УДК 004.075

СРЕДСТВА ОПТИМИЗАЦИИ GRID-ВЫЧИСЛЕНИЙ НА ОСНОВЕ GLOBUS TOOLKIT

А.Е. Дорошенко, К.А. Рухлис

Институт программных систем НАН Украины
03187, Киев, проспект Академика Глушкова, 40,
тел.: 526 1538, dor@isofts.kiev.ua

Использование Grid-платформ достаточно распространенно в научных исследованиях. Такой тип систем позволяет использовать разнотипные вычислительные ресурсы для решения наиболее сложных научных проблем. Более того, такие системы используются в гетерогенных нестабильных сетях, где использование классических параллельных платформ типа MPI-неэффективно из-за отсутствия в последних средств мониторинга состояния, поддержки транзакций, управления задачами, поддержки средств безопасности. Однако, современные Grid-платформы имеют и ряд существенных недостатков, связанных с отсутствием таких качеств, как мобильность сервисов, каталогизаторов Grid-сервисов, качественного управления параллельными задачами и т.д. В работе предложены средства расширения популярной Grid-платформы Globus Toolkit для решения этих проблем. Проанализировано текущее состояние архитектуры Globus Toolkit и приведены результаты по реализации параллельных вычислений с использованием таких расширений.

The use of the Grid platforms in science is widely spread now. Such kind of systems allows using many types of computing systems to solve the most complicated scientific problems. Furthermore such platforms are used in heterogeneous and unstable networks, where classical MPI-like platforms is ineffective because of their lack of monitoring tools, transactions support, tasks management and low security level. However most of the Grid platforms, have problems connected with the absence of the service mobility, service directory subsystems, qualitative task scheduling and resource management, etc. The paper offers a kind of extensions for the popular Grid platform Globus Toolkit 3.2. Existing Globus Toolkit architecture and services are reviewed and the approach of the parallel programs quality improvement by use of such extensions is illustrated.

Введение

То, что теперь понимают под термином Grid, представляет собой инфраструктуру, построенную на основе Интернет и Всемирной паутины (World Wide Web), которая обеспечивает масштабируемые, безопасные, быстродействующие механизмы для обнаружения и доступа к удаленным вычислительным и информационным ресурсам [1, 2]. Использование Grid-платформ становится достаточно распространенным в современном научном мире. Такие платформы позволяют использовать сколько-нибудь существенную вычислительную единицу для решения сложных научных задач. Кроме того, Grid-системы могут применятся в слабосвязанных гетерогенных средах, где использование таких ранее разработанных платформ как MPI и PVM [3 - 5] уже малоэффективно из-за отсутствия в них средств управления состоянием распределённой системы, средств послеаварийного восстановления, поддержки транзакционности операций, низкого уровня подсистемы внутренней безопасности.

Однако, несмотря на все эти дополнительные качества Grid-платформ, в любом случае перед разработчиками параллельного (распределённого) программного обеспечения встаёт вопрос – насколько эффективно их программ использует ресурсы вычислительного комплекса, насколько они отказоустойчивы, насколько способны подстраиваться под динамически меняющуюся среду. Кроме того, чем «тяжелее» оказывается Grid-среда, тем сложнее становится концентрировать усилия на решении прикладной задачи и тем больше усилий приходится тратить на поддержку необходимой инфраструктуры выбранной среды. Особенно актуальны эти вопросы для программ, которые имеют достаточно большое время выполнения. До недавнего времени качественно решить эти проблемы удавалось лишь для узкого круга задач. При этом в каждом конкретном случае использовался свой подход, свои решения. Это послужило причиной создания целого ряда “интеллектуальных” Grid-платформ таких как OptimalGrid [6,7], JXTA [8] и других. В некоторой степени это решило те или иные проблемы, однако в Grid-платформах типа Globus Toolkit и OptimalGrid (т.е. таких, которые основаны на платформе Java) есть ещё один недостаток − малая скорость вычислений на Java сравнительно с исполняемым кодом компилированных параллельных приложений, например, с использованием интерфейса параллельного программирования MPI.

В работе предложен подход к расширению базовой архитектуры Grid-платформы Globus Toolkit (GT) свойствами мобильности сервисов, наличием каталожного сервиса, наличием средств обеспечения двунаправленного взаимодействия с обычными (nongrid) приложениями. Продемонстрировано применение такой расширенной платформы на примере параллельного решения вычислительной задачи, в том числе, в части взаимодействие с MPI.

1. Платформа Globus Toolkit

Globus Toolkit представляет собой реализацию OGSA/OGSI(Open Grid Services Architecture/Open Grid Services Infrastructure) стандарта на архитектуру и функциональность Grid-платформ, основанную на технологии веб-сервисов (рис. 1 [9]).

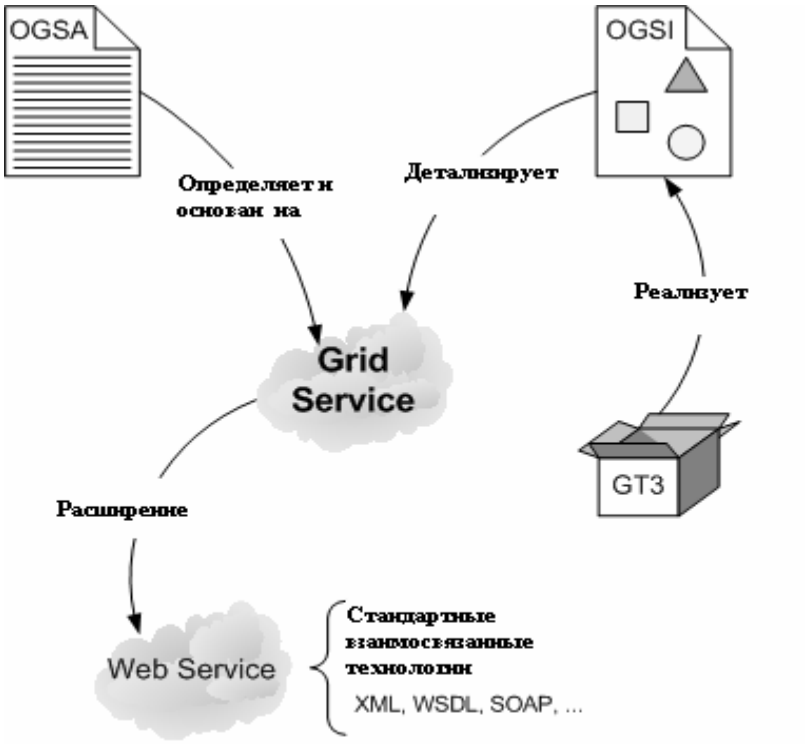

Рис 1. Отношения между grid сервисами, веб сервисами и стандартами OGSA/OGSI

Такой подход позволяет GT системам использовать уже существующие отлаженные реализации Web Service платформ в качестве базового уровня, быть в принципе переносимым под любые серверы приложений. Кроме того, хотя Globus Toolkit и оперирует Grid-сервисами, они всё же являются надстройкой над веб-сервисами, а значит, принципиально можно перенести очень большое количество существующих веб-сервисов на платформу Globus Toolkit без особых затрат. К преимуществам Grid-сервисов перед традиционными веб-сервисами относятся :

- кумулятивное изменение параметров своего состояния в процессе исполнения по вызовам клиентов;
- нерезидентность сервисов;
- наличие служебных данных окружения сервиса;
- система уведомлений;
- возможность организации сервисных групп;
- наследуемость и расширяемость описаний сервисов в WSDL [10],
- возможность управления жизненным циклом каждого экземпляра сервисов.

Типовая Grid-платформа GT имеет следующую структуру (рис. 2 [9]):

Рис. 2. Типовая Grid-платформа GT

В состав GT входят различные вспомогательные сервисы для управления безопасностью, средства взаимодействия с Grid-Ftp, средства пакетного запуска задач.

Взаимодействие GT с grid сервисом происходит следующим образом (см. рис. 3).

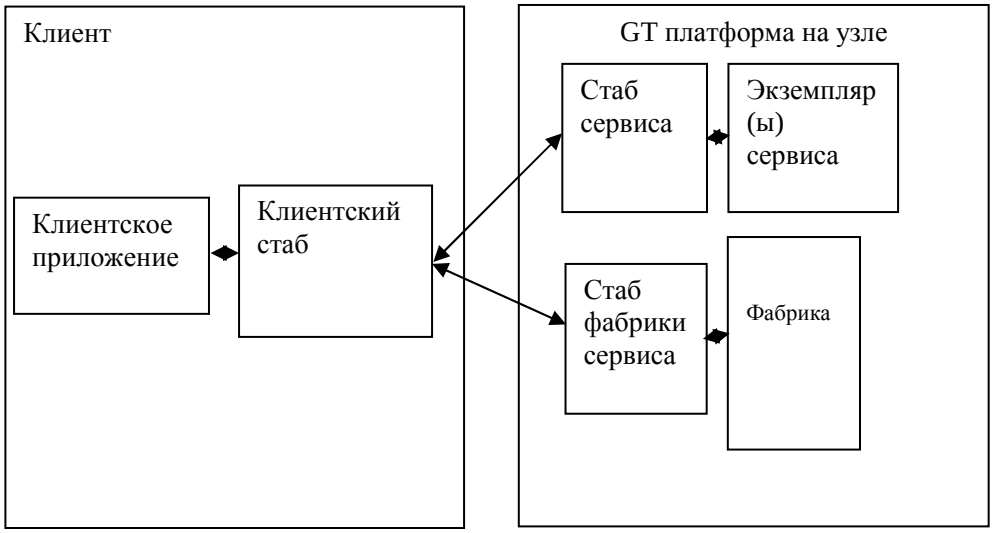

Рис. 3. Взаимодействие клиента с GT grid сервисом

Клиентское приложение (клиент) через заранее сгенерированный при создании сервиса промежуточный объект (стаб) обращается к фабрике сервисов (также через стаб) для создания экземпляра нужного сервиса. После проверки системой безопасности GT прав на выполнение этой операции клиент получает от фабрики сервисов GSH (Grid Service Handle) уникальный идентификатор экземпляра сервиса. Через этот идентификатор посредством клиентского стаба уже можно вызывать методы требуемого сервиса.

Несмотря на достаточно продуманную реализацию и наличие целого ряда дополнительных возможностей, повышающих качество распределённого приложения, у GT есть целый ряд недостатков, например, отсутствуют средства мониторинга состояния узлов, средства обеспечения мобильности пользовательских сервисов, сервисы каталогов и т.д. Отсутствие таких средств накладывает достаточно серьёзные ограничения. Например, отсутствие средств мониторинга состояний узлов ведёт к тому, что при распределении подзадач по узлам кластера пользователи не имеют ни малейшего представления о текущей производительности его узлов, ни о качестве (пропускной способности) каналов связи между ними. Отсутствие мобильности приводит к тому, что для запуска необходимых пользовательских сервисов на узлах приходиться предварительно копировать их туда вручную и устанавливать. Отсутствие сервисов каталогов является серьёзным ограничением масштабируемости. Из-за этого сервисы на одном узле не имеют никакого представления обо всех доступных на кластере сервисах и не могут их вызывать.

Для устранения этих недостатков в данной работе предложены дополнительные сервисы GT.

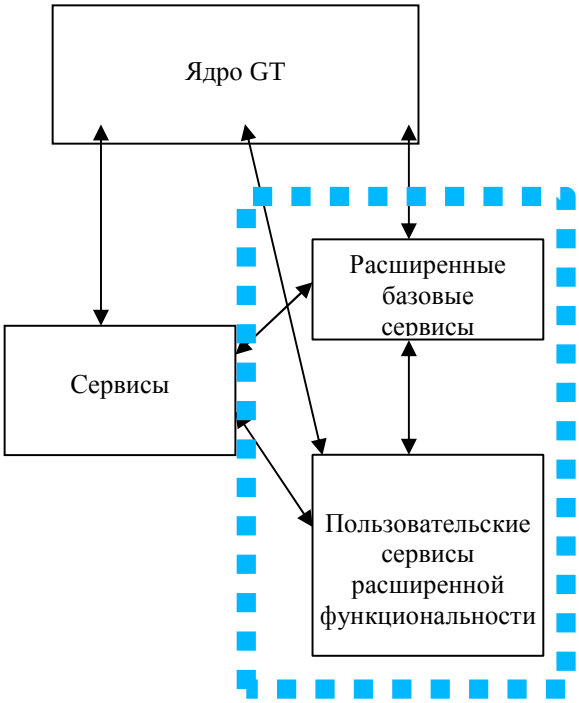

Рис. 4. Расширенная Grid-инраструктура одного узла

2. Функциональное расширение платформы Globus Toolkit

В данной работе классическая схема инфраструктуры узла grid платформы (рис. 2) расширяется добавлением в нее расширенных базовых сервисов и пользовательских сервисов расширенной функциональности (рис. 4).

Расширенные базовые сервисы представляют собой дополнительный программный слой в архитектуре GT, в котором содержатся следующие компоненты:

установщик инсталляций grid сервисов (GAR) на узел;
резидентный агент для измерения производительности узла;
сервис каталогов;
прикладной программный мост (ППМ) для связи с обычными приложениями.

Установщик инсталляций Grid-сервисов GAR предназначен для обеспечения мобильности Grid-сервисов, а также для обмена файлами между вычислительными узлами. Эта подсистема в первую очередь используется интерфейсным узлом (рис. 6), с которого поступают задачи. Резидентный агент для измерения производительности узла может использоваться для мониторинга состояния узла при распределении задач по узлам инфраструктуры. Сервис каталогов предназначен для поиска установленных сервисов на узле, для создания новых сущностей сервисов (например, при вызовах сервисов с другого узла или из обычных приложений). Прикладной программный мост используется для связи с обычными приложениями и состоит из нескольких компонент – точек входа-выхода.
Его структура и схема работы показана на рис. 5.

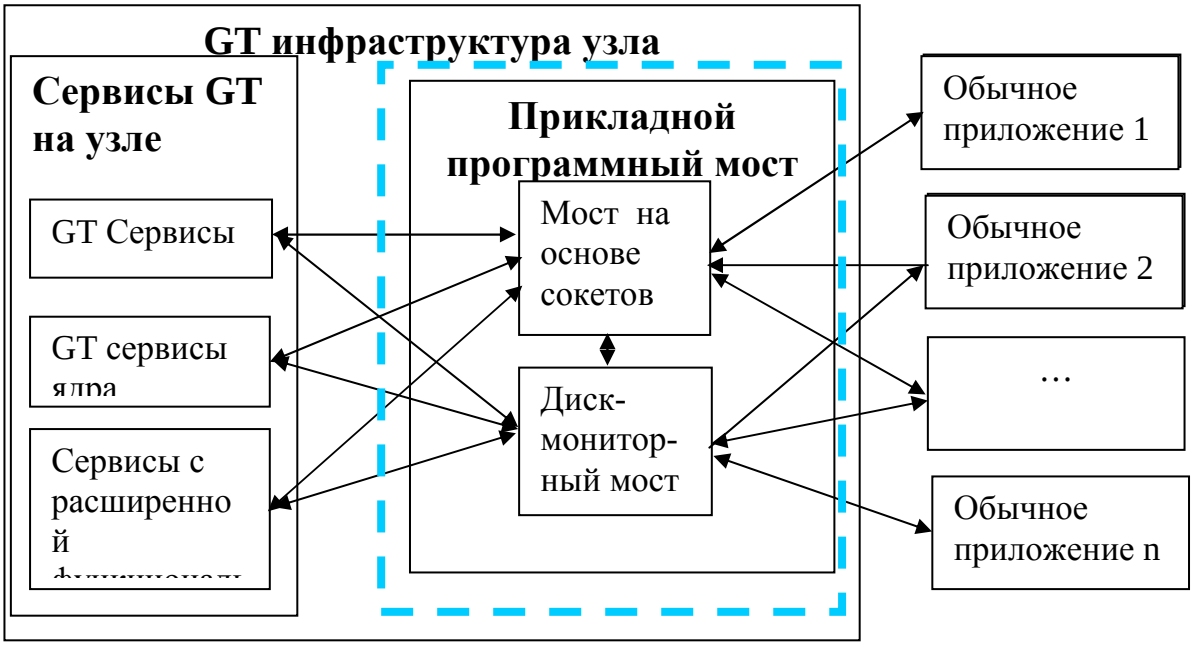

Рис. 5. Структура и схема работы прикладного программного моста

ППМ состоит из двух подчастей – из моста на основе сокетов (СМ) и диск-мониторного моста (ДММ). СМ представляет собой сервис, который принимает команды и возвращает результат через сокет. ДММ общается с клиентом несколько другим образом. Чтобы вызвать ДММ, клиентское приложение должно создать в определённом каталоге xml-файл с задачами. В этот каталог клиент должен предварительно положить все необходимые для вызовов данные. ДММ с определённым интервалом времени просматривает указанный каталог и при возникновении там файла с задачей выполняет её, возвращая результаты в этот же каталог.

Кроме вышеописанного способа работы, когда клиент использует ППМ для вызовов GT-сервисов, возможен ещё один вариант работы, в обратном порядке, т.е. возможно, чтобы GT-сервисы через мосты СМ или ДММ вызывали обычные приложения.

Таким образом, кроме расширения структуры каждого конкретного узла модифицируется структура связей узлов, а на уровне всей Grid-инфраструктуры к вычислительным узлам и узлу с графическим интерфейсом пользователя (ГИП) добавляется командный узел (рис. 6).

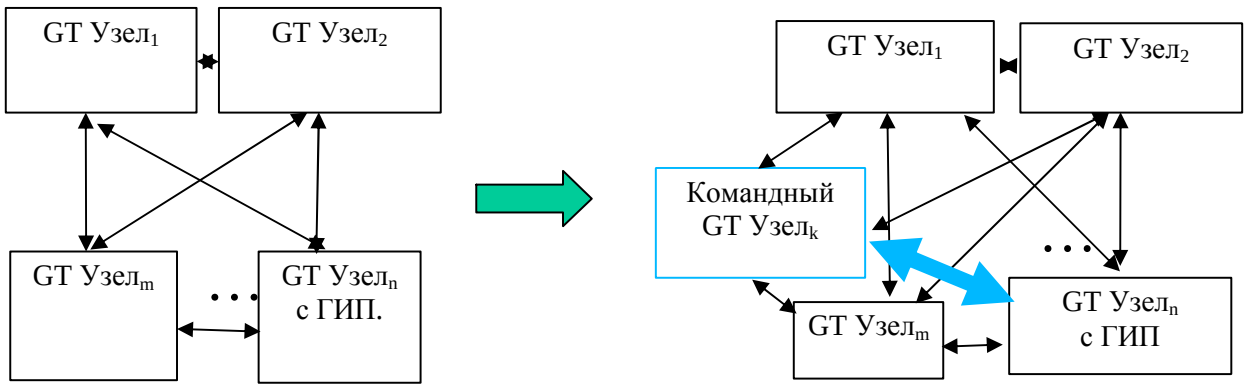

Рис. 6. Изменение структуры связей между GT-узлами

Если стандартная архитектура одноранговая (за тем лишь исключением, что один из узлов может также использоваться как интерфейсный), то в модифицированной архитектуре выделяется один из узлов для управления всей Grid-инфраструктурой GT. В функции командного узла входит:

- хранение и поддержание списка доступных узлов Grid-платформы;
- хранение и поддержание списка свободных узлов;
- сбор статистики по каждому из доступных узлов;
- репозиторий инсталляций сервисов (Gar-файлов);
- распределение задач по узлам.

Процесс выполнения задачи из-за этого несколько видоизменяется. Теперь, в отличие от стандартного подхода GT, для выполнения задачи нет потребности сперва устанавливать все необходимые Grid-сервисы на выделенные для решения узлы. Нет необходимости вручную копировать на узлы и наборы данных или дополнительные программы для решения задач. Запуск задачи на выполнение происходит следующим образом:

- пользователь создает соответствующий архив (zip) с задачей на выполнение, набором конфигурационных файлов и дополнительными ресурсами и программами;
- через интерфейсный узел задача запускается на выполнение;
- интерфейсный узел принимает от пользователя задачу, проводит необходимые проверки безопасности и корректности архива с задачей;
- интерфейсный узел передаёт задачу командному узлу;
- командный узел прорабатывает конфигурационные файлы задачи, производит проверку производительности свободных узлов и выделяет узлы под задачу;
- командный узел организовывает развёртывание необходимой инфраструктуры на выделенных под решение этой задачи узлах и запускает на них задачу на выполнение;
- после получения решения командный узел тем или иным образом возвращает результат пользователю.

3. Пример использования функционально расширенной платформы Globus Toolkit

Grid-системы применяются для решения достаточно широкого круга научных задач, например, в биологии, медицине, астрономии, метеорологии, физике и т.д. [1, 11]. Во многих из этих применений используются операции линейной алгебры, например, умножение матриц. Несмотря на достаточно хорошо проработанные алгоритмы параллельного матричного умножения, их эффективная реализация всегда являлась желанной для разработчиков. Однако есть целый ряд проблем, связанных с этими алгоритмами.

Одной из главных проблем, связанных с умножением матриц, побудивших использовать параллельные системы, является достаточно большой размер самих матриц. Умножение таких больших матриц требует кроме вычислительных ресурсов ещё и достаточно ёмкие носители данных, а также скоростные каналы связи между узлами вычислительного кластера. Кроме того, для параллельного умножения матриц необходимо организовывать ещё и подсистему обмена блоками данных. Рассмотрим классический алгоритм параллельного умножения двух квадратных матриц [11].

Пусть `A,B` – квадратные матрицы порядка `(size*size)`, причем `size = m*blk`, и требуется вычислить матрицу `C=A*B`, используя для этого `m*m` узлов. Исходные матрицы `A` и `B` можно считать составленными из m*m квадратных блоков размерностью `(blk*blk)`. Положение этих блоков в исходных матрицах описывается парой индексов, каждый из которых изменяется в пределах от 0 до (m-1). Обозначим блоки матрицы A как `A[i,j] (i,j = 0...(m-1))`, а элементы матрицы `A – a[i,j] (i,j = 0...(size-1))`. Например, при `size = =4, m = 2, blk = 2`, матрица `A`:

$$A = \begin{matrix} a[0,0] & a[0,1] & a[0,2] & a[0,3] \\ a[1,0] & a[1,1] & a[1,2] & a[1,3] \\ a[2,0] & a[2,1] & a[2,2] & a[2,3] \\ a[3,0] & a[3,1] & a[3,2] & a[3,3] \end{matrix}$$

И, соответственно, блоки:

$$A[0,0] = \begin{matrix} a[0,0] & a[0,1] \\ a[1,0] & a[1,1] \end{matrix}; A[0,1] = \begin{matrix} a[0,2] & a[0,3] \\ a[1,2] & a[1,3] \end{matrix};$$

$$A[1,0] = \begin{matrix} a[2,0] & a[2,1] \\ a[3,0] & a[3,1] \end{matrix}; A[1,1] = \begin{matrix} a[2,2] & a[2,3] \\ a[3,2] & a[3,3] \end{matrix}$$

Тогда, матрица A может быть представлена как:

```
        A[0,0]  A[0,1]
A =
        A[1,0]  A[1,1]
```

Известно, что классическую формулу поэлементного умножения двух матриц:

```
    c[i,j] = a[i,0]*b[0,j] + a[i,1]*b[1,j] + ... + a[i,
size-1]*b[size-1,j],( i,j = 0...(size-1) )
```

можно заменить вычислением блоков матрицы C по формуле

```
    C[i,j] = A[i,0]*B[0,j] + A[i,1]*B[1,j] + ... + A[i,m-1]*B[m-1,j],
( i,j = =0...(m-1) );
```

в которой операции над блоками являются матричными операциями.

Такой подход к параллельному умножению матриц является распространённым в среде сильносвязанных вычислительных кластеров и суперкомпьютеров. Он прекрасно подходит для систем с одинаковыми вычислительными возможностями узлов и специализированными каналами связи между ними. Однако в этом заключается и его недостаток. Применение такого алгоритма для слабосвязанных гетерогенных сред сопряжено с теми трудностями, что каждый вычислительный узел уникален по своим вычислительным возможностям, может иметь свою собственную аппаратную архитектуру, а различных способов связи между узлами как по скорости, так и типу может быть очень много. В этом случае деление на одинаковые по размеру блоки приведёт к неравномерному выполнению вычислений и возникновению ситуаций несбалансированности приложений. Кроме того, в связи со свойством слабой связанности таких сред критичным становится также мониторинг состояния вычислительных узлов, наличие средств восстановления вычислений и т.д.

Наличие вышеописанных недостатков классического алгоритма параллельного умножения матриц в случае гетерогенной слабосвязанной среды логически ведёт к другому решению – разбиению матриц на неравные по размеру блоки. Причём размер таких блоков должен соответствовать итоговой производительности узла, на котором они должны выполняться. Именно такой подход и был выбран для реализации умножения матриц на платформе Globus Toolkit с модифицированной архитектурой.

Реализация на модифицированном GT принимает во внимание два фактора для подсчёта итогового рейтинга узла – производительность его процессора, а также скорость канала связи с ним. При этом пользователь может задать в конфигурационном файле веса этих двух параметров в итоговой оценке, т.е.:
$\mathtt{NodeMark}=w_{net}*\mathtt{NetMark}+w_{cpu}*\mathtt{CpuMark}$.
В случае если параметры не заданны пользователем, система сама рассчитывает веса, основываясь на заложенных в неё эмпирических данных (подобный подход применяется и в OptimalGrid [7]).

После получения задачи командный узел проверяет доступность необходимого количества вычислительных узлов (пользователь может задать минимально необходимое количество узлов в конфигурации задачи). В случае доступности такого количества узлов, он анализирует текущую производительность свободных узлов. Со всех доступных на данный момент узлов выбирается необходимое количество наиболее производительных узлов, после чего матрица разбивается в соответствии с итоговой производительностью каждого узла. Далее, с помощью средств модернизированного GT задача разворачивается на этих узлах и выполняется. После чего командный узел собирает результаты умножения блоков матриц в единую результирующую матрицу и возвращает её пользователю.

Для исследования возможностей расширенного GT разработаны и реализованы два варианта решения задачи. Первый заключается в создании отдельного GT-сервиса, который на вход получает блоки матриц и имеет единственную общедоступную точку входа - подпрограмму умножения двух матриц. Этот сервис компилируется в единый архивный файл и разворачивается через расширенные возможности платформы на узлах.

Второй вариант реализован через приложение MPI и предназначен для использования в тех ситуациях, когда необходимо организовать одновременную работу обычных высокопроизводительных приложений под управлением Globus Toolkit. Сама программа умножения двух матриц является приложением MPI. Она считывает параметры из файлов с матрицами, и результат возвращает также в файл. Взаимодействие GT узла с этой программой происходит через расширенный мост ДММ (рис. 7).

В качестве подсистемы передачи файлов используется SSH\SFTP сервисы. После получения задачи и разбиения матриц на блоки, командный сервис через SFTP передаёт на каждый узел файл с приложением MPI, и файлы данных матриц. После чего, с помощью SSH сервис запускает на каждом из этих узлов MPI программу

и переходит в ждущий режим, собирая через SFTP результаты по мере их поступления. После окончания работы всех узлов командный сервис «собирает» результирующую матрицу и возвращает её клиенту.

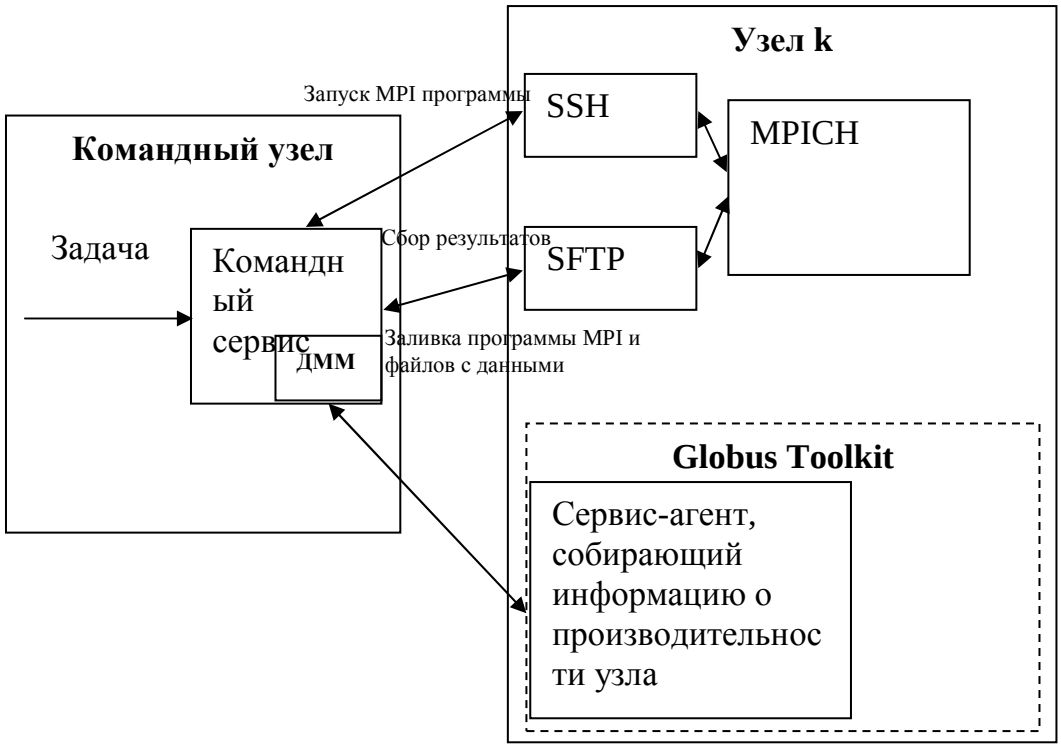

Рис. 7. Реализация умножения двух матриц с использованием GT и MPI

Сравнение быстродействия, в случае, когда решающим фактором для скорости умножения является скорость процессора можно видеть на рис. 8. Тесты проводились на локальной сети, стандарта Gigabit Ethernet, состоящих из компьютеров с процессорами Pentium 4 2.8 – 3Ггц, объёмом ОЗУ 512 Мб – 2 Гб.

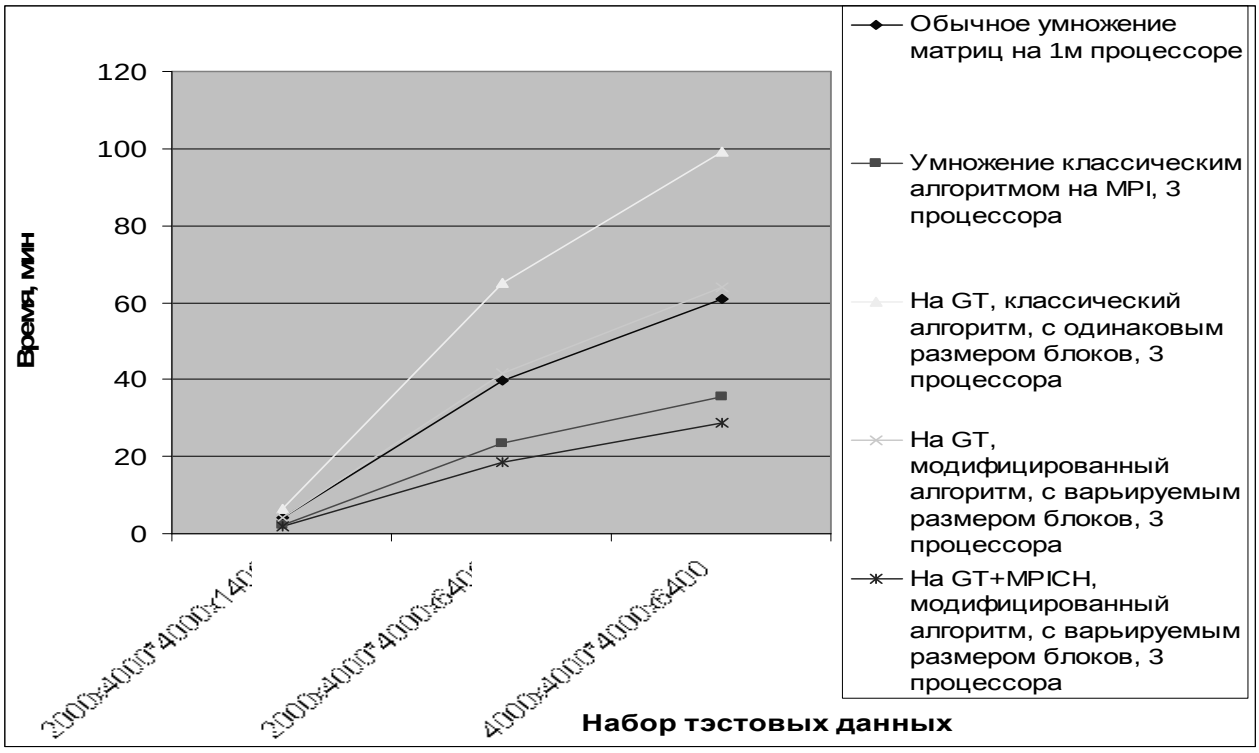

Рис. 8. Время работы реализаций

В случае же, когда приоритетным является быстродействие каналов связи (ввиду относительно больших размеров матриц), ускорение по отношению к реализации на одном узле выглядит следующим образом (рис. 9.).

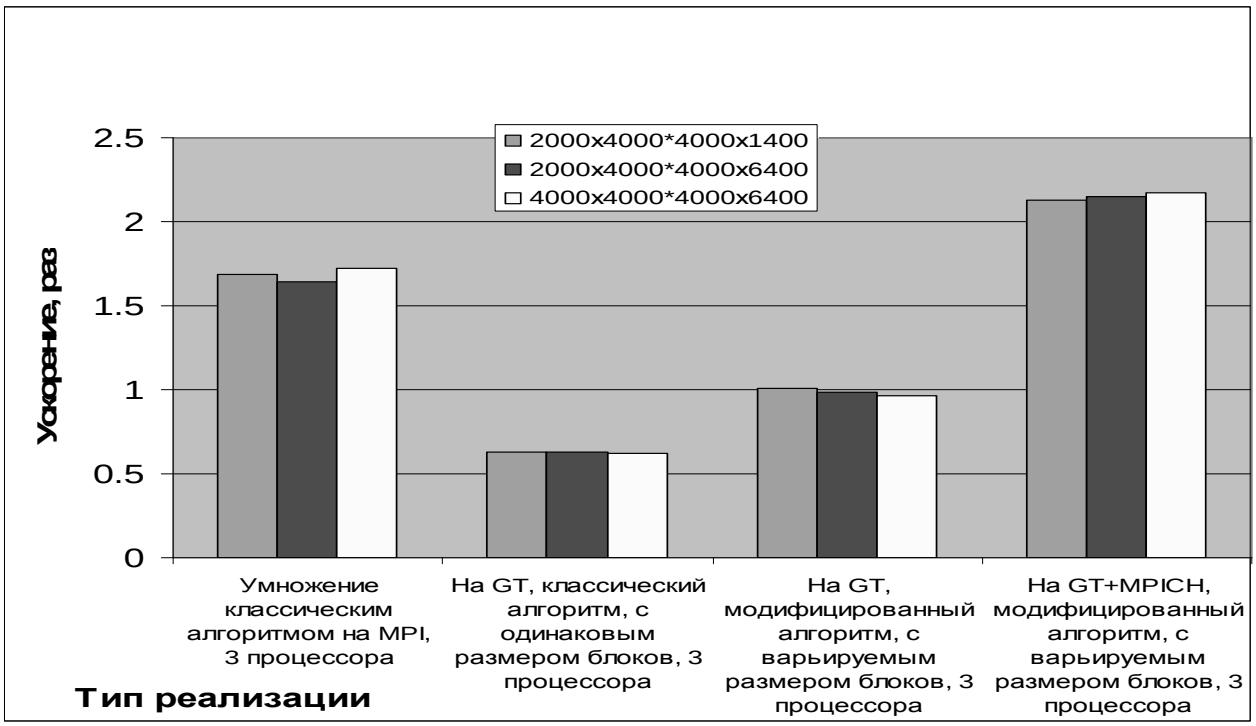

Рис. 9. Ускорение вычислений по отношению к реализации классическим алгоритмом на 1м узле

По результатам видно, что применение разработанных нами дополнений к стандартной GT-платформе ведёт к увеличению скорости вычислений в 1.5 раза в случае использования реализации на Java, и в 1.3 раза при использовании MPI под управлением модифицированного GT на гетерогенном кластере. Синхронность работы, т.е. разброс по времени окончания работы между самым медленным и самым быстрым узлом, также улучшилась, а разброс снизился в 1.6 – 2.2 раза (рис. 10).

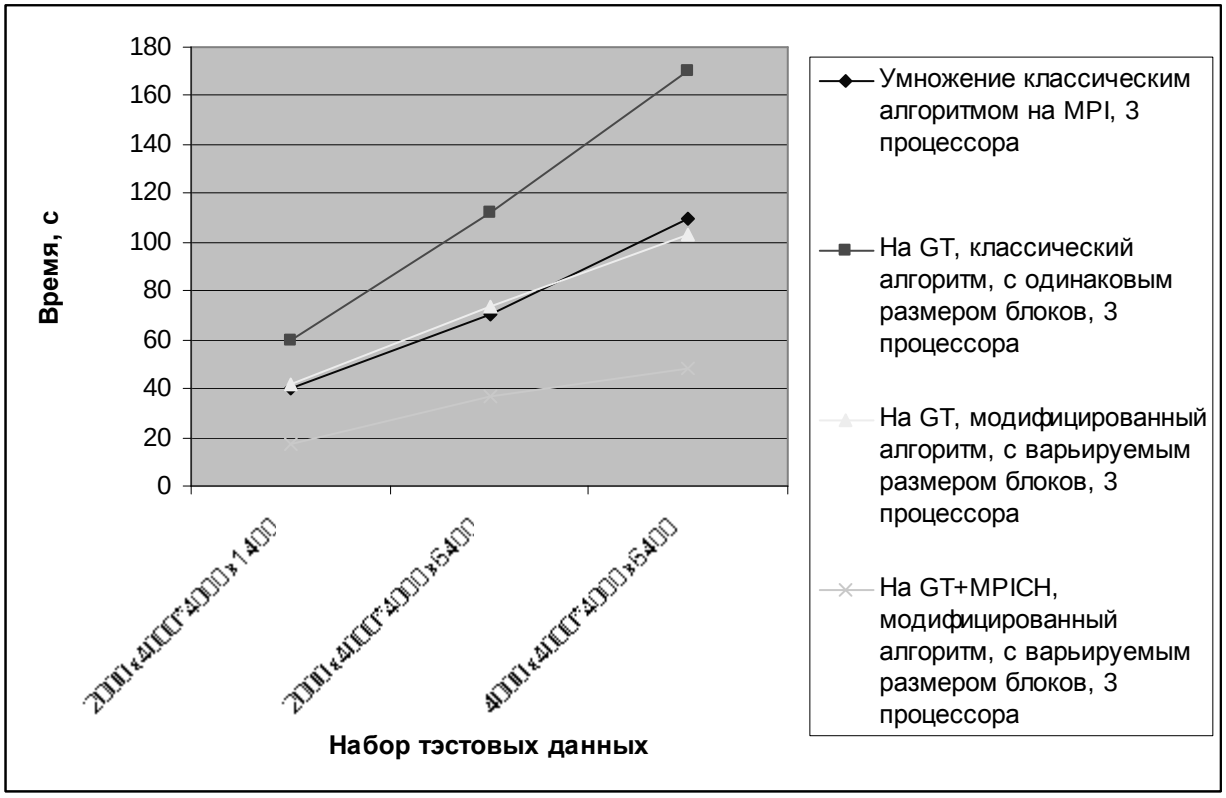

Рис. 10. Синхронность работы реализаций

Кроме того, использование расширенной версии Globus Toolkit сократило время и усилия на развёртывание задачи, настройку окружения, организацию взаимодействия между узлами по сравнению со стандартной версией GT.

Выводы

Применение grid-систем позволяет значительно повысить скорость и качество вычислений, особенно в случае использования слабосвязанных гетерогенных вычислительных комплексов. В работе описана архитектура современной Grid-системы на основе спецификации OGSA – Globus Toolkit 3.2 . Рассмотрены слабые места этой системы и предложены средства для повышения производительности и интеллектуальности вычислений с использованием Globus Toolkit.

Разработанный в работе пакет дополнительных средств привносит в классическую архитектуру grid систем такие компоненты как мобильность сервисов, подсистему мониторинга состояния вычислительных узлов, интеллектуализацию выделения вычислительных ресурсов для решения задачи, сервисы каталогов и повышает степень автономности платформ.

Вместе с тем следует отметить, что предложенный подход имеет и свои органичения, а именно: отсутствие полноценного искусственного интеллекта при принятии решений о разбиении задачи, а также увеличение степени централизации платформы. Эти проблемы могут быть в той или иной степени решены с применением технологий экспертных и мультиагентных систем.

1. *Дорошенко А.Е., Алистратов О.В., Тырчак Ю.М. Розенблат А.П., Рухлис К.А.* Системы Grid вычислений – перспектива для научных исследований // Проблемы программирования. – 2005. - № 1. – С. 14 –38.
2. *IBM* Autonomic Computing. – http://www.research.ibm.com/autonomic
3. *Воеводин В.В., Воеводин В. В.* Параллельные вычисления – СПб.: БХВ-Петербург, 2002. – 608 с.
4. *Немнюгин С., Стесик О.* Параллельное программирование для многопроцессорных вычислительных систем. СПб.: БХВ-Петербург, 2002. – 400 с.
5. *Эндрюс Г.Р.* Основы многопоточного, параллельного и распределённого программирования. М.: ИД «Вильямс», 2003. – 512 с.
6. *Berman, F., Fox, G. and Hey (eds), T.* Grid Computing: Making the Global Infrastructure a Reality. Chichester: John Wiley & Sons, 2003. – 1060 p.
7. *Optimal* Grid:Overview. -- http://www.alphaworks.ibm.com/tech/optimalgrid
8. *Sun* JXTA. – http://jxta.org
9. *Sotomayor B.* The Globus Toolkit 3 Programmer's Tutorial.
10. www.w3.org
11. *Хабибуллин И. Ш.* Разработка Web-служб средствами Java. –СПб.: БХВ-Петербург, 2003. – 400 с.
12. *Дорошенко А.Е.,* Математические модели и методы организации высокопроизводительных параллельных вычислений. Алгебродинамический подход.- Киев: Наук. думка, 2000. – 177 с.