

РОССИЙСКОЙ АКАДЕМИИ НАУК
ОРДЕНА ЛЕНИНА
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
ИМЕНИ М.В.КЕЛДЫША

П.С.БЕРЕЗОВСКИЙ, В.Н.КОВАЛЕНКО

Состав и функции системы
диспетчеризации заданий
в гриде с некластеризованными ресурсами

Москва
2007 г.

УДК 519.68

П.С.Березовский, В.Н.Коваленко. Состав и функции системы диспетчеризации заданий в гриде с некластеризованными ресурсами.

На сегодняшний день известен ряд систем для организации вычислений на множестве пространственно распределённых компьютеров путём занятия процессорного времени, которое не используется их владельцами. Каждая из них внесла вклад в развитие методов распределённого компьютеринга. В то же время, ни одна из рассмотренных систем не решает всех задач, существенных для создания полноценных грид-инфраструктур. В работе определены требования к программному обеспечению грида с формой организации ресурсов, которая не предполагает кластеризацию компьютеров, и предложена архитектура системы диспетчеризации заданий.

Ключевые слова: грид, распределенные вычисления, Peer-to-Peer системы, управление ресурсами.

P.S.Berezovskiy, V.N.Kovalenko. Structure and Functionality of the Job Management System for Grid with Non-Clustered Resources.

Nowadays, there are known a number of systems, which are employed for the management of a set of various distributed computers and utilization of their idle CPU cycles. Each of these systems has made an important contribution to the development of distributed computing methods. At the same time, none of them solves all the problems that are essential for creation of full-fledged grid-infrastructures. In the paper the requirements for middleware, that is capable to integrate globally distributed separate non-clustered computers and complies with the grid standards, are defined. Also, the architecture of the job management system for such resource infrastructure is proposed.

Key words: grid, distributed computing, Peer-to-Peer systems, resource management.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект 06-07-89111-а) и гранта Научной школы (НШ-383.2006.9).

Содержание

| | |
|-------------------------------------------------------------------------------------------|----|
| 1. ВВЕДЕНИЕ | 4 |
| 2. ОБЛАСТЬ ПРИМЕНЕНИЯ ОДНОУРОВНЕВОГО ГРИДА..... | 6 |
| 3. СУЩЕСТВУЮЩИЕ ПРОГРАММНЫЕ РЕШЕНИЯ..... | 8 |
| 3.1. Проекты @Home и платформа VOINC..... | 8 |
| 3.2. P2P-подход. Системы Cluster Computing On the Fly и OurGrid..... | 10 |
| 3.3. Entropia..... | 13 |
| 3.4. Condor..... | 14 |
| 4. ВОЗМОЖНОСТЬ ПРИМЕНЕНИЯ СУЩЕСТВУЮЩИХ СИСТЕМ ДЛЯ ПОСТРОЕНИЯ ОДНОУРОВНЕВОГО ГРИДА..... | 16 |
| 5. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ ОДНОУРОВНЕВОГО ГРИДА..... | 18 |
| 6. АРХИТЕКТУРА СИСТЕМЫ ДИСПЕТЧЕРИЗАЦИИ..... | 20 |
| 6.1. Диспетчер одноуровневого грида | 21 |
| 6.2. Агент | 25 |
| 6.3. Пользовательский интерфейс | 27 |
| 7. ЗАКЛЮЧЕНИЕ | 28 |
| 8. ЛИТЕРАТУРА..... | 28 |

1. ВВЕДЕНИЕ

Концепция распределённой среды грид становится всё более популярной. В рамках известных типов грида (семантический грид, грид данных, информационный грид и т.д.) появляются новые формы, но наиболее активно развивается вычислительный грид, который рассматривается в настоящей работе. Возможные реализации вычислительного грида мы будем различать по:

- типу использования ресурсов (отчуждаемые и неотчуждаемые ресурсы);
- типу объединения ресурсов (кластеризованные и некластеризованные ресурсы);
- классу обрабатываемых заданий (однопроцессорные, параллельные и сериализуемые).

В настоящей работе мы будем рассматривать грид, состоящий из неотчуждаемых некластеризованных ресурсов и ориентированный на обработку простых заданий, которым для выполнения требуется один процессор, а также сериализуемых заданий, то есть набора простых заданий, которые решают одну задачу, но не взаимодействуют между собой в процессе выполнения.

Наибольшее распространение получил метод построения грид-инфраструктур из находящихся в разных точках сети кластерных узлов, в которые объединяется множество компьютеров, принадлежащих одному административному домену. Такой способ организации будем называть двухуровневым (или горизонтально интегрированным) [1]. При двухуровневой организации ресурсы объединяются в кластеры и находятся под управлением локального менеджера (например, PBS [2] или LSF [3]), который осуществляет непосредственное распределение заданий и их запуск на вычислительных установках. Эта форма наиболее развита, так как сама концепция грида зародилась в организациях, обладающих сформированной кластерной инфраструктурой. Двухуровневая организация ресурсов поддерживается наиболее распространённым программным обеспечением грида, например, инструментально-базовой системой Globus Toolkit [4] и основанным на ней комплексом gLite [5]. Существенно, что способ распределения (виртуализации) ресурсов, реализованный в gLite и большинстве других комплексов, также основанных на Globus Toolkit, фактически предполагает, что компьютеры должны полностью выделяться в грид и не могут использоваться владельцами – сотрудниками тех организаций, в которых они установлены. Такие ресурсы мы называем отчуждаемыми.

Помимо кластеров в сети имеются некластеризованные вычислительные ресурсы – рабочие станции пользователей, серверы и пр., обладающие большой суммарной производительностью, в то время как их средняя загрузка очень мала. Поскольку таких компьютеров довольно много, они представляют большой интерес для пользователей грида, которые могли бы решать на них свои задачи, но далеко не всегда владельцы компьютеров имеют возможность

организовывать и поддерживать кластерную инфраструктуру. Кроме того, использование этих компьютеров в гриде составляет проблему в силу двух обстоятельств: 1) владельцы выполняют на них собственные программы и 2) включают/выключают компьютеры в непредсказуемые моменты.

Сейчас в основном такие ресурсы используются для решения сверхкрупных по требуемому времени счёта задач в рамках отдельных проектов [6], [7], когда владельцы ресурсов на короткий период времени объединяют имеющиеся мощности, а затем либо прекращают своё участие в данной деятельности, либо подключаются к другой подобной инфраструктуре для решения очередной задачи.

Ограниченные возможности системного программного обеспечения, применяемого в проектах, не позволяют квалифицировать его как средство поддержки грида. Однако популярность этой формы распределённых вычислений демонстрирует, что способ организации грида на ресурсной базе из отдельных некластеризованных компьютеров представляет интерес с практической точки зрения. В работе [1] предложены основные принципы построения грид-инфраструктур с такой организацией и введен термин одноуровневый грид (или с вертикальной интеграцией). В данной статье ставится вопрос об архитектуре и свойствах программного обеспечения одноуровневого грида.

Главное отличие двух типов – двухуровневого и одноуровневого – грида заключается в режиме использования ресурсов, от которого существенно зависят способы управления исполнительными узлами. В первом случае сформированная кластерная инфраструктура обычно целиком предоставляется для нужд грида, то есть отчуждается от владельца. В случае одноуровневого грида пространственно распределённые исполнительные компьютеры целиком принадлежат своим владельцам, а задания грида поступают на счет только в том случае, если они не мешают выполнению локальных заданий. При этом владелец должен сохранять способность полностью контролировать свой компьютер, определяя условия предоставления его ресурсов для обработки задач грида. В предлагаемых решениях разделение ресурсов между владельцем и гридом рассматривается как основной вариант функционирования. В то же время сама по себе одноуровневая организация не накладывает ограничений на режим использования компьютера и может применяться для отчуждаемого режима – в котором компьютер выделяется в грид целиком. Однако в любом случае предполагается, что компьютеры включаются в грид без кластеризации.

В данной статье рассматриваются существующие программные решения для объединения некластеризованных ресурсов, а также предлагается способ построения системы диспетчеризации, осуществляющей управление заданиями в гриде, состоящем из некластеризованных неотчуждаемых ресурсов.

2. ОБЛАСТЬ ПРИМЕНЕНИЯ ОДНОУРОВНЕВОГО ГРИДА

Одноуровневая архитектура имеет два преимущества. О первом речь шла выше: позволяя задействовать для задач грида холостые циклы процессоров, она даёт возможность создавать гриды на основе существующей ресурсной базы, в том числе из персональных и домашних компьютеров.

Второе преимущество – в её направленности на упрощение создания гридов. Для поддержки функционирования одноуровневого грида необходим управляющий центр – диспетчер, однако он один на всё множество глобально распределённых компьютеров, интегрируемых в грид. Диспетчер нужен и для двухуровневого грида, но, помимо того, в каждом ресурсном узле должна быть установлена система управления кластером и службы доступа из грида. В одноуровневом гриде от владельцев ресурсов-компьютеров требуется лишь установка компактного и просто конфигурируемого программного обеспечения, после чего компьютер может использоваться для обработки задач грида. В любом случае, в деле создания и поддержки функционирования грида необходим профессиональный подход, но в одноуровневой архитектуре не требуется наличия специального персонала, который бы занимался обслуживанием исполнительных ресурсов в связи с их включением в грид.

Рассматривая одноуровневый грид как широко доступное средство дистанционной работы с ресурсами, можно указать несколько сценариев его применения.

1. Создание распределённых инфраструктур высокой пропускной способности. Одноуровневый грид сохраняет возможность создания широкомасштабных грид-инфраструктур, которые в современной практике строятся на основе локальной кластеризации ресурсов. Необходимо, однако, отметить ограничение на класс задач, которые связаны с неотчуждаемостью ресурсов: становится невозможным обработка параллельных (многопроцессорных) приложений ввиду того, что процессы, находящиеся на разных компьютерах, будут выполняться дискретно – в периоды малой активности их владельцев – и несогласованно друг с другом. По той же причине полное время обработки обычного задания на исполнительном компьютере не совпадает с требуемым чистым процессорным временем. В связи с этим, одна из наиболее важных проблем, решаемых программными средствами грида – обеспечение гарантированного завершения задания в заданное время.

В таких условиях наиболее привлекательным выглядит применение масштабных одноуровневых гридов для выполнения серийных расчётов в виде набора независимых заданий, которые могут обрабатываться параллельно на разных ресурсах, не обмениваясь данными. В этом смысле их можно рассматривать как части одного слабо связанного параллельного задания.

2. Объединение ресурсов в рамках временных коллективов. Грид чаще всего ассоциируется с рекордными по необходимым вычислительным ресурсам задачами. Простота одноуровневой организации может сделать грид

обыденным средством для решения более широкого класса задач, для которых требуется кратное (а не на порядки) увеличение мощности по сравнению с современными товарными компьютерами. Для этого необходимо, чтобы один центр управления гридом был способен поддерживать деятельность множества небольших коллективов, в которые объединяются лица, предоставляющие ресурсы, и лица, их использующие. Такие коллективы могут образовываться динамически, и их можно рассматривать как аналоги крупных виртуальных организаций современных гридов. Диспетчер одноуровневого грида должен обеспечивать автономность подобных “виртуальных организаций” в рамках объединённой ресурсной инфраструктуры многих таких организаций.

3. Персональный грид. Вычислительные средства, которыми располагает отдельный пользователь, как правило, ограничены единственным компьютером. Накопление ресурсного парка создаёт предпосылки для преодоления этого “барьера одного компьютера”, но необходима адекватная техническая и технологическая поддержка вычислительной деятельности в более сложной среде, состоящей из нескольких компьютеров. В качестве средства, обеспечивающего такую поддержку, может выступать одноуровневый грид. Владелец нескольких компьютеров может подключить их к гриду и образовать персональную виртуальную организацию, ограничивая доступ к своим ресурсам всем, кроме себя самого. В результате он получает общую точку доступа ко всей совокупности компьютеров – через управляющий центр грида, который обеспечивает эффективную балансировку их загрузки. Аналогичный эффект можно получить и от кластеризации компьютеров, но подход грида исключает проблемы обслуживания кластера.

4. Предоставление дистанционного доступа к приложениям. Понятие ресурса в гриде является очень широким и не ограничивается только системными ресурсами компьютеров (процессор, память, дисковое пространство). Ресурсом также может являться, например, устройство, подключенное к сети, а также любое приложение, которое по каким-либо причинам не может быть установлено у всех, кто хочет обрабатывать с его помощью свои данные. Путём подключения к гриду владелец компьютера, на котором установлено такое приложение, может предоставить к нему доступ и определить круг лиц, которые могут им пользоваться. Альтернативой может служить оформление приложения в виде грид-службы, но этот вариант более сложен.

Приведённые сценарии использования одноуровневой организации ресурсов свидетельствуют о наличии ситуаций, в которых применение такой модели даёт дополнительные возможности при решении вычислительных задач. Содержание работы посвящено вопросам создания программного обеспечения одноуровневого грида, реализующего взаимодействие участников инфраструктуры, безопасность, распределение ресурсов, передачу файлов и пр.

3. СУЩЕСТВУЮЩИЕ ПРОГРАММНЫЕ РЕШЕНИЯ

На сегодняшний день практика объединения компьютеров, находящихся в персональном владении, достаточно распространена. Задача объединения решается как в географически распределённой среде, так и в рамках одного предприятия.

При организации подобных инфраструктур можно выделить три подхода: первый основан на создании проектов, к которым подключаются исполнительные компьютеры; второй подход состоит в применении P2P-технологий [8] и объединении исполнительных компьютеров в одноранговые сети; третий подход заключается в реализации корпоративных систем.

3.1. Проекты @Home и платформа BOINC

Одним из первых примеров применения одноуровневой схемы организации пространственно распределённых ресурсов стал проект SETI@Home Калифорнийского университета, направленный на решение задачи поиска внеземного разума. Участникам проекта предлагается скачать с веб-сайта университета небольшую программу, работающую вместо хранителя экрана в те периоды, когда компьютер не используется его владельцем. Эта программа получает данные радиотелескопических наблюдений с сервера, анализирует их и отправляет результат обратно. Радиотелескоп генерирует большое количество информации, но она может быть разбита на небольшие порции, которые могут быть обработаны независимо с помощью одного приложения. Приложение меняется довольно редко, поэтому закачивается на исполнительный компьютер однократно и обновляется по мере необходимости.

Позднее появилось множество проектов из серии @Home, предназначенных для решения других прикладных задач, обладающих похожими свойствами. Разработки университета вылились в создание программной платформы BOINC [9], упрощающей создание проектов для новых задач.

Распределённые вычисления в системе BOINC основаны на проектах. Один сервер может поддерживать несколько независимых проектов, каждый из которых создаётся для решения конкретной задачи и может обладать собственной программной и аппаратной инфраструктурой. Предполагая, что имеется компьютер, удовлетворяющий необходимым минимальным системным требованиям, с установленной операционной системой, процесс создания проекта состоит в последовательном выполнении следующих шагов.

1. Установка необходимого программного обеспечения. BOINC опирается на стороннее программное обеспечение, которое должно быть установлено и сконфигурировано перед началом работы. Так, для хранения различной информации (пользователи, исполнительные компьютеры, поддерживаемые платформы, блоки входных данных, приложения и пр.) используется СУБД MySQL [10], а доступ пользователей к информации

проекта (регистрация, настройка учётной записи, мониторинг и пр.) осуществляется через веб-интерфейс, реализованный на веб-сервере Apache. Дополнительно для настройки BOINC-сервера необходимо установить Python (для запуска конфигурационных скриптов), PHP (для генерации html-страниц веб-интерфейса) и libcurl (для передачи данных).

2. Установка BOINC-сервера и создание проекта. Для установки сервера скачиваются его исходные коды, и выполняется несложный процесс сборки. Затем запускается специальный скрипт для создания проекта и указывается его имя. При этом для нового проекта создаётся структура директорий, а также база данных, в которой будет храниться вся информация, относящаяся к этому проекту.

3. Установка и настройка веб-сервера. BOINC использует веб-сервер Apache, который выполняет три основные функции. Во-первых, он предоставляет пользователям доступ к информации о проекте через веб-интерфейс. Во-вторых, в процессе функционирования через веб-сервер осуществляется взаимодействие BOINC-сервера с исполнительными компьютерами для сбора информации об их характеристиках и состоянии. И, в-третьих, веб-сервер обеспечивает возможность передачи данных для загрузки входных порций на исполнительные компьютеры, а также для отправки результатов вычислений обратно на сервер.

4. Конфигурирование проекта. Основная часть конфигурирования состоит в настройке демона, который генерирует порции входных данных и планировщика, распределяющего эти порции по исполнительным компьютерам. Кроме того, необходимо изменить информацию о проекте на веб-сайте, для чего достаточно воспользоваться прилагаемым шаблоном и модифицировать его в соответствии со своим проектом.

5. Создание и подготовка приложения. Ключевым моментом подготовки проекта является создание приложения, которое необходимо написать в соответствии с требованиями системы BOINC. Написанное по таким правилам приложение может быть скомпилировано для различных платформ (Windows, Mac, Linux и др.), и каждая версия приложения будет автоматически распределяться на исполнительный компьютер с соответствующей архитектурой.

Создавая свой собственный сервер в инфраструктуре BOINC (Рис. 1), организация публикует информацию о своём проекте, чтобы привлечь добровольцев для решения сложных задач. Эта информация содержит описание проекта, адрес в сети, откуда можно скачать клиентское приложение, а также адрес самого проекта, который необходимо указать в настройках клиентского приложения.

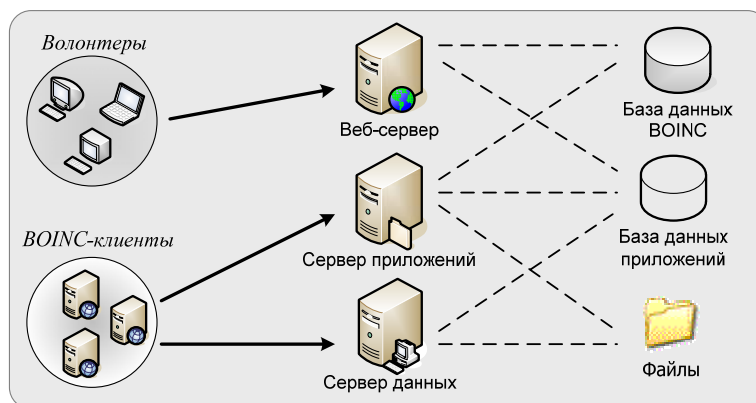


Рис.1 Инфраструктура системы BOINC

Перед началом работы владелец компьютера регистрируется на сайте проекта и получает персональный ключ, по которому он будет идентифицироваться при последующих подключениях. После подключения сервер определяет архитектуру компьютера и предоставляет соответствующую версию приложения, а также порцию входных данных. Далее осуществляется доставка приложения и входных данных на исполнительный компьютер-клиент, и приложение начинает выполняться в те моменты, когда владелец снижает свою активность. После завершения обработки полученной порции данных, клиент отправляет результат серверу и получает очередную порцию. Как только все порции данных будут обработаны, полученные от клиентов частичные результаты объединяются на сервере, и выполнение проекта прекращается.

3.2. P2P-подход. Системы Cluster Computing On the Fly и OurGrid

Cluster Computing On the Fly. Примером системы, позволяющей строить одноранговые сети, является разработка Орегонского университета Cluster Computing On the Fly (CCOF) [11]. Архитектура системы соответствует подходу P2P и построена по аналогии с файлообменными сетями, только вместо файлов в качестве ресурса выступает процессор исполнительного компьютера. В состав компонентов программной архитектуры этой системы входят клиентский планировщик, работающий на машине пользователя, и планировщик исполнительного узла. Система решает задачу управления распределённой инфраструктурой, которую образуют исполнительные узлы, путём координации планирования между множеством их планировщиков.

Как и в любой P2P-сети, здесь все узлы равноправны, нет выделенного центрального сервера и клиентов. Каждый узел является связующим звеном между своими соседями (Рис. 2) и хранит у себя информацию о ресурсах соседей. Даже когда владелец узла начинает интенсивно использовать его ресурсы, и никакие внешние задания не поступают, узел продолжает принимать и пересылать сообщения, которые проходят через него. Узел предоставляет всем остальным участникам шаблон, описывающий занятость

ресурсов в определённые промежутки времени. Владелец узла может составить такой шаблон вручную, либо его можно сгенерировать автоматически с помощью программы-монитора.

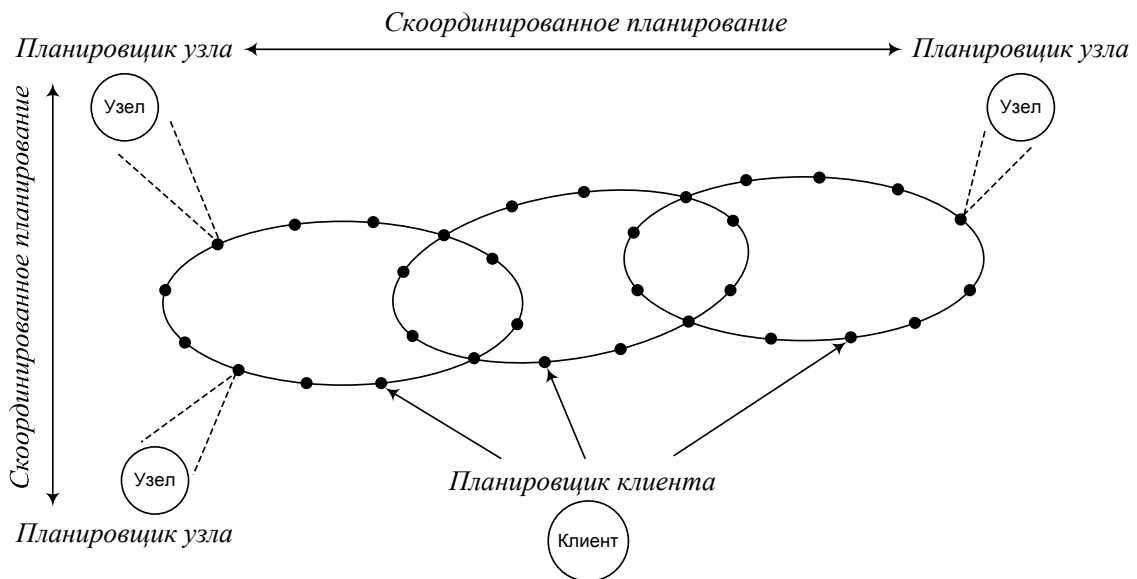


Рис.2 Архитектура системы SCOF

Поиск свободных ресурсов осуществляется на основе представленных шаблонов. Имеется несколько алгоритмов поиска ресурсов, суть которых состоит в том, что, получив от клиента ресурсный запрос, узел сравнивает его со своим шаблоном и определяет, может ли он запустить это задание. Если запрос удовлетворяет шаблону, и узел свободен, клиент передаёт ему задание.

Во время выполнения задания клиентский планировщик периодически получает сигналы от исполнительного узла о том, что узел “жив”, и выполнение задания продолжается. В том случае, если с исполнительным узлом что-то произошло, сигналы прекращаются, а клиентский планировщик перезапускает своё задание на другом узле.

По сути, распределение заданий осуществляется клиентским планировщиком, который расположен на машине пользователя. Его запросы принимаются планировщиком исполнительного узла – кандидата для приёма задания. На основе локальных политик по некоторым критериям (производительность, ранг, степень доверия и т.д.) планировщик исполнительного узла выбирает те задачи из предлагаемых клиентским планировщиком, которые он готов запустить у себя. Таким образом, система SCOF реализует децентрализованную схему планирования, которая обладает всеми преимуществами и недостатками, свойственными этому подходу.

OurGrid. В системе OurGrid [12] сделана попытка разработать универсальный подход, пригодный для построения распределённых инфраструктур, построенных как из кластеров, так и из отдельных компьютеров. Кроме того, OurGrid сочетает централизованное распределение ресурсов с децентрализованным на основе P2P-сетей.

Ресурсная инфраструктура поделена на части, каждая из которых называется сайтом и состоит из компьютеров, относящихся к одному административному домену. Эти компьютеры могут представлять собой как множество отдельных персональных машин, так и входить в состав кластера, находящегося под управлением таких менеджеров ресурсов, как PBS и LSF.

В системе реализовано три типа компонентов: диспетчер сайта, компонент исполнительной машины и пользовательский интерфейс (Рис. 3). Для управления сайтом выделяется отдельный компьютер, на котором устанавливается программный компонент OurGrid peer – диспетчер сайта. Компонент SWAN (Sandboxing Without A Name) располагается на всех исполнительных компьютерах каждого сайта, которые занимаются непосредственной обработкой заданий. Пользователи взаимодействуют со всей системой через персонального брокера MyGrid, которого они устанавливают на свои машины.

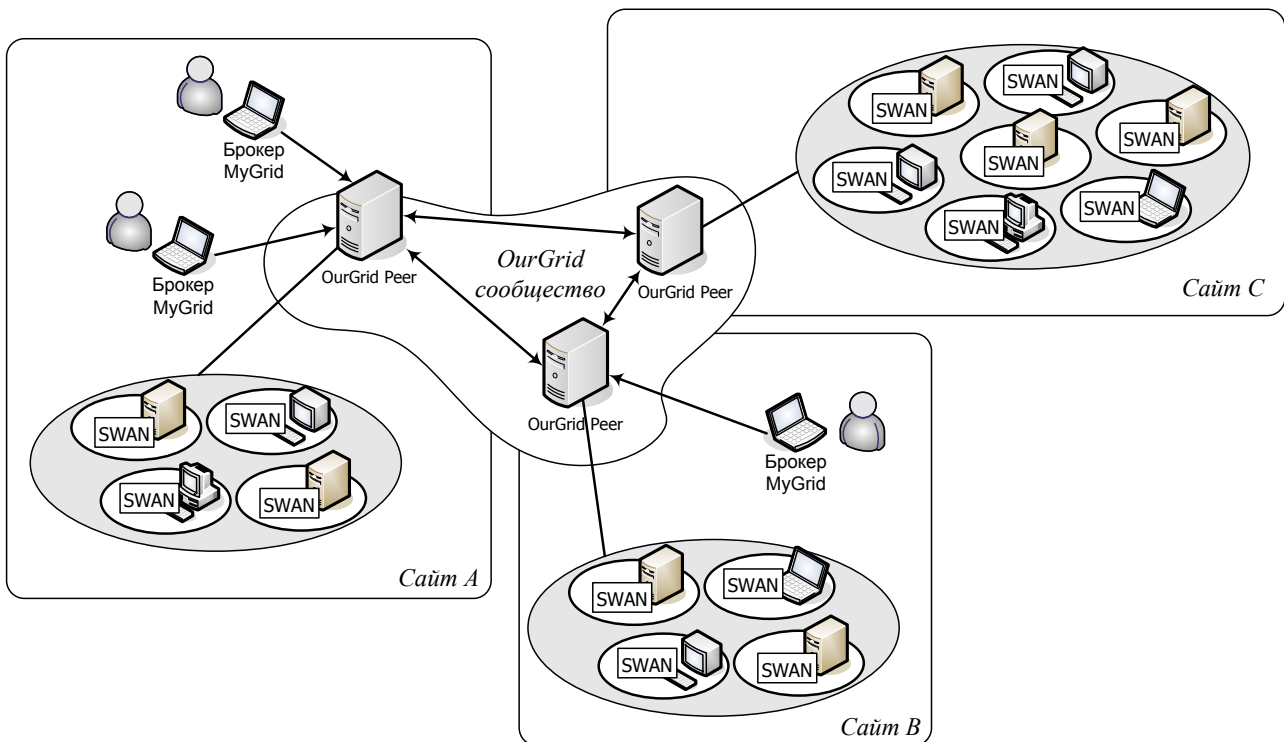


Рис.3 Архитектура системы OurGrid

Брокер MyGrid осуществляет распределение заданий по ресурсам и предоставляет набор абстракций, скрывающих от пользователя гетерогенность вычислительной среды. Пользовательское задание передаётся брокеру, который запрашивает необходимые ресурсы у всех сайтов, на которых пользователь имеет учётную запись (account). Каждый сайт находится под управлением своего диспетчера (OurGrid peer), который по запросу брокера сообщает, какое количество ресурсов доступно пользователю. Эти ресурсы могут быть локальными (входить в состав сайта, который контролируется самим диспетчером) или внешними, то есть находящимися под управлением диспетчеров других сайтов. Компьютеры, на которых установлен компонент

OurGrid peer, образуют P2P-сообщество (или одноранговую сеть). Информацию о внешних ресурсах диспетчер получает, взаимодействуя с другими диспетчерами сайтов в P2P-сети способом, аналогичным применяемому в системе CCOF.

Реализованный в брокере способ планирования ориентирован на сериализуемые задания. Получив информацию о ресурсах, брокер приступает к распределению частей сериализуемых заданий, используя в качестве целевой функции планирования критерий минимизации времени выполнения задания в целом. Для того, чтобы гарантировать завершение задания, брокер использует механизм миграции, учитывая возможность отключения исполнительных компьютеров.

Служба безопасности SWAN реализована с помощью монитора виртуальных машин Xen [13], который изолирует код пользовательского приложения внутри оболочки (sandbox). Выполняясь внутри этой оболочки, задание не только не имеет доступа к данным реального компьютера, но также не может использовать сеть. Таким образом, в системе поддерживается обработка заданий, не требующих обмена данными во время выполнения.

Хотя изначально система OurGrid разрабатывалась для построения гридов, состоящих из кластеров, с её помощью можно организовать вычисления в одноуровневом гриде. Для этого на каждый исполнительный компьютер необходимо помимо компонента SWAN установить диспетчер OurGrid peer.

3.3. Entropia

Отдельный класс систем распределённого компьютеринга составляют корпоративные решения, нацеленные на повышение эффективности работы машинного парка предприятия. Так, в системе Entropia [14] предложен новый подход к объединению персональных компьютеров для обработки заданий, в том числе сериализуемых.

Инфраструктура Entropia состоит из сервера и множества клиентов, на которых установлена агентская часть системы. Компонент сервера делит задание на множество подзадач и запускает их на клиентских машинах. После завершения вычислений результаты выполнения подзадач собираются вместе и возвращаются пользователю.

Программная архитектура системы Entropia (Рис. 4) состоит из трех уровней. Компоненты уровней управления исполнительными компьютерами и распределения ресурсов устанавливаются на сервере и клиентских машинах. Компоненты уровня управления заданиями устанавливаются только на сервере. Совместно с системой Entropia также могут функционировать и другие системы управления заданиями.

Система поддерживает выполнение любого Win32-приложения без каких-либо изменений, путём изоляции его внутри виртуальной машины.

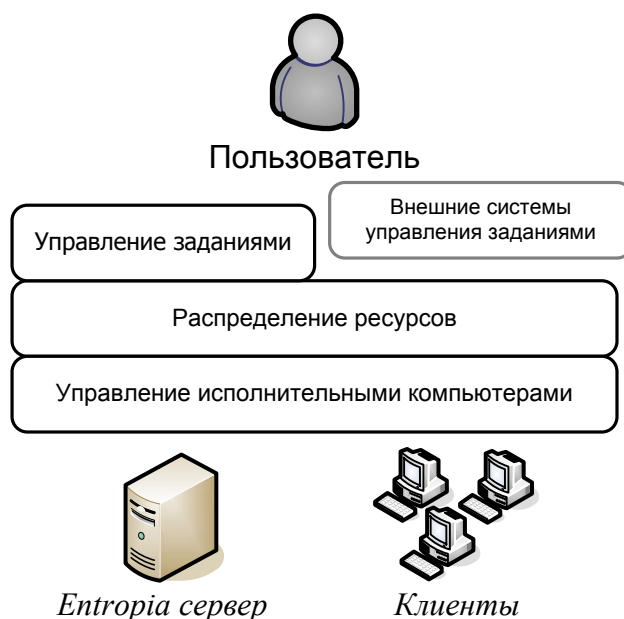


Рис.4 Архитектура системы Entropia

Программное обеспечение системы Entropia опирается на частные протоколы взаимодействия и базовую программную среду Windows. Система использует компьютеры организации для решения ресурсоёмких задач (например, анализ финансовых рисков) в те часы, когда они слабо загружены, либо простаивают.

3.4. Condor

Одной из широко используемых разработок в области распределённого компьютеринга, осуществляющих распределение заданий на некластеризованные ресурсы, является система пакетной обработки заданий Condor [15], которая была создана в университете штата Висконсин, США. Система разрабатывалась для объединения компьютеров университета и используется для решения сложных задач, требующих большого количества вычислительных ресурсов. В отличие от традиционных систем пакетной обработки, которые управляют отчуждаемыми ресурсами, Condor позволяет распределять задания как на отчуждаемые, так и не отчуждаемые компьютеры, эффективно используя их вычислительные мощности.

Объединяя множество ресурсов в один пул, система Condor (Рис. 5) предоставляет средства распределения заданий на исполнительные компьютеры, запуска заданий, а также управления заданиями и ресурсами.

Распределение ресурсов основано на использовании языка ClassAd (Classified Advertisement), с помощью которого описываются и ресурсы компьютеров, и требования заданий к этим ресурсам. Важной особенностью ClassAd является расширяемость множества используемых в нём атрибутов. На основе этого языка реализован механизм (matchmaking), осуществляющий поиск исполнительных компьютеров для каждого задания из очереди путём сопоставления информации о компьютерах и ресурсных запросов заданий.

Компоненты системы Condor на исполнительных компьютерах осуществляют разделение ресурсов между локальными заданиями и заданиями внешних пользователей, соблюдая неотчуждаемость компьютера. В случае возрастания активности владельца, Condor-задание может быть приостановлено и затем возобновлено с того места, на котором оно остановилось.

Для того, чтобы запустить задание в системе Condor, пользователю необходимо создать для него файл описания, содержащий требования к исполнительным машинам, а также подготовить все необходимые входные файлы, включая исполняемый файл. Все файлы должны находиться на машине запуска в директории задания. На основе этой информации по команде с машины запуска создается описание задания на языке ClassAd, которое используется для поиска ресурсов, а задание попадает в очередь.

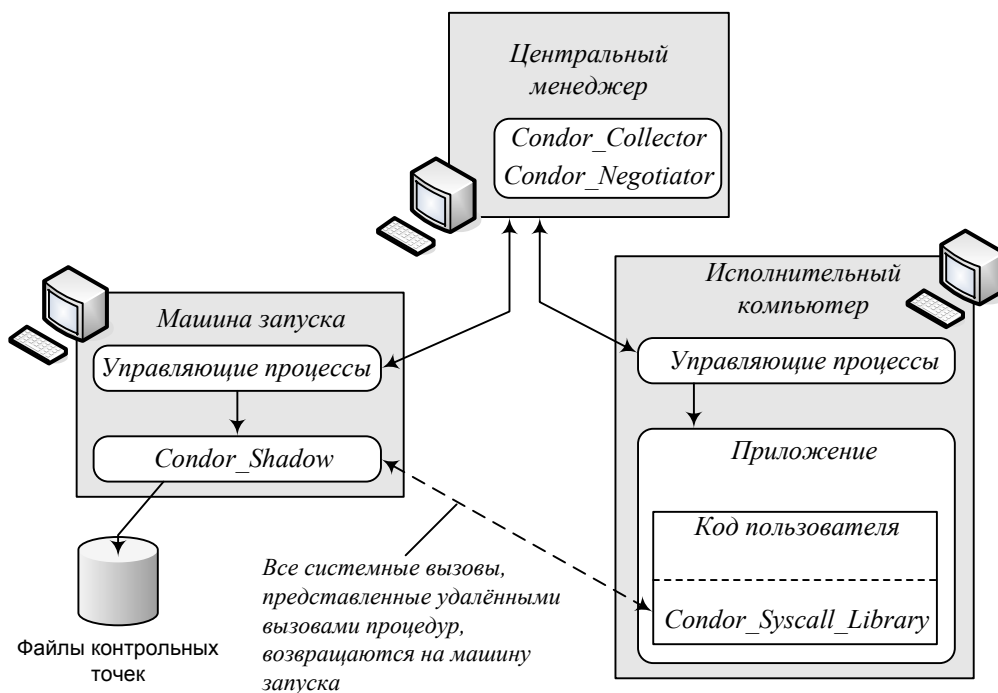


Рис.5 Архитектура системы Condor

В системе Condor поддерживается распределенная очередь заданий, обработка которой осуществляется на двух уровнях. На первом уровне каждая машина запуска поддерживает очередь заданий, которые были через нее запущены. Эти задания обрабатываются в соответствии с приоритетами пользователей, кроме того, пользователь может установить порядок выполнения своих заданий. На втором уровне центральный менеджер собирает полную информацию об очередях запускающих машин и формирует глобальную очередь. Получая информацию о состоянии ресурсов пула и их характеристиках, он осуществляет поиск исполнительных компьютеров для каждого задания из этой очереди в порядке их приоритетов.

После того, как для задания найдены ресурсы, оно запускается на исполнительном компьютере, а машина запуска осуществляет управление

этим заданием. Средства управления заданиями системы Condor позволяют пользователю в любой момент времени снять своё задание с выполнения, а также узнать его статус.

После завершения выполнения задания результат автоматически передаётся на машину запуска с помощью средств доставки файлов. Имеется два способа доставки. Первый из них предполагает использование общей файловой системы, которая должна объединять машину запуска и все исполнительные компьютеры. При отсутствии доступа к общей файловой системе, вторым способом является использование собственного механизма системы Condor для перемещения файлов.

В системе Condor поддерживаются также механизмы миграции заданий, создания контрольных точек, а также удалённого вызова процедур. В случае возрастания активности владельца исполнительного компьютера механизм миграции совместно с механизмом создания контрольных точек позволяют продолжить обработку задания на другом компьютере с того места, на котором она приостановилась.

С помощью механизма удалённого вызова процедур система Condor обеспечивает доступ к среде машины запуска, несмотря на то, что выполнение задания осуществляется на удалённом компьютере. Пользователям не приходится заботиться об обеспечении доступа к файлам с данными, поскольку задание может работать с ними так, как если бы оно выполнялось на той же машине, откуда было запущено. При этом файлы остаются на машине запуска и не передаются на исполнительный компьютер.

Эти механизмы позволяют эффективно выполнять задания пользователей на ресурсах пула, которые не отчуждаются от своих владельцев. Однако, для их использования необходимы исходные коды приложения, которые должны быть скомпилированы с библиотеками системы Condor.

Механизмы безопасности и поддержания целостности данных отвечают требованиям, предъявляемым к грид-системам, обеспечивают надёжную передачу информации между компонентами системы, а также гарантируют защиту данных пользователя, запустившего задание, и владельца ресурса. Одним из возможных способов аутентификации в системе является GSI аутентификация на основе публичных ключей с использованием сертификатов.

4. ВОЗМОЖНОСТЬ ПРИМЕНЕНИЯ СУЩЕСТВУЮЩИХ СИСТЕМ ДЛЯ ПОСТРОЕНИЯ ОДНОУРОВНЕВОГО ГРИДА

Каждая из рассмотренных систем внесла вклад в развитие методов распределённого компьютеринга, но в то же время ни одна из них не решает всех задач, существенных для создания полноценных грид-инфраструктур.

1. Платформа VOINC предназначена для поддержки проектов, а не для запуска приложений. Основное отличие состоит в том, что проект поддерживает выполнение одного или нескольких приложений,

разработанных организацией, которая создаёт VOINC-сервер. Все проекты независимы, имеют свои серверы, базы данных и выполняют разные приложения. Таким образом, для каждого нового приложения необходимо создавать свою программно-аппаратную инфраструктуру. Для такого рода систем характерно отсутствие средств запуска и управления произвольными пользовательскими приложениями, оформляемыми в виде заданий.

2. Entropia представляет собой корпоративное решение на частных протоколах, отличных от применяемых в гриде и ориентированных на локальную среду предприятия. В связи с этим, ресурсы, находящиеся под управлением системы Entropia, невозможно интегрировать в существующие грид-инфраструктуры.

3. Системы OurGrid и CCOF организованы по типу P2P-сетей и поддерживают выделение ресурсов для произвольных заданий. Обе системы реализуют механизм миграции, кроме того, OurGrid обеспечивает защиту исполнительной машины. В то же время в них не используются ставшие фактическими стандартами протоколы грида. Это приводит к тому, что инфраструктуры такого рода могут функционировать лишь изолированно и не являются интероперабельными с другими гридами.

С точки зрения архитектуры грида общим недостатком систем VOINC, Entropia, CCOF и OurGrid является отсутствие информационной службы, публикующей информацию о ресурсах. Наличие такой службы необходимо для включения сегмента из некластеризованных ресурсов в грид-инфраструктуру более высокого уровня.

4. Наиболее близка к рассматриваемой задаче система Condor, в которой поддерживается использование отдельных исполнительных компьютеров в неотчуждаемом режиме. Однако эта система получила распространение прежде всего как локальный менеджер ресурсов в одном административном домене, аналогичный, например PBS или LSF. Более того, именно в этой роли она широко применяется в гридах, построенных с помощью комплекса Globus Toolkit, в котором служба управления заданиями GRAM [16] имеет встроенный интерфейс с системой Condor в качестве одного из вариантов.

Некоторые механизмы Condor, такие как удалённый вызов процедур и общая файловая система, очевидно локальны, но для них имеются альтернативы, сближающие Condor со стандартами грида. По-видимому, это и даёт основание разработчикам позиционировать её как систему, способную работать в глобальной сети. Тем не менее, при большом числе локальных установок, проекты, в которых Condor управляет глобально распределёнными некластеризованными ресурсами, неизвестны, и такое её применение остаётся проблематичным.

Анализ архитектуры системы Condor показывает ряд её недостатков с точки зрения одноуровневого грида. В частности, в ней используются собственные интерфейсы взаимодействия компонентов, не основанные на принятых в гриде стандартах. Узким местом системы является её пользовательский интерфейс. В Condor машина запуска, через которую пользователи отправляют свои задания на обработку, также выполняет роль

монитора той части заданий, которые через неё запускаются. Для каждого выполняющегося задания на машине запуска стартует отдельный процесс, который завершается только после окончания выполнения задания. Это накладывает дополнительные требования на аппаратные ресурсы машины запуска и делает систему в целом слабо масштабируемой.

Также следует отметить, что механизм распределения ресурсов не в полной мере учитывает специфику неотчуждаемых ресурсов, не обеспечивая, например, завершение обработки заданий в приемлемое время. Для этого необходима дополнительная информация, описывающая состояние процесса вычислений на исполнительном компьютере, а также наличие поставщика, предоставляющего эту информацию. Соответствующим образом должен быть модифицирован сам алгоритм распределения заданий.

Тем не менее, Condor предоставляет широкий набор средств, с помощью которых можно решить некоторые из обозначенных проблем. Так, гибкий механизм описания ресурсов и заданий ClassAd позволяет расширить множество атрибутов, описывающих выполнение задания, и учитывать их при поиске подходящих ресурсов.

5. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ ОДНОУРОВНЕВОГО ГРИДА

Цель настоящей работы – разработка архитектуры программного обеспечения грида (ПОГ) для управления заданиями в гриде с некластеризованными ресурсами (компьютерами), в том числе такими, которые используются совместно с их владельцами, то есть не отчуждаются целиком в грид.

Среди основных условий, которым должно удовлетворять ПОГ, объединяющее некластеризованные ресурсы в полноценную грид-инфраструктуру, мы выделяем следующие.

1) **Запуск и управление заданиями на разделяемых ресурсах.** ПОГ должно реализовывать главную функцию вычислительного грида: обеспечивать выполнение заданий на множестве ресурсов. Программные средства должны принимать задания, описания которых представлены в общепринятой форме (например, RSL [17], JSDL [18] или JDL [19]). Заметим, что используемые языки описания допускают введение дополнительных параметров, связанных со спецификой одноуровневого грида. Одним из таких параметров может быть ограничение на общее время обработки задания.

Должно поддерживаться управление заданиями – удаление, получение информации о состоянии. Программные интерфейсы запуска и управления должны определяться в соответствии с архитектурой грида. Тем самым, обеспечивается доступ к гриду из любых приложений, которым необходимы дополнительные ресурсы для выполнения прикладной обработки данных. Пользовательский интерфейс управления заданиями является примером такого приложения.

Ресурсы грида используются коллективно и разделяются между различными заданиями. При выполнении задания система должна выделять ему определённое количество исполнительных ресурсов, доставлять входные файлы и результаты, а также контролировать выполнение прикладного кода.

2) **Стандартизация.** ПОГ должно в максимальной степени удовлетворять общепринятым грид-стандартам. Это обеспечит возможность функционирования соответствующих грид-инфраструктур как автономно, так и в составе объёмлющего грида с любыми формами организации ресурсов. В качестве таких стандартов мы рассматриваем архитектуру OGSA [20] и стандарты веб-служб WSDL, SOAP и др. [21].

3) **Поддержка виртуальных организаций.** ПОГ должно поддерживать деятельность большого числа динамических объединений пользователей (виртуальных организаций), обеспечивая разграничение доступа к ресурсам, принадлежащих разным виртуальным организациям.

4) **Интероперабельность.** ПОГ должно в стандартной форме предоставлять информацию о ресурсах поддерживаемой инфраструктуры, на основе чего может организовываться управление иерархически вложенными сегментами грида.

В дальнейшем изложении в качестве основного варианта одноуровневого грида мы будем рассматривать грид с неотчуждаемыми ресурсами. Это предполагает, что они, с одной стороны, используются для обработки заданий грида, а с другой – что на них выполняются приложения, которые запускают локальные пользователи компьютера (будем называть их заданиями владельца компьютера или просто локальными заданиями). Заметим, однако, что большинство предлагаемых решений подходит и для другой важной формы ресурсной организации – когда компьютеры не кластеризуются, но выделяются в грид полностью.

Специфические особенности неотчуждаемых ресурсов определяют дополнительные требования, которым должна удовлетворять разрабатываемая система.

1) **Соблюдение приоритетов.** Задания владельца ресурса должны обладать более высоким приоритетом по сравнению с глобальными заданиями, то есть выполнение глобального задания не должно мешать выполнению локального.

2) **Автономность ресурсов.** Ресурсы должны сохранять автономность, то есть владелец должен иметь возможность устанавливать политику их использования. Объём ресурсов, получаемых заданиями грида, должен быть лимитирован. Особое внимание следует уделить безопасности как исходных кодов и файлов с данными грид-задания, так и безопасности исполнительного компьютера в целом.

3) **Динамичность среды.** Должна учитываться динамичность грида, то есть непрогнозируемые выключения и включения отдельных ресурсов, а также обеспечиваться возможность автоматического, не требующего административного вмешательства, включения компьютеров в состав ресурсной базы грида.

Эти требования накладывают ограничения на способ реализации ПОГ. Составляющая ПОГ, которая будет устанавливаться на исполнительный компьютер, должна быть максимально компактной, расходовать минимум системных ресурсов, а её установка и администрирование не должны требовать от владельца дополнительных знаний и навыков. Требование минимизации программного обеспечения делает невозможным применение распространённых средств грида, поскольку все они подразумевают установку серверных компонентов с соответствующими накладными расходами на их сопровождение.

Далее мы предлагаем решения для:

- архитектуры ПОГ;
- механизмов взаимодействия компонентов ПОГ;
- обеспечения интероперабельности разрабатываемого ПОГ с подобным обеспечением грида для иных форм ресурсной организации.

6. АРХИТЕКТУРА СИСТЕМЫ ДИСПЕТЧЕРИЗАЦИИ

Для объединения некластеризованных ресурсов и включения их в состав грид-инфраструктуры мы предлагаем архитектуру ПОГ в виде системы диспетчеризации, которая состоит из трёх компонентов: диспетчера, агента и пользовательского интерфейса (Рис. 6).

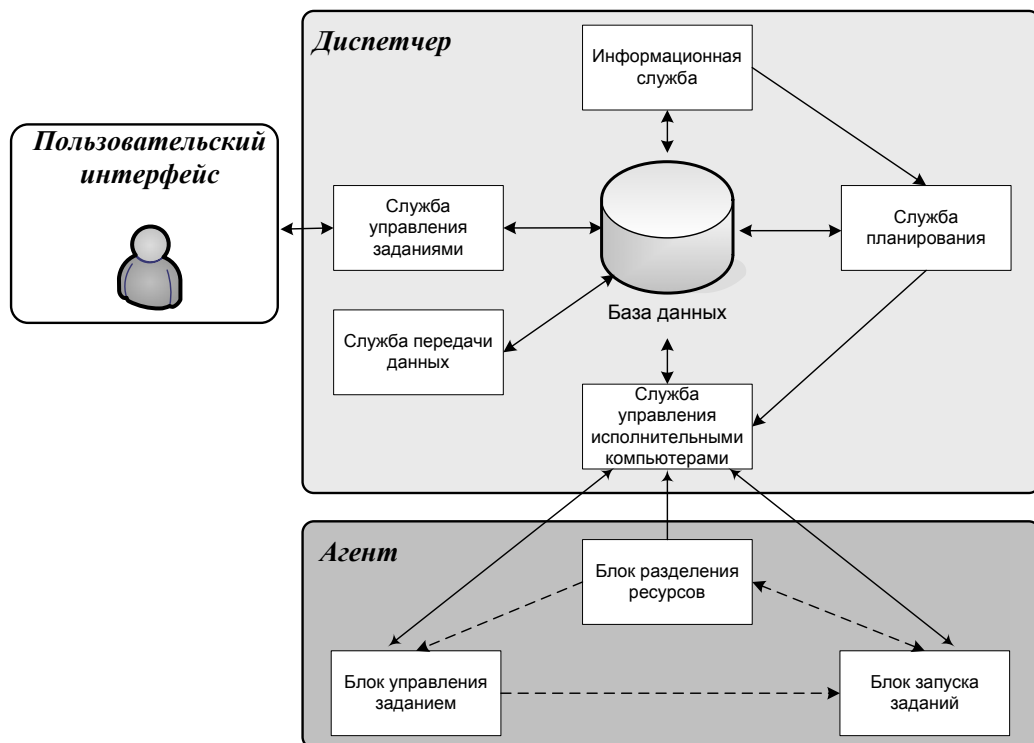


Рис.6 Архитектура системы диспетчеризации

В соответствии с предложениями работы [1] исполнительные компьютеры ресурсной инфраструктуры не доступны непосредственно, а все

внешние интерфейсы доступа к ним сосредоточены в диспетчере. Диспетчер устанавливается на выделенный сервер в сегменте грида, и к нему подключается множество пространственно распределённых компьютеров. Для обеспечения интероперабельности с существующими грид-системами, диспетчер имеет стандартный интерфейс запуска заданий (аналогичный GRAM), а реализация информационной службы позволяет публиковать данные о доступных ресурсах сегмента в информационной системе объёмлющей инфраструктуры.

Основная функция агента, устанавливаемого на исполнительный компьютер, – это управление заданием на стадии выполнения. Для того, чтобы сделать агента компактным, почти все функции как управления гридом, так и взаимодействия компонентов реализуются в диспетчере.

Пользовательский интерфейс предоставляет возможность пользователям управлять заданиями стандартным для грида способом, а также получать информацию о состоянии задания.

Далее рассмотрим все три программные компоненты и их функции.

6.1. Диспетчер одноуровневого грида

Диспетчер одноуровневого грида реализует интерфейсы доступа к сегменту грида с некластеризованными ресурсами, выполняя распределение заданий между зарегистрированными в нём компьютерами.

Диспетчер представляется набором грид-служб, в котором реализованы необходимые базовые компоненты ПОГ, а также средства взаимодействия с агентами. Как известно, грид-службы опираются на стандарты веб-служб, которые, во-первых, широко применяются при построении распределённых систем, а во-вторых, используются при разработке грид-приложений. Кроме того, реализация подсистем в виде служб позволяет получить модульную систему. В состав диспетчера входят следующие службы:

- служба управления заданиями;
- служба планирования (планировщик);
- информационная служба;
- служба взаимодействия с агентами;
- служба передачи данных.

Центральную роль в системе диспетчеризации играет **информационная база**, которая хранит информацию о заданиях, а также о ресурсах – исполнительных компьютерах.

Информацию о задании можно разделить на регистрационную и информацию о текущем состоянии. Регистрационная информация поставляется службой управления заданиями и содержит следующее: приоритет задания, описание задания (включая ресурсный запрос и информацию о необходимых входных данных), а также информацию о пользователе, который запустил задание. Задание проходит несколько стадий

обработки, и его состояние меняется. После того, как заданию выделены ресурсы, в базу заносится информация о том, на каком компьютере оно будет запущено. В процессе выполнения задания информация о нём поставляется службой взаимодействия с агентами и содержит: состояние задания (новое, выполняется, завершено и т.д.), а также количество полученных заданием ресурсов (процессорное время, дисковое пространство).

Служба взаимодействия с агентами заносит в базу информацию о ресурсах всех компьютеров, подключенных к гриду, а также их статус (занят или свободен). В момент регистрации нового компьютера предоставляется конфигурационный файл с описанием ресурсов (производительность процессора, дисковое пространство, память). В нём указывается максимальный объём ресурсов, который владелец компьютера отдаёт под нужды грида. Кроме того, он может указать предпочтительный интервал времени использования своего компьютера (например, это могут быть ночные часы, когда компьютер простаивает).

Служба управления заданиями принимает задания (осуществляет разбор файла описания задания и добавляет информацию о нём в информационную базу), а также выполняет другие команды управления по запросам пользователей. В результате приёма задания пользователю возвращается идентификатор, используя который он может управлять своим заданием, посылая соответствующие команды (снять) службе, а также получать информацию о состоянии задания и количестве потреблённых заданием ресурсов.

Служба планирования (планировщик) отвечает за распределение заданий по ресурсам. В системе используется приоритетное планирование с очередью заданий, цель которого состоит в том, чтобы запустить максимальное количество заданий на имеющихся ресурсах и обеспечить окончание заданий в отведённое время.

Все необходимые для планирования данные хранятся в информационной базе. Первичная информация (статическая и динамическая информация о ресурсах, а также информация о заданиях) поступает в базу от служб управления ресурсами и заданиями, после чего обрабатывается информационной службой.

Для каждого задания из очереди планировщик сначала ищет подходящий исполнительный компьютер и инициирует запуск задания. После запуска он осуществляет мониторинг выполнения для обеспечения своевременного завершения задания. Под подходящим компьютером мы понимаем такой, который удовлетворяет ресурсным требованиям задания (архитектура, операционная система, дисковое пространство, оперативная память и пр.), ожидаемое (по статистической информации) получение ресурсов обеспечивает выполнение в срок, а пользователь обладает правами запуска на этом компьютере.

Специфика рассматриваемых ресурсов накладывает ограничения на способ распределения заданий. Сейчас в гриде применяются два способа: диспетчер сам запускает задание (push) или задание запрашивается у него

исполнительным компьютером (pull). Первый способ широко применяется для грида из кластеризованных ресурсов. Такой подход предполагает наличие на исполнительном компьютере службы приёма заданий, поэтому в рассматриваемом варианте некластеризованных ресурсов предпочтительным представляется второй способ (pull): агент, работающий на исполнительном компьютере, сообщает диспетчеру о своей готовности принять задание.

Важным механизмом в условиях плохо предсказуемых ресурсов является миграция заданий. Она применяется в ситуации, когда задание не успевает завершиться в срок на изначально выделенном ему исполнительном компьютере. Такая ситуация отслеживается планировщиком, который осуществляет перезапуск задания на другом компьютере.

Учитывая динамичность среды (частое изменение состояния ресурсов), а также то, что прогноз освобождения ресурсов не является точным, можно повысить вероятность успешного выполнения задания и скорость его исполнения за счёт запуска нескольких копий на различных исполнительных компьютерах. Например, можно фиксировать количество копий для каждого задания и ждать завершения первых двух из них, либо запускать задание на все подходящие ресурсы, а с приходом новых заданий, претендующих на те же ресурсы, снимать со счёта часть копий. При этом увеличивается вероятность успешного завершения задачи, а наличие как минимум двух результатов позволяет контролировать их достоверность.

Для реализации планирования в условиях неотчуждаемости ресурсов необходимо прогнозировать их доступность на каждом из компьютеров грида. Такой прогноз основывается на данных о том, сколько процессорного времени компьютеры отдают заданиям грида. Эта информация собирается на исполнительных компьютерах и периодически передаётся в информационную базу. Так как величина выделяемого времени носит случайный характер, необходимо накапливать статистику по различным периодам, учитывая, например, что в ночное время и выходные дни вычислительные ресурсы обычно простаивают или слабо загружены.

При планировании выполняется сопоставление данных о предоставляемом времени с характеристиками задания: временем выполнения (количество нормированного процессорного времени, необходимого заданию, walltime) и сроком завершения (время, до которого задание должно быть выполнено, deadline).

Информационная служба (ИС) осуществляет двухэтапную обработку непосредственно поставляемой в базу первичной информации о ресурсах.

Неотчуждаемый компьютер может быть занят владельцем и недоступен для выполнения заданий грида. Предсказать, когда это произойдет, невозможно, поэтому, распределяя задания, нельзя ориентироваться на показатели доступных для грида ресурсов в какой-то определённый момент времени. Поэтому на первом этапе первичная информация о каждом компьютере, поставляемая агентами, перерабатывается информационной службой в статистическую, которая отражает количество предоставляемых им ресурсов (процессорного времени) заданиям грида в различные интервалы

некоторого периода времени, например суток или недели. Для каждого такого интервала собирается статистика, на основе которой определяется оценка вероятности того, что грид может получить определённое количество ресурсов на данном компьютере. Статистика является основой для планирования распределения заданий.

Второй этап обработки направлен на обобщенное представление суммарного количества ресурсов, которыми располагает сегмент грида. Компьютеры, имеющие схожие характеристики (архитектуру, операционную систему, быстродействие процессора, объём памяти), объединяются в группы. На основе статистической информации в каждой группе вычисляется агрегированное количество ресурсов, доступных для грида в те или иные интервалы времени.

Агрегированная по группам информация служит средством представления сегмента грида в объемлющих грид-инфраструктурах и может публиковаться во внешних информационных системах. Это позволяет остальным участникам грида (пользователям и брокерам ресурсов) видеть информацию о ресурсах сегмента и автоматически обрабатывать её. Если внутри системы диспетчеризации информация может храниться и представляться в произвольном виде, то для публикации информации в объемлющей инфраструктуре необходимо использовать стандартные и общепринятые протоколы. Обновление агрегированной информации происходит в зависимости от интенсивности изменения ситуации, а также по запросу внешних систем на получение информации о ресурсах сегмента.

Служба взаимодействия с агентами поддерживает связь между диспетчером и исполнительными компьютерами. Эта служба выполняет следующие функции:

- предоставляет интерфейс автоматической регистрации новых исполнительных компьютеров: при регистрации агент обращается к этой службе и передаёт файл с описанием ресурсов компьютера;
- разбирает файл с описанием ресурсов компьютера и заносит полученную информацию в базу данных;
- принимает от агента сведения о состоянии задания и количестве потреблённых им ресурсов.

Взаимодействие агента с диспетчером происходит по внутренним протоколам, которые могут отличаться от протоколов грида. Безопасность обеспечивается взаимным предъявлением сертификатов и наличием доверительных соглашений.

Служба передачи и хранения данных осуществляет доставку файлов на шлюзовую машину, а также на исполнительные компьютеры. В соответствии со стандартным протоколом управления заданиями GRAM система диспетчеризации поддерживает доставку стандартных файлов: исполняемого, входных данных и диагностики. Мы предполагаем, что прикладные данные располагаются на серверах хранения и считываются по протоколам грида управления данными (GridFTP [22]).

Рассматриваемая служба осуществляет временное хранение стандартных файлов до завершения выполнения задания. Если задание не может быть завершено на отведенном компьютере (задание получает недостаточно процессорного времени или компьютер выключили), то необходим весь этот набор файлов для перезапуска задания на другом компьютере. Таким образом, службой реализуются следующие функции:

- передача стандартных файлов задания на исполнительный компьютер;
- доставка результата по адресу, который указан в описании задания;
- временное хранение всех необходимых заданию данных до его завершения.

6.2. Агент

Агент устанавливается на исполнительных компьютерах, владельцы которых хотят предоставить свои ресурсы в грид.

Ресурсы исполнительного компьютера делятся между процессами владельца и глобальными заданиями. При регистрации исполнительного компьютера его владелец заполняет конфигурационный файл, в котором указывает, какое количество ресурсов каждого типа (процессорное время, дисковое пространство, оперативная память и т.д.) он отчуждает в грид-инфраструктуру. При этом задание не может потребить больше ресурсов, чем было заявлено в его ресурсном запросе, несмотря на то, что свободные ресурсы на исполнительном компьютере фактически ещё могут оставаться. Как только задание выходит за границы потребления ресурсов, указанных в запросе (например, занимает больше дискового пространства или выполняется дольше, чем было заявлено), его выполнение прекращается.

Агент выполняет запуск задания и управление им на исполнительном компьютере, осуществляя разделение ресурсов между процессами владельца и заданием грида, а также обеспечивая передачу данных и безопасность.

Безопасность рассматривается в трех аспектах:

- защита владельца, то есть обеспечение полнофункциональности компьютера путём контроля уровня потребления ресурсов внешним приложением;
- защита конфигурации компьютера, находящихся на нём программ и данных;
- защита грид-приложения, включая программу, данные и результаты.

Агент состоит из трех блоков.

Блок запуска заданий информирует диспетчер о том, что ресурсы исполнительного компьютера освободились, и запрашивает новое задание. По этому запросу планировщик выбирает из очереди подходящее задание и сообщает его идентификатор агенту. Затем блок запуска заданий осуществляет доставку файлов задания на исполнительный компьютер, формирует

исполнительную среду и запускает задание. После того, как задание запущено, управление передаётся блоку разделения ресурсов.

В том случае, если подходящего задания для освободившегося ресурса нет, агент периодически обращается за ним через определённые интервалы времени.

Блок разделения ресурсов обеспечивает изоляцию программы и данных пользователя грида от данных владельца компьютера, а также осуществляет контроль за потреблёнными заданием ресурсами. Периодически данные об использовании ресурсов заносятся в информационную базу. Кроме того, блок разделения ресурсов осуществляет мониторинг активности владельца исполнительного компьютера. В ситуации, когда выполняющееся задание превысило выделенный ему лимит ресурсов, блок разделения ресурсов инициирует принудительное завершение задания. Если задание не завершено, а владелец исполнительного компьютера активизировал свою деятельность, блок управления заданием приостанавливает выполнение задания и информирует об этом диспетчер. После того, как активность владельца снизилась, выполнение задания возобновляется.

Блок управления заданием посредством взаимодействия с операционной системой управляет заданием во время его выполнения. По сигналам от блока управления ресурсами блок управления заданием прекращает выполнение задания, а также осуществляет приостановку и возобновление выполнения задания.

Таким образом, на исполнительном компьютере задание обрабатывается по следующей схеме:

1. Ресурсы на исполнительном компьютере освобождаются, и агент запрашивает у диспетчера новое задание.

- Блок запуска заданий информирует службу взаимодействия с агентами о том, что исполнительный компьютер готов принять новое задание. Эта служба обращается к службе планирования (планировщику), которая выдаёт идентификатор подходящего задания с наивысшим приоритетом, а также меняет статус исполнительного компьютера в базе данных.
- Затем блок запуска заданий создаёт на исполнительном компьютере структуру директорий и по полученному на шаге 1 идентификатору доставляет файлы задания через службу передачи и хранения данных.
- Блок запуска заданий формирует исполнительную среду, запускает задание и передаёт идентификатор системного процесса, соответствующего глобальному заданию, блоку разделения ресурсов. Кроме того, блок запуска заданий через службу взаимодействия с агентами меняет статус задания (выполняется).

2. Агент обеспечивает выполнение задания на исполнительном компьютере с соблюдением неотчуждаемости ресурсов.

- Блок разделения ресурсов контролирует количество потреблённых заданием ресурсов. Если задание начинает потреблять больше ресурсов, чем было заявлено в запросе или увеличивается загрузка компьютера за счёт локальных процессов владельца, то посылается запрос блоку управления заданием о снятии задания или приостановлении вычислений. Также блок управления заданием прекращает выполнение задания по запросу пользователя. При этом в информационной базе меняется статус задания.
- Во время выполнения задания блок разделения ресурсов отправляет диспетчеру информацию о количестве потреблённых ресурсов, а блок управления заданием посылает информацию о статусе задания, если статус меняется. Эта информация так же заносится в базу данных.

3. После окончания вычислений агент завершает процесс выполнения задания.

- После завершения выполнения задания блок запуска отправляет выходные файлы через службу передачи и хранения данных. Затем все данные, связанные с заданием, удаляются с исполнительного компьютера.
- Служба передачи и хранения данных информирует службу управления заданиями о завершении задания и сообщает его идентификатор.
- По идентификатору задания служба управления заданиями доставляет выходные данные пользователю и меняет статус задания в информационной базе (завершено).
- Блок запуска заданий запрашивает статус исполнительного компьютера у блока разделения ресурсов. Если ресурсы свободны, блок запуска заданий информирует диспетчер о том, что исполнительный компьютер готов принять новое задание, и цикл повторяется.

6.3. Пользовательский интерфейс

Пользовательский интерфейс представляет собой набор клиентских приложений, созданных на основе API служб диспетчера. Пользователю предоставляется средства описания задания (например, с помощью языка JDL), а также средства мониторинга и управления заданием. В любой момент выполнения задания пользователь может узнать, в каком состоянии оно находится, количество потреблённых заданием ресурсов, а также прекратить выполнение задания.

7. ЗАКЛЮЧЕНИЕ

Основная идея грида – интеграция пространственно распределённых ресурсов может быть полезной в различных условиях и разных сценариях применения. В рамках концепции грида возможна организация ресурсной инфраструктуры, которая не обязательно должна создаваться на базе кластерных и многопроцессорных систем. Использование распределённых вычислений на практике показало наличие прикладных областей, где возникают задачи, для решения которых не требуется создание кластерных конфигураций. Такие задачи могут распараллеливаться и решаться на множестве обычных персональных компьютеров.

Работа посвящена проблеме создания программного обеспечения для распределённых вычислительных инфраструктур, образованных из некластеризованных ресурсов, на основе принципов грида. Как показывает приведённый обзор, существующие подходы к решению этой проблемы имеют недостатки, в том числе и с точки зрения потребительских качеств.

Сформулированы два главных требования к программному обеспечению для управления некластеризованными компьютерами. Во-первых, оно должно быть способным работать с неотчуждаемыми компьютерами в режиме разделения ресурсов с их владельцами. Во-вторых, оно должно быть построено в соответствии с основными положениями грида, обладать свойством интероперабельности с распространёнными средствами управления заданиями, иметь стандартизованные внешние интерфейсы и использовать протоколы, принятые в гриде.

Исходя из этих требований, предложена архитектура системы диспетчеризации, определяющая состав компонентов и их функции. Специфицированы способы и последовательность обработки заданий, включая приём, планирование, выделение ресурсов и доставку файлов. Рассмотрены механизмы управления заданием на стадии его выполнения (разделение ресурсов с процессами владельца компьютера, поставка информации о состоянии).

Предложенная архитектура системы диспетчеризации позволяет использовать некластеризованные ресурсы как автономно в виде изолированного сегмента грида, так и в составе объёмных грид-инфраструктур с иной формой организации ресурсов.

8. ЛИТЕРАТУРА

- [1]. В.Н.Коваленко, Д.А.Корягин: Организация ресурсов в грид. Препринт № 63. Москва: ИПМ им. М.В.Келдыша РАН, 2004. 25 с.
- [2]. Система пакетной обработки заданий PBS – <http://www.openpbs.org/>
- [3]. Система пакетной обработки заданий Platform LSF – <http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/>
- [4]. Инструментарий Globus Toolkit – <http://www.globus.org>

- [5]. Проект gLite – <http://glite.web.cern.ch/>
- [6]. D. Anderson, J. Cobb, E. Korpela. SETI@home: An Experiment in Public-Resource Computing. Communications of the ACM, Vol. 45 No. 11, November 2002, pp. 56-61.
- [7]. Проект SZTAKI Desktop Grid – <http://szdg.lpds.sztaki.hu/szdg/>
- [8]. Peer-to-Peer – <http://www.openp2p.com/>
- [9]. D. P. Anderson: BOINC: A System for Public-Resource Computing and Storage. 5th IEEE/ACM International Workshop on Grid Computing, November 2004, pp 1-7.
- [10]. СУБД MySQL - <http://www.mysql.com/>
- [11]. D. Zhou, V. Lo: Cluster Computing on the Fly: resource discovery in a cycle sharing peer-to-peer system. Fourth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'04), 2004, pp. 66-73.
- [12]. W. Cirne, F. Brasileiro, N. Andrade, R. Santos, A. Andrade: Labs of the World, Unite!!! Journal of Grid Computing, Vol.4, No. 3, September 2006, pp. 225-246.
- [13]. Монитор виртуальных машин Xen – <http://www.xensource.com/>
- [14]. Система Entropia – <http://www.entropia.com>
- [15]. Система Condor – <http://www.cs.wisc.edu/condor>
- [16]. Служба управления заданиями GRAM – http://www.globus.org/grid_software/computation/gram.php
- [17]. Resource Specification Language – http://www.globus.org/toolkit/docs/4.0/execution/wsgram/schemas/gram_job_description.html
- [18]. A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, A. S. McGough, D. Pulsipher, and A. Savva. Job Submission Description Language (JSDL) specification, version 1.0 <http://www.ogf.org/documents/GFD.56.pdf>, March 2007.
- [19]. Job Description Language – <http://www.grid.org.tr/servisler/dokumanlar/DataGrid-JDL-HowTo.pdf>
- [20]. I. Foster, C. Kesselman, J. Nick, S. Tuecke: The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, <http://www.globus.org/reseach/papers/ogsa.pdf>
- [21]. S. Graham, S. Simeonov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, R. Neyama. Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, 2001.
- [22]. GridFTP – http://www.globus.org/grid_software/data/gridftp.php