Моделирование параллельных вычислительных процессов в среде Грид на примере Intel Grid Programming Environment*

А.В. Дунаев, А.В. Ларченко, А.В. Бухановский

Интерпретация Грид как параллельной вычислительной архитектуры подразумевает развитие способов проектирования вычислительно эффективных алгоритмов, учитывающих специфические особенности распределенной среды, в первую очередь - стохастическую изменчивость параметров, их нестационарность и неоднородность. Предложен подход к исследованию производительности приложений в Грид. Рассматриваются аналитические приближения моделей времени выполнения, с целью определения меры изменчивости производительности проводится имитационное моделирование. Обсуждается возможность использования различных техник оптимальной декомпозиции для повышения параллельной эффективности вычислений в Грид.

1. Введение

Современный этап развития технологий Грид характеризуется многообразием концепций их интерпретации и использования. Классические (и независимые) определения, сформулированные I. Foster и C. Kesselman [1] и M. Livny [2] по мере внедрения Грид расширяются и модифицируются. В обзорной работе [3] отмечается, что основной чертой Грид, отделяющей его от других систем распределенных вычислений (например, P2P, utility computing и пр.) является открытая система стандартизации, на основании которой осуществляется объединение вычислительных и информационных ресурсов. Как следствие, в настоящее время наиболее часто Грид интерпретируют как единую вычислительную среду, в рамках которой выполняются задачи различных пользователей. Другими словами, во главу угла ставится виртуализация вычислительных ресурсов, а не пользовательских приложений. Это связано как с технологическими проблемами разработки приложений под Грид, так и с вопросами их коммерческого использования [4].

В рамках определения [3] допустимой является интерпретация Грид как среды параллельных вычислений, наряду с более традиционными (например, кластерными, гибридными, P2P) архитектурами. В отличие от использования Грид для удаленного выполнения заданий на мощных вычислительных системах, проблема параллельных вычислений связана с объединением и синхронизацией большого количества вычислительных узлов (в общем случае - географически разнесенных и принадлежащих разным пользователям) для решения одной задачи. Очевидно, что параллельные вычисления в Грид по производительности не могут сравняться с традиционными кластерными системами в силу высокой коммуникационной составляющей. Однако использование концепции Грид способно предоставить пользователю сравнительно дешевую и неограниченно расширяемую параллельную архитектуру. Как следствие, потенциальными потребителями таких технологий могут быть задачи, связанные с моделированием сложных систем [5], состоящих из огромного количества взаимодействующих объектов с дальними связями, размещаемых в оперативной памяти. В частности, к таким задачам относится моделирование распространения ВИЧ и вируса Гепатит-В в глобальном масштабе на основе аппарата комплексных сетей [6].

Реализация параллельных вычислений в Грид, по сравнению, например, с кластерными системами, даже на корпоративном уровне требует учета ряда специфических факторов, в общем случае отрицательно влияющих на параллельную эффективность. В первую очередь к ним относятся стохастическая изменчивость параметров сетей и вычислительных узлов, их нестационарность и неоднородность [7]. В данной работе рассматриваются особенности организации параллельных вычислений в корпоративной среде Грид на примере Intel Grid Programming Environment.

*Работа выполнена при частичной финансовой поддержке проекта корпорации Intel SPB/R&D/77/2007 «Методология портирования приложений в среду GPE»

2. PEG: программная система параллельных вычислений на основе Intel Grid Programming Environment

Многообразие интерпретаций Грид породило большое количество соответствующих программных систем, оболочек и инструментария (например, библиотек). Наиболее известные, и, пожалуй, всеобъемлющие пакеты Globus® [9] и gLite [10] представляют собой громоздкие и сложные программные системы, которые при больших затратах на разработку Грид-приложений, очевидно, предоставляют наибольшие возможности. Для упрощения технологий разработки приложений под Грид существуют объектно-ориентированные надстройки над названными продуктами, среди которых можно выделить Intel® GPE™ (Grid Programming Environment™) [11] и SAGA (The Simple API for Grid Applications) [12]. В данной работе в качестве базовой рассматривается среда Intel GPE™. Она является надстройкой над Globus, реализована на языке Java™, что автоматически делает среду кроссплатформенной, а также подлежит свободному распространению в соответствии с лицензией GPL [13]. Несмотря на широкие потенциальные возможности Intel GPE™, в данной работе рассматривается Грид только корпоративного уровня, в котором в качестве рабочих узлов используются рабочие станции, располагающиеся в сети одной организации. К тому же, такой выбор подкрепляется тем, что эффективность параллельных вычислений при «вынесении» рабочих узлов за пределы высокоскоростной сети (например, в область WAN) существенно снижается.

Для реализации параллельных вычислений общего вида на основе среды Intel GPE™ авторами разработана программная система PEG (Parallel Execution on GPE). Ее назначением является автоматизация и упрощение процесса переноса параллельных приложений кластерного уровня под Грид. Архитектура системы представлена на Рис. 1а.

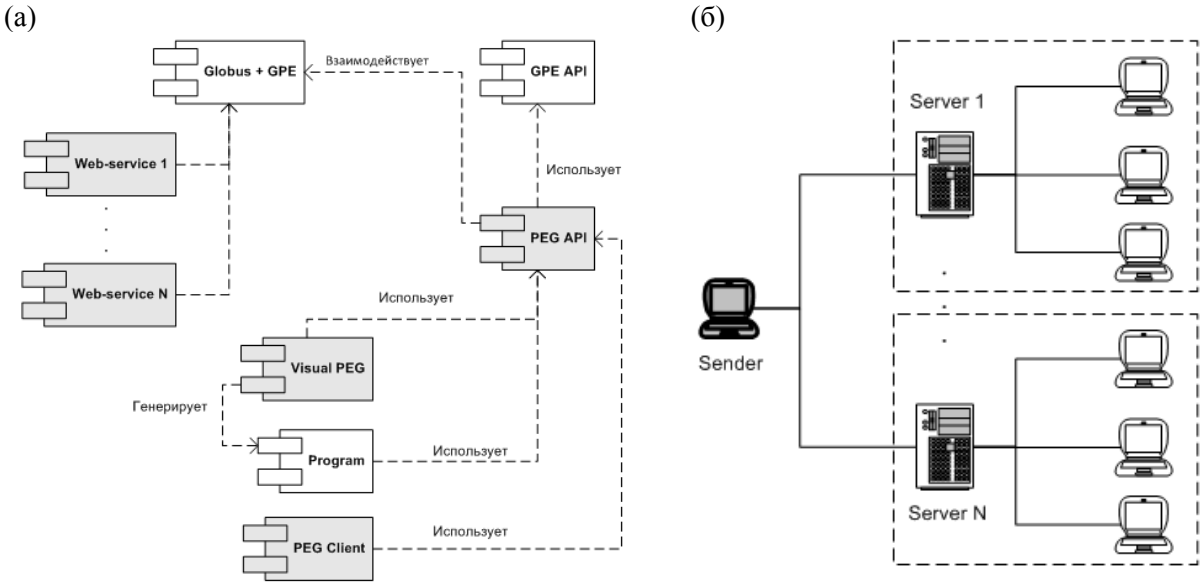

Рис. 1. Программная (а) и аппаратная (б) архитектура среды PEG

PEG является как исполняющей системой, так и набором программных интерфейсов (API). Исполняющую часть реализует программный модуль PEG Client, который позволяет запускать приложения в среде Грид. Блоком с пометкой Program в схеме указан исполняемый модуль программы, которую предполагается выполнять параллельно на Грид. Таким образом, PEG позволяет «обернуть» готовую программу и выполнять ее параллельно, если распараллеливание ведется функционально или по данным. Данные типы распараллеливания встречаются в подавляющем большинстве научных и инженерных задач. Процедуры, отвечающие за проведение функциональной декомпозиции и декомпозиции по данным, отданы на откуп пользователя.

На рис. 1б показана архитектура аппаратного решения Грид-системы, которую использует PEG. Из рисунка видно, что система представляет собой отдельные объединения рабочих уз-

лов, каждое — со своим сервером. Узел, производящий запуск параллельного задания, показан серым цветом (слева). Он ведет обмен данными только с серверами, не обращаясь напрямую к рабочим узлам, что позволяет скрыть аппаратную архитектуру системы и абстрагироваться от ее состава. На каждом сервере установлен контейнер Globus с функционирующими в нем сервисами GPE. Таким образом, разработанное программное средство позволяет использовать (разумно) неограниченное количество рабочих узлов, объединенных серверами, для параллельного исполнения приложений в среде Грид.

3. Параллельные вычислительные процессы в Грид под управлением системы PEG

PEG не является в полной мере системой прототипирования. Оптимальное решение задач декомпозиции, связывания, агломерации и отображения на вычислительную архитектуру должны быть решены пользователем на этапе проектирования своего приложения, что требует наличия априорных знаний о специфике параллельных вычислительных процессов под Грид.

Параллельные вычислительные процессы в Грид, по сравнению, например, с кластерными аналогами, обладают, как минимум, тремя специфическими особенностями. К ним относятся:

- стохастический характер трафика в коммуникационных сетях (как следствие, изменчивость пропускной способности);
- стохастический характер производительности вычислительных узлов, обусловленный возможным наличием многих пользователей;
- неоднородность коммуникационной сети и вычислительных узлов, заключающаяся в выборе требуемых для выполнения задачи узлов из случайного (на данных момент) множества доступных ресурсов.

Наличие стохастического трафика естественным образом обуславливает случайное время работы приложений в Грид; понятие производительности таких распределенных систем требует вероятностного описания. В качестве иллюстрации на рис. 2 приведены вероятностные характеристики параллельного ускорения, полученные на тестовом полигоне PEG, состоящем из четырех однотипных рабочих станций, включенных в общую сеть. Распределение параллельного ускорения вычислялось по методике, описанной авторами в [13].

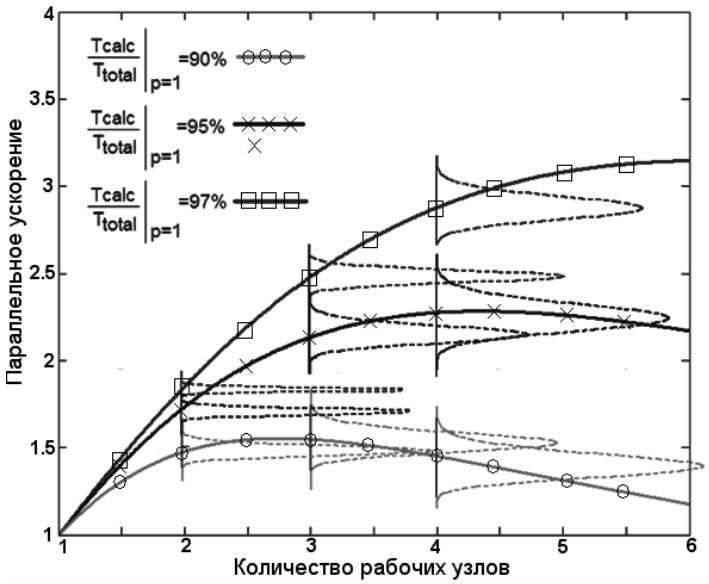

Рис 2. Изменение вероятностных характеристик параллельного ускорения в зависимости от количества узлов по экспериментальным данным.

Из рис. 2 следует, что разброс параллельного ускорения, обусловленный стохастичностью трафика корпоративной сети, может быть достаточно велик – до 30-50% для четырех рабочих узлов. С увеличением числа узлов разброс увеличивается.

3.1 Аналитические модели параллельной производительности в Грид

Для объяснения экспериментальных результатов (рис. 2) рассмотрим аналитические модели параллельной производительности в Грид для наиболее простых случаев. Для этого ограничимся одной парой Sender-Server, см. рис. 1б, и рассмотрим работу тестового приложения, соответствующего BSP-модели параллельной программы [14] из трех супершагов:

1. Sender разбивает исходные данные объемом N на p частей N_i (этот процесс занимает время T_0), после чего отправляет каждую часть в отдельном потоке Server через выделенный канал с характеристикой[1] $t_w^{(0)}$. Все потоки имеют одинаковый приоритет и разделяют между собой канал. Процесс передачи заканчивается барьером.
2. Server играет роль маршрутизатора. Он параллельно раздает части данных N_i на вычислительные узлы по независимым каналам с характеристикой $t_w^{(i)}$. Как только передача данных на конкретный вычислительный узел закончена, он выполняет N_i операций, каждая из которых характеризуется временем $t_c^{(i)}$. Как только вычисления завершены, на Server с вычислителя передаются результаты расчетов объемом M. Все операции расчетов и пересылок выполняются параллельно и ресурсы не разделяют. Этот этап также заканчивается барьером.
3. С Server на Sender передаются результаты расчетов с p вычислителей объемом Mp. После этого Sender выполняет операции по их обработке, которые занимают время T_n

Ограничимся однородной вычислительной системой, т.е. $t_{c_i} = t_{c_j} = t_c$, и случаем, когда канал между Sender и Server имеет фиксированное время передачи единицы информации и зададим случайное изменение характеристик пропускной способности сети в форме

$$t_w^{(i)} = t_w + \eta_i, \text{ где } \eta_i \in N(m_\eta, \sigma_\eta). \quad (1)$$

Тогда длительность первого и третьего супершагов суть детерминированная величина, а второго – случайная величина $T_c = \max_{i=1,p}[T_i]$, где T_i — время работы с каждым вычислителем. В соответствии с (1) оно является гауссовой случайной величиной со средним значением и среднеквадратичным отклонением:

$$m_T = (t_w + m_\eta)\left(\frac{N}{p} + M\right) + t_c\frac{N}{p}, \qquad \sigma_T = \left(\frac{N}{p} + M\right)\sigma_\eta, \quad (2)$$

соответственно. Тогда закон распределения величины T_c (как максимума случайной выборки) асимптотически (при значительных p) является предельным распределением I типа (Гумбеля, или Фишера-Типпета) [15]:

$$F_{T_c}(x) = \exp\left[-\exp\left[-a_p(x - b_p)\right]\right], \quad (3)$$

где коэффициенты

$$a_p = \frac{\sqrt{2\ln(p)}}{\sigma_T} \text{ и } b_p = \left[\sqrt{2\ln(p)} - \frac{\ln(\ln(p)) + \ln(4\pi)}{2\sqrt{2\ln(p)}}\right]\sigma_T + m_T \quad (4)$$

зависят от количества рабочих узлов p. На рис. 3 в качестве примера приведены результаты расчета распределений общего времени выполнения задачи от количества вычислительных узлов. На рисунке 3а изображено время работы с распределениями при гомогенной вычислительной структуре и гетерогенной сетевой. На рисунке 3б имеется также гетерогенность по вычислительным мощностям (экспериментальные данные с ядерными оценками). Видно, что при увеличении количества процессоров разброс времени выполнения имеет тенденцию умень-

[1] Время на передачу единицы информации

шаться. Также, можно указать, что при внесении неоднородности по вычислительной мощности разброс увеличивается при остальных фиксированных параметрах.

Модель (3-4) обобщается на случай системы с однородной вычислительной сетью и случайной изменчивостью производительности вычислительных узлов. Тогда выражение (1) заменяется $t_{c_i} = t_c + \pi_i$, где $\pi_i \in N(m_\pi, \sigma_\pi)$, и параметры (2) в (4) приобретают вид:

$$m_T = t_w\left(\frac{N}{p} + M\right) + t_c\frac{N}{p} + m_\pi, \quad \sigma_T = \left(\frac{N}{p}\right)\sigma_\pi \tag{5}$$

В том случае, если стохастический характер свойственен одновременно и производительности узлов, и пропускной способности коммуникационной сети, класс экстремального распределения (3), по-видимому, сохраняется. Однако его коэффициенты уже не могут быть выписаны в аналитическом виде. Потому для решения исследования таких систем необходимо применять методы численного (имитационного) моделирования.

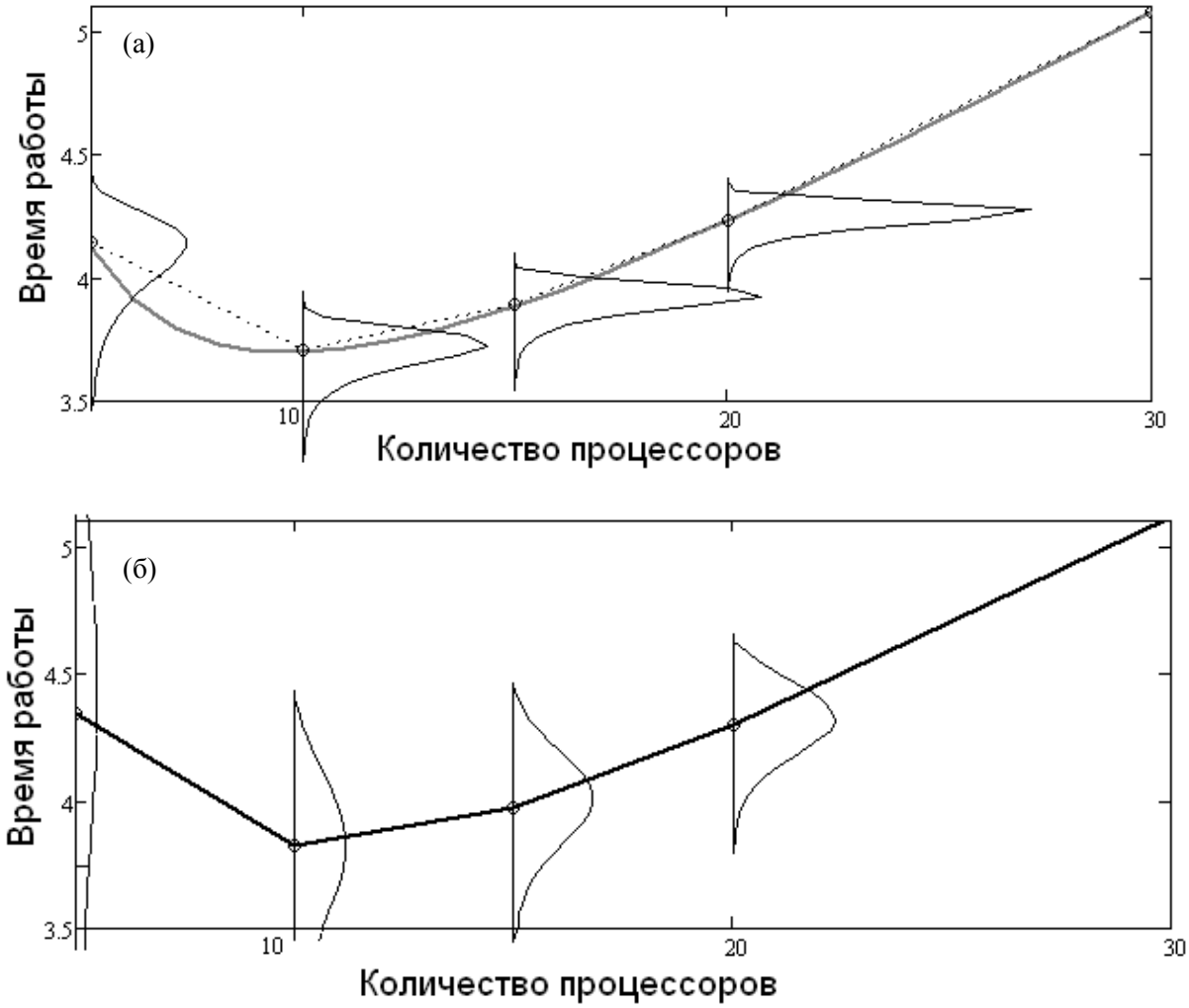

Рис 3. Время работы в зависимости от количества вычислителей
(а) – стохастичность сетевого трафика (15%), (б) – стохастичность сетевого трафика и производительности вычислителей (15%+15%).

3.2 Имитационное моделирование параллельных вычислительных процессов в Грид

Для исследования производительности параллельных приложений в Грид-системах со сложной топологией, напрямую не описываемой аналитическими моделями (1-5), разработан программный симулятор среды Грид. В процессе разработки использован опыт работы [16]. Детальная, низкоуровневая симуляция работы Грид среды учитывает влияние большого числа

факторов, ни один из которых не имеет лидирующего значения. Программный симулятор позволяет задавать произвольные сетевые топологии, производительность и загруженность для всех вычислительных узлов, пропускную способность и загруженность для всех сетевых каналов; законы распределения для шума загруженностей вычислительных узлов и сетевых каналов. При этом на его основе могут быть реализованы различные методы параллельной декомпозиции и связывания. Кроме того, программный симулятор отображает детальную картину процесса выполнения параллельного приложения в среде Грид. Он учитывает процесс разделения сетевого канала между двумя независимыми соединениями, при котором происходит падение скорости передачи по каждому из соединений.

На рис. 3 ядерные оценки распределений времени работы приложения по данным моделирования сопоставлены с аналитическими результатами для вычислительной системы с однородными узлами, см. раздел 3.1. Также приведены характеристики, полученные для системы, у которой и производительность, и пропускная способность коммуникационной сети стохастизированы, и аналитические выражения для распределений в явном виде получить невозможно. Хорошее совпадение модельных и аналитических кривых для тестового случая из раздела 3.1 является определенной гарантией адекватной работы симулятора и возможностью его применения для решения более сложных задач.

4. Управление параллельной эффективностью вычислений в PEG

Из раздела 3 следует, что время работы (как и ускорение) параллельного приложения является случайной величиной, распределение которой характеризуется параметрами масштаба и сдвига. Параметр масштаба, управляющий размахом распределения, может быть интерпретирован как объективный случайный фактор (своего рода случайная ошибка системы). Параметр сдвига, в свою очередь, интерпретируется как систематический фактор, управление которым позволяет сократить время работы приложения и, как следствие, повысить параллельную эффективность. Построенные выше аналитический модели описывают случай, когда данные разбиваются на равные порции. В этом случае не учитывается текущая топология Грид-среды, а, значит, время работы программы при таком разбиении не является лучшим из возможных. Решением проблемы может стать проведение оптимальной декомпозиции, которая бы учитывала текущую топологию Грид-среды (вычислительные мощности узлов, пропускные способности сети), тем самым, оптимизируя время выполнения приложения и сокращая границы его колебаний. В отличие от кластерных решений, имеющих в распоряжении быстрые сетевые каналы, которые не заняты «побочным» трафиком, в Грид-среде время на коммуникации становится более значительным по отношению к времени вычислений, и менее детерминированным. Это позволяет использовать более гибкие схемы декомпозиции, оптимальным образом разделяя коммуникационные каналы между вычислительными узлами.

С использованием программного симулятора исследованы три основных модели оптимальной декомпозиции: пропорциональная (только по производительности узлов), барьерная и каскадная. Результаты экспериментов показали, что для линейных по данным алгоритмов наиболее эффективной является каскадная декомпозиция. Она в некоторых случаях позволяет в 1,5-2 раза сократить среднее значение времени выполнения приложения по сравнению с равномерной декомпозицией.

5. Заключение

Множество прикладных задач, для которых допустимо использовать Грид как систему параллельных вычислений, ограниченно, но достаточно велико. Их эффективное решение возможно только за счет планирования выполнения приложений на основе информации о топологии Грид-среды. Главным инструментом для достижения этой цели должна стать разработка новых эффективных методов оптимальной декомпозиции на основе вероятностного подхода к описанию производительности Грид. Это позволяет сформулировать новый подход к оценке и нормированию машинного времени в Грид, в терминах вероятностных рисков превышения приложением заданного временного ограничения.

Литература

1. Foster I., Kesselman. C. The Grid: Blueprint for a New Computing Infrastructure. Morgan-Kaufman, 1999.
2. Thain D., Tannenbaum T., and Livny M. Condor and the Grid // Berman F., (Editor), Fox G. (Editor), Hey J.G. A. (Editor) Grid Computing: Making The Global Infrastructure a Reality. —John Wiley, 2003.
3. Stockinger H. Defining the Grid — a snapshot of the current view // Journal of Supercomputing. –Springer Science+Business Media, 2007.
4. Уэйлгам Т. Grid взята в заложники // Директор ИС, №1, 2006.
5. Sloot P. M. A., Frenkel D., Van der Vorst H.A., van Kampen A., Bal H. E., Klint P., Mattheij R.M.M., van Wijk J., Schaye J., Langevelde H.-J., Bisseling R. H., Smit B., Valenteyn E., Sips H., Roerdink J. B. T. M. and Langedoen K. G. Computational e-Science: Studying complex systems in silico. —A National Coordinated Initiative. White Paper. —2007 [http://www.science.uva.nl/research/pscs/papers/archive/Sloot2007a.pdf]
6. Sloot P.M.A., Ivanov S.V., Boukhanovsky A.V. et al. Stochastic simulation of HIV population dynamics through complex network modeling // International Journal of Computer Mathematics. — In press, 2007.
7. Ларченко А.В., Ковальчук С.В., Иванов С. В., Одинцов И. О., Бухановский А.В. Проблемы переноса вычислительных приложений кластерного уровня в среду Грид на примере Grid Programming Environment // Труды Всероссийской научной конференции Научный сервис в сети Интернет — М.: Издательство Московского университета, 2007. — с. 134.
8. Globus® Toolkit: [http://globus.org/toolkit/].
9. gLite: [http://glite.web.cern.ch/glite/].
10. Grid Programming Environment™: [http://gpe4gtk.sourceforge.net/].
11. SAGA: [http://saga.cct.lsu.edu/].
12. GNU General Public License: [http://www.gnu.org/copyleft/gpl.html/].
13. Ларченко А.В., Дунаев А.В., Бухановский А.В. Анализ и моделирование производительности параллельных стохастических алгоритмов, адаптированных к особенностям многоядерных вычислительных архитектур // Труды Всероссийской научной конференции Научный сервис в сети Интернет — М.: Издательство Московского университета, 2007. — с. 156.
14. Gerbessiotis A.V. Architecture independent parallel algorithm design: theory vs practice // Future Generation Computer Systems, 18, 2002, pp. 573—593.
15. Лидбеттер М., Линдгрен Г., Ротсен Х. Экстремумы случайных последовательностей и рядов. —М., Мир, 1989, 392 с.
16. Blythe J., Jain S., Deelman E., Gil Y., Vahi K., Mandal A., Kennedy K. Task Scheduling Strategies for Workflow-based Applications in Grids. — 2005, pp. 1—9.