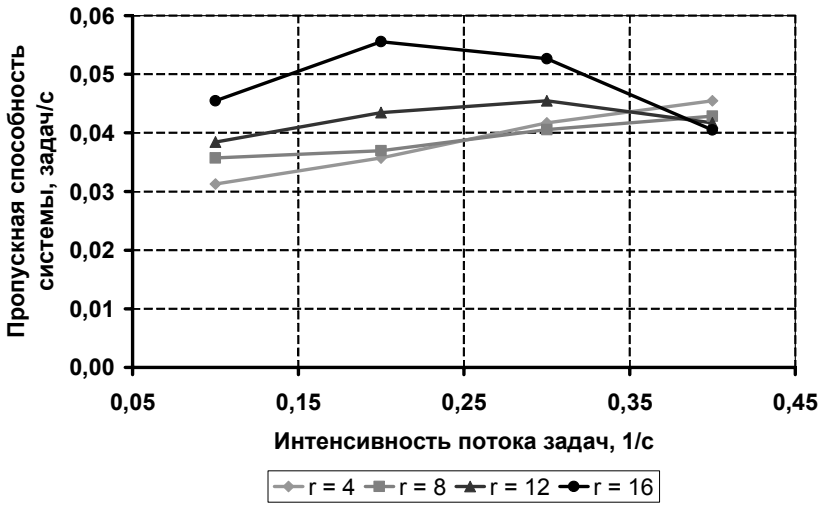

Рисунок 4. Зависимости пропускной способности системы от интенсивности λ потока задач.

Пропускная способность системы (рис. 4) растет при увеличении интенсивности потока поступления. При больших значениях интенсивности происходит снижение пропускной способности системы, что связано с образованием очередей на обслуживание в системе пакетной обработки. С увеличением среднего ранга это влияние интенсивности на пропускную способность системы усиливается.

Заключение

Среднее время обслуживания задач при децентрализованной диспетчеризации сопоставимо с централизованной диспетчеризацией. При этом обеспечивается отказоустойчивость в случае выхода из строя отдельных подсистем. Время диспетчеризации достаточно мало по сравнению со временем выполнения программ.

В дальнейшем планируется осуществить поддержку стандарта DRMAA для взаимодействия с системами пакетной обработки заданий, а также реализовать интеграцию с программным пакетом GridWay.

Работа выполнена при поддержке РФФИ (грант № 08-07-00018).

Литература

1. Хорошевский В. Г. Архитектура вычислительных систем: Учеб. Пособие для вузов. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2008.
2. R. Montero, E. Huedo, I. Llorente. Grid Resource Selection for Opportunistic Job Migration // 9th International Euro-Par Conference. – 2003. – V. 2790. – P. 366-373.
3. E. Huedo, R. Montero, I. Llorente. A Framework for Adaptive Execution on Grids // Software – Practice and Experience (SPE). – 2004. – V. 34 – P. 631-651
4. W. Xiaohui , D. Zhaohui , Y. Shutao. CSF4: A WSRF Compliant Meta-Scheduler // In Proc. of World Congress in Computer Science Computer Engineering, and Applied Computing. – 2006. – P. 61-67.
5. R. Buyya , D. Abramson , J. Giddy. 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.
6. J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids // Cluster Computing. – 2001. – V. 5. – P. 237-246.
7. P. Andreetto, S. Borgia, A. Dorigo. Practical approaches to grid workload and resource management in the EGEE project // In CHEP '04: Proceedings of the Conference on Computing in High Energy and Nuclear Physics. – 2004. – V. 2. – P. 899-902.
8. E. Caron, V. Garonne, A. Tsaregorodtsev. Evaluation of Meta-scheduler Architectures and Task Assignment Policies for High Throughput Computing // Technical report. Institut National de Recherche en Informatique et en Automatique. – 2005.
9. Пазников А.А. Средства децентрализованной диспетчеризации задач в распределенных вычислительных системах // Материалы XLVII Международной научной студенческой конференции «Студент и научно-технический прогресс». – Новосибирск: НГУ, 2009.– С. 210.