ИНСТРУМЕНТАЛЬНАЯ ОБОЛОЧКА ПРОЕКТИРОВАНИЯ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ГРИД-СЕРВИСОВ КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ И ОБРАБОТКИ ДАННЫХ

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

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 распределенных и принадлежащих разным пользователям или организациям) для решения одной задачи. Очевидно, что параллельные вычисления в Грид по производительности не могут сравняться с традиционными кластерными системами в силу высокой латентности коммуникаций. Однако использование концепции Грид способно предоставить пользователю сравнительно дешевую и практически неограниченно расширяемую параллельную архитектуру. Как следствие, потенциальными потребителями таких технологий могут быть задачи, связанные с моделированием сложных систем [5, 6], состоящих из огромного количества взаимодействующих объектов с дальними связями, размещаемых в оперативной памяти.

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 связи с этим эффективное выполнение параллельных вычислений в Грид требует использования особых подходов к проектированию приложений, применения специфических механизмов исполнения и балансировки нагрузки.

В данном докладе обсуждается реализация этих подходов в рамках инструментальной оболочки проектирования высокопроизводительных приложений для Грид-архитектур (авторский акроним — PEG2), целью которой является поддержка разработки вычислительно эффективных прикладных сервисов компьютерного моделирования и обработки данных.

НАЗНАЧЕНИЕ ИНСТРУМЕНТАЛЬНОЙ ОБОЛОЧКИ

Инструментальная оболочка PEG2 позиционируется как высокоуровневая надстройка над системой промышленного уровня Intel Grid Programming Environment (Intel GPE), которая, в свою очередь, является надстройкой над более низкоуровневым решением Globus Toolkit. Пакет Globus Toolkit представляет собой набор библиотек и сервисов для построения крупномасштабных распределенных систем (не обязательно «полноценных» Грид). Используемая в PEG2 система Intel GPE представляет собой набор приложений, сервисов и программных библиотек, необходимых для развертывания и использования Грид-систем, соответствующих (де-факто) стандартам сообщества Грид. Поскольку функциональные возможности Intel GPE ограничиваются реализацией протоколов передачи данных между вычислительными узлами и обеспечением возможности запуска одиночных задач и организации получение информации об их выполнении (в достаточно простой форме), то более сложные функции (возможности администрирования выполнения сложных одиночных или параллельных задач, сбор статистической информации и пр.) должны быть реализованы пользователем. Потому оболочка PEG2 использует Intel GPE как среду исполнения и набор программных интерфейсов для реализации функционала более высокого уровня. Она ориентируется на пользователей-предметников, даже не знакомых со спецификой организации вычислений в Грид. Иерархия решений представлена на рис. 1а.

Основной идеей PEG2 является предоставление удобных инструментов для создания вычислительных Грид-сервисов на основе существующих программных средств путем написания специальных «оберток» (wrapper). Модификация исполняемого модуля программы (при условии удовлетворения ряду априорных требований) при этом не требуется. Таким образом, удаленный пользователь получает возможность обращаться к приложению средствами Грид. Встроенная в PEG2 визуальная оболочка проектирования дает возможность пользователю на основе существующих сервисов создавать сложные потоки задач (workflow), компонуя крупноуровневую структуру процесса вычислений. В данной модели вызов сервиса представляется отдельным блоком, который может запускаться как в одиночном, так и в параллельном режимах. Эта возможность позволяет повысить эффективность вычислений за счет распределения задач не только функционально, но и по данным. Для эффективного выполнения workflow требуется реализация сложных балансировочных алгоритмов для распределенных неоднородных систем. Это становится возможным за счет использования моделей параллельной производительности приложений, входящих в состав Грид-сервисов, на основе знаний об их внутренней структуре. Использование параметрических моделей производительности, идентифицированных посредством информации о характеристиках удаленных узлов и пропускной способности отдельных сетевых каналов, позволяет прогнозировать характеристики выполнения workflow (время, параллельное ускорение для отдельных блоков, объем передаваемых данных и пр.).

АРХИТЕКТУРА ИНСТРУМЕНТАЛЬНОЙ ОБОЛОЧКИ

На рис. 1а. представлена крупноуровневая архитектура инструментальной оболочки PEG2. На схеме указаны основные подсистемы от самого нижнего уровня (Globus) до самого верхнего (визуальная оболочка прототипирования). Блоки верхних уровней используют функциональности блоков нижних уровней для реализации своей функциональности.

Самый нижний уровень представляет собой контейнер Globus из состава Globus Toolkit. Он являет собой Web-сервер с поддержкой HTTPS и предоставляет среду выполнения для Web-сервисов. В контейнер могут быть включены любые пользовательские сервисы, реализующие стандарт WSRF [8] и использующие Globus API.

Следующие два уровня (GPE Server API и GPE Services) необходимы для реализации серверной части Intel GPE. GPE Server API предоставляет набор интерфейсов для упрощенного взаимодействия с сервисами контейнера Globus. Сервисы GPE используют этот API для реализации своей функциональности, обеспечивая работу системы.

На следующем уровне находится GPE Client API, предназначенный для обеспечения работы клиентских частей приложений с серверной частью. Это базовый API для работы с Грид сторонних приложений. Он используется PEG2 напрямую. На основе GPE Client API реализован модуль PEG Grid API, который предоставляет высокий уровень абстрагирования от более низкоуровневого GPE Client API, позволяя естественным образом использовать возможности Грид-технологий.

Одним из важнейших модулей PEG2 является Workflow API — модуль реализации работы со сложными процессами вычислений (workflow). Он позволяет создавать структуру workflow, которая будет представлять собой исполняемый каркас, готовый к запуску в среде Грид. Модуль Execution API (Модуль исполнения) отвечает за запуск скомпонованных с помощью Workflow API потоков вычислений в среде Грид. Он реализует следующие функциональные характеристики: запуск готового workflow в среде Грид, имитационный запуск workflow с целью получения прогноза времени исполнения, сбор статистики процесса выполнения, обработка ошибок, возникающих в Грид во время выполнения. В свою очередь, модуль Performance API (модуль производительности) отвечает за мониторинг производительности удаленных целевых систем. Его задачей является обеспечение возможности запуска удаленных тестов и получения данных о таких характеристиках, как производительность узла, время на передачу единицы информации по сети, латентность сети, накладные расходы на общение с Грид-сервером и пр. Перечисленные модули PEG2, являясь достаточно изолированными, могут использовать функциональности друг друга; каждый из них имеет четко описанные внешние интерфейсы, через которые происходит взаимодействие.

Визуальная оболочка (ВО) образует самый верхний уровень комплекса. Она предоставляет пользователю удобный интерфейс для прототипирования параллельного приложения и его запуска, мониторинга Грид-инфраструктуры и изучения ее производительности.

СОСТАВ ТИПОВОГО ПРИКЛАДНОГО ГРИД-СЕРВИСА

Пользователь при работе с оболочкой PEG2, обращается не к приложениям как таковым, а к представляющим их Грид-сервисам, которые могут быть доступны удаленно и не зависеть от аппаратно-программной платформы пользователя. Таким образом, Грид-сервис представляет собой уровень виртуализации пользовательских приложения для обеспечения возможности их использования в Грид.

Для реализации приложения в виде прикладного Грид-сервиса (ПГС), для него необходимо создать для приложения набор компонент (составляющих wrapper — обертку), которые сделают возможным его выполнение в рамках Грид. Такая обертка необходима по причине того, что при встраивании существующего программного средства в модель Грид-сервиса (при условии неизменности его исходного кода) требуется создать соответствующие

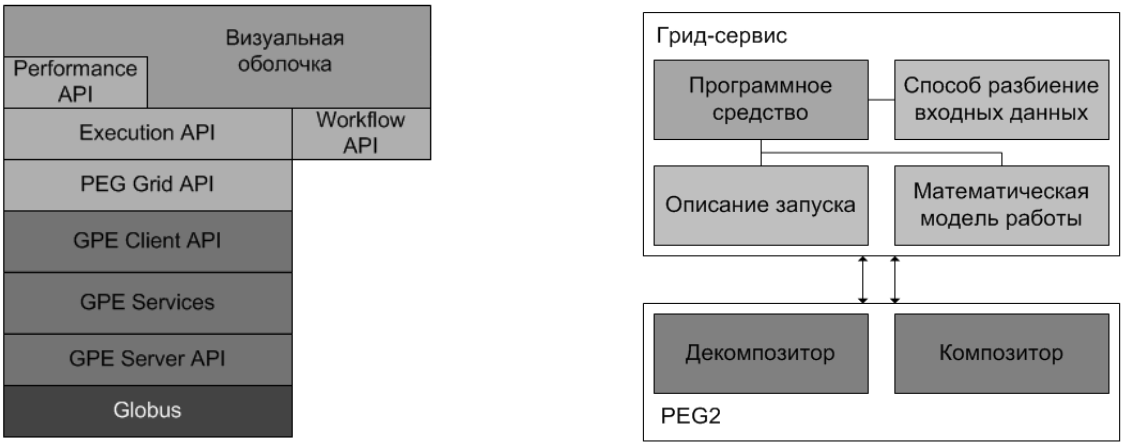
интерфейсы для взаимодействия с Грид. Разработка любого ПГС, таким образом, состоит в реализации этих интерфейсов в разрабатываемых компонентах и дальнейшее связывание полученных компонентов. Схематически структура типового ПГС показан на рис. 1б.

ПГС включает в себя ряд компонентов, взаимодействующих друг с другом, таких как:

1) программное средство — собственно программа, для которой создается ПГС, реализованное с учетом ряда требований (например, консольное приложение, исключающее интерактивное взаимодействие с пользователем).
2) описание программного средства на формальном языке. Она включает в себя данные и знания, необходимые для использования программного средства в рамках ПГС:
    a) информация о способе запуска, входных и генерируемых данных (файл, аргументы);
    b) формализация вычислительного алгоритма, заложенного в программное средство, которая позволяет оценить его сложность (как количество базовых операций) в зависимости от характеристик данных. При отсутствии достоверной информации о характеристиках алгоритма могут использоваться экспериментальные профили вычислений (например, в виде таблицы или некоторой функциональной аппроксимации);
    c) правила распараллеливания в форме информации о способах разбиения входных данных и объединения результатов расчетов;
3) декомпозитор — программный модуль, задающий алгоритм разбиения исходных данных на основании информации из 2c;
4) композитор — программный модуль, задающий алгоритм сбора выходных данных, полученных с различных целевых систем на основании информации из 2c.

Важно, что композитор и декомпозитор могут представляться как непосредственно в составе ПГС, так и реализовываться отдельно на основании информации из раздела 2c.

Прикладной Грид-сервис, соответствующий изображенной на рис. 1б структуре, может быть использован в оболочкой PEG2 независимо, или в составе потока выполнения (workflow).



(а) (б)

Рис. 1. Схемы: (а) общая архитектура системы и (б) архитектура прикладного Грид-сервиса

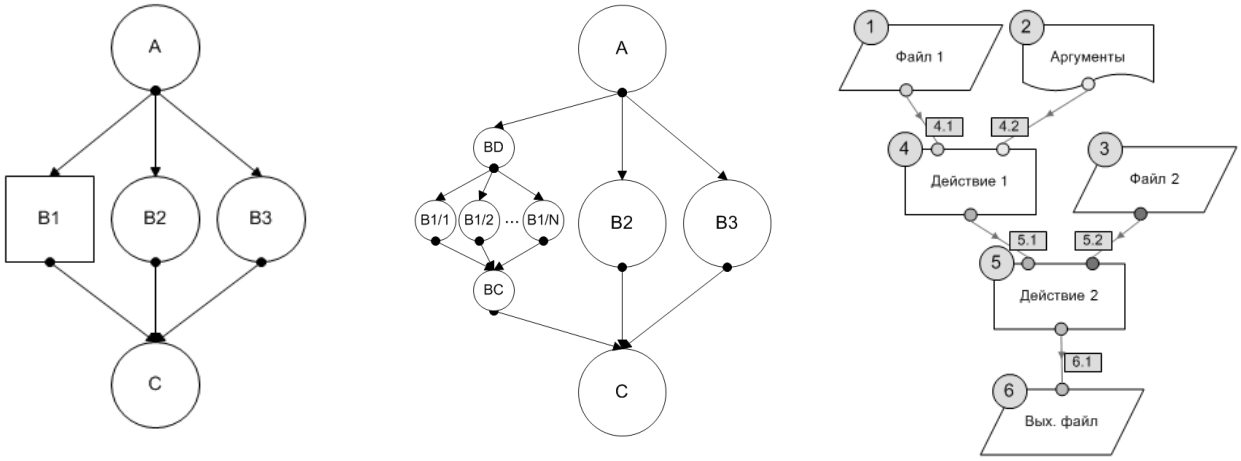
РЕАЛИЗАЦИЯ ПОТОКОВ ВЫПОЛНЕНИЯ (WORKFLOW)

Для реализации сложных вычислительных процессов в практике Грид-вычислений нередко применяются т.н. потоки выполнения (workflow), которые позволяют компоновать различные вычислительные задачи в последовательные и параллельные структуры, направленные на достижение общего результата. Workflow может быть описан графом, где вершины являются действиями, выполняемые над данными, а ребра — потоками данных (см. рис. 2). Таким образом реализуется концепция крупноуровневого программирования, когда пользователь задает лишь общий поток вычислений, блоками которого являются Грид-сервисы. Такая модель позволяет абстрагироваться от тонкостей запуска отдельных приложений.

В оболочке PEG2 в качестве блоков действий выступают прикладные Грид-сервисы, описанные в предыдущем разделе. Визуальная оболочка, встроенная в PEG2, позволяет пользователю визуально проектировать будущий workflow и задавать характеристики его выполнения. Таким образом, при наличии готовых Грид-сервисов, которые могут предоставляться сторонними разработчиками, пользователь имеет возможность проводить вычисления, создавая наглядные конструкции, которые отражают семантику вычислительных процессов. К тому же, одной из

главных особенностей PEG2 является то, что Грид-сервисы могут запускаться в параллельном режиме (см. рис. 2б), что позволяет пользователю в полной мере задействовать возможности Грид как параллельной вычислительной архитектуры.

В случае параллельного выполнения Грид-сервиса, происходит декомпозиция исходных данных, рассылка их на вычислительные узлы, выполнение расчетов, сбор и слияние, в соответствии с правилами, установленными в разделе 2 описания ПГС.



(а) (б) (в)

Рис. 2. Представление задачи в виде потока выполнения (workflow):
(а) с простыми действиями; (б) с распараллеливаемыми действиями;
(в) представление в нотации визуальной оболочки PEG2.

УПРАВЛЕНИЕ ЭФФЕКТИВНОСТЬЮ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ

Для реализации задачи оптимального (с точки зрения параллельных вычислений) выполнения workflow оболочка PEG2 позволяет использовать различные алгоритмы статической и динамической балансировки параллельного исполнения, что является принципиальным для неоднородных вычислительных систем, в которых различие статических и динамических показателей отдельных узлов и сетевых каналов очень велико. Для построения оптимальных алгоритмов балансировки требуются знания как о производительности приложений, используемых в workflow, так и о составе и характеристиках самой вычислительной среды, на которой предполагается осуществлять запуск. Знания о структуре приложений предоставляет сам ПГС. Информация о характеристиках вычислительной среды обеспечивается в PEG2 путем тестирования производительности Грид (статические характеристики: производительность узлов, пропускные способности сетевых каналов) и его последующего мониторинга (динамические характеристики: загрузка узлов и сетевых каналов).

Задача построения оптимальных алгоритмов балансировки усложняется еще и тем, что для вычислений в Грид является обычным разделение ресурсов, когда, например, один и тот же вычислительный узел может одновременно использоваться несколькими пользователями. То же самое относится и к сетевым каналам. Для решения этой проблемы применяются стохастические модели производительности, учитывающие «коммунальные» эффекты в Грид в терминах случайных процессов с заданными характеристиками. В этом случае время выполнения приложения выражается случайной величиной, которая формируется за счет двух факторов: систематического (обусловленного характерной неоднородностью вычислительной архитектуры) и случайного (обусловленного непредсказуемыми «коммунальными» эффектами в Грид). В таком случае управление эффективностью параллельных вычислений за счет статической балансировки направлено на уменьшение систематической составляющей времени вычислений, оставляя случайную составляющую неизменной. Динамическая балансировка, напротив, позволяет в отдельных случаях уменьшить случайную составляющую; однако в общем случае ее эффект остается дискуссионным.

Учитывая особенности специфической для Грид-вычислений шинной архитектуры, в рамках PEG2 реализованы алгоритмы каскадных статических балансировок (прямая, обратная, с заданным распределением). Каскадные схемы балансировки направлены на минимизацию простоев коммуникационного канала, обеспечивая условие, чтобы в каждый момент времени по каналу осуществлялось синхронное взаимодействие только между одной парой узлов. Каскадные схемы балансировки (см. рис. 3) имеют ряд преимуществ перед асинхронными схемами, которые делят канал между несколькими узлами. Во-первых, при их использовании не происходит пересечения разбитых данных, передаваемых на различные узлы, а во-вторых, узел первым получивший свои данные, сразу же начинает их обработку.

Теоретический анализ балансировочных схем показал, что для однородных систем наиболее эффективной является прямая каскадная схема (рис. 3а). Обратная каскадная схема (рис. 3б) также обеспечивает неплохую эффективность, однако при некоторых значениях параметров аргументов стремится к вырождению. Для неоднородных систем используются экзотические виды балансировок (рис. 3в), когда последовательность выбора целевых систем для передачи данных задается распределением, получаемым на основе численного решения задачи оптимизации (например, с помощью генетического алгоритма) или некоторого эвристического критерия.

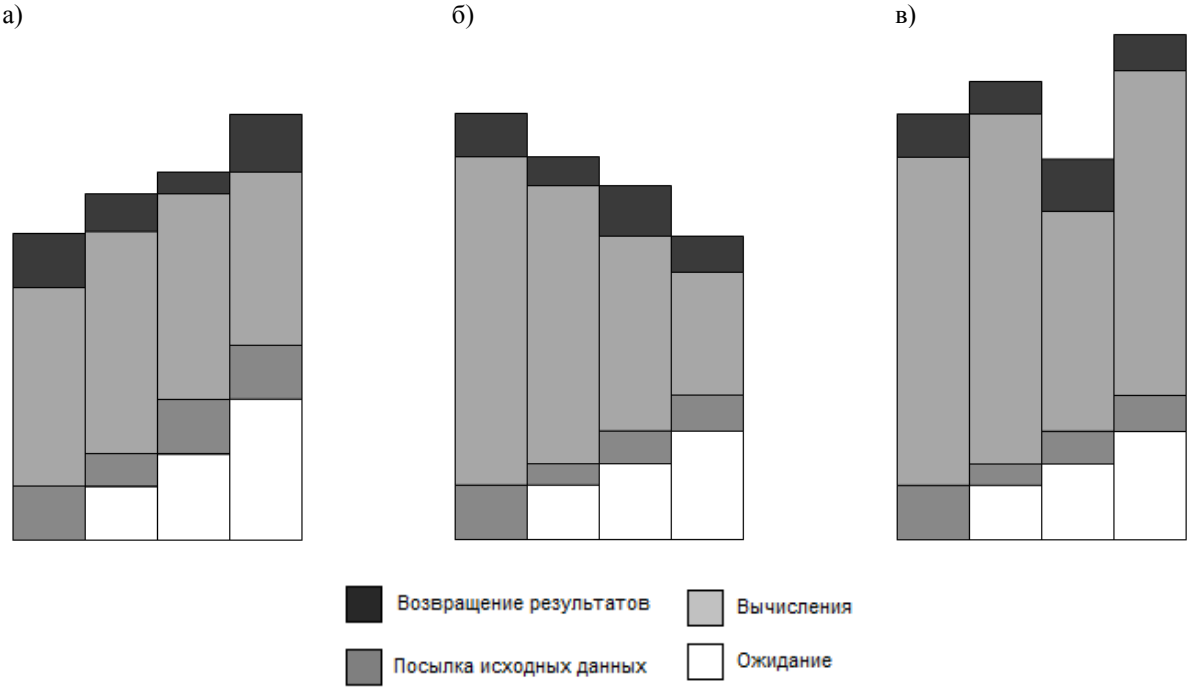


Рис. 3. Виды каскадных балансировок:
(а) прямая (б) обратная (в) с заданным распределением

Проведенные экспериментальные исследования реализации каскадных балансировок в PEG2 показали их эффективность. В качестве простейшего примера на рис. 4 представлены графики времени выполнения (в терминах ядерных оценок плотности распределения) параллельного запуска модельного Грид-сервиса, реализующего задачу параметризации климатических спектров морского волнения [9] с использованием равномерного разбиения и прямой каскадной балансировки. Вычисления производились на четырех целевых системах с процессорами класса Intel Core 2 Duo в общей вычислительной сети на основе Ethernet.

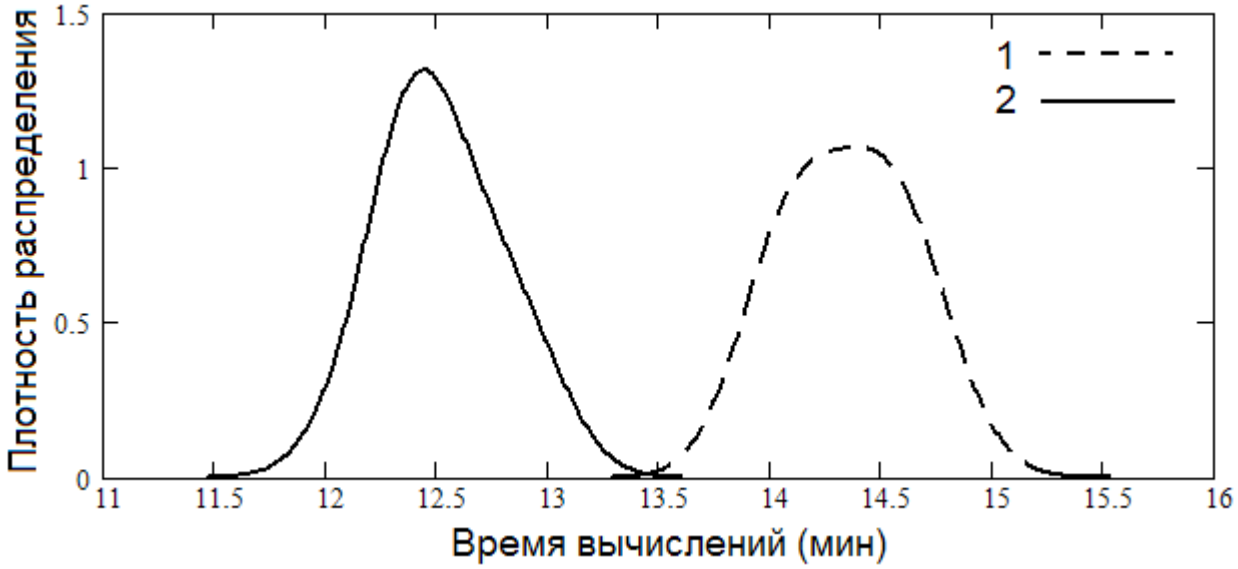


Рис. 4. Плотность распределения (ядерная оценка) времени вычислений на четырех целевых системах. 1 – с равномерной балансировкой (асинхронная схема), 2 – с прямой каскадной балансировкой.

Из рис. видно, что при применении каскадной балансировки происходит сдвиг распределения времени работы Грид-сервиса при сохранении разброса. При этом оценки плотностей распределения практически не пересекаются, что свидетельствует о систематическом уменьшении времени работы параллельного приложения (примерно на 15% для данного случая) за счет использования процедуры каскадной балансировки.

ЗАКЛЮЧЕНИЕ

Инструментальная оболочка PEG2 является высокоуровневой надстройкой над Intel Grid Programming Environment и Globus Toolkit, и реализует управление параллельными процессами в Грид на основе концепции workflow (потоков выполнения), что позволяет параллельно задействовать в одном запуске несколько приложений и организовать взаимодействие между ними. Для представления workflow разработана специальная графическая нотация, которая обеспечивает возможность предоставления доступа к основным функциональностям PEG2 через визуальную оболочку, что существенно повышает эргономические показатели системы и упрощает ее использование по сравнению с существующими технологиями разработки приложений под Грид. Для управления эффективностью параллельных вычислений в PEG2 реализован ряд схем каскадной балансировки; экспериментально подтверждена их применимость.

Работа выполнена в рамках проекта «Разработка инструментальной оболочки проектирования высокопроизводительных приложений для Грид-архитектур в целях создания прикладных сервисов компьютерного моделирования и обработки данных», шифр 2007-4-1.4-20-01-025 ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2007 – 2012 годы».

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 Все-

российской научной конференции Научный сервис в сети Интернет — М.: Издательство Московского университета, 2007. — с. 134.
8. Globus Toolkit Homepage с сайта http://globus.org/toolkit/
9. Estimation of extreme wind wave heights / Lopatoukhin L.J. [et al] // Reports of World Meteorological Organization, WMO/TD, vol. 1041, 2000, 76 p