

О Р Д Е Н А Л Е Н И Н А
И Н С Т И Т У Т П Р И К Л А Д Н О Й М А Т Е М А Т И К И
И М Е Н И М . В . К Е Л Д Ы Ш А
Р О С С И Й С К О Й А К А Д Е М И И Н А У К

В . Н . К О В А Л Е Н К О , Д . А . К О Р Я Г И Н

О Р Г А Н И З А Ц И Я Р Е С У Р С О В Г Р И Д

Москва
2004 г.

В.Н. Коваленко, Д.А. Корягин. Организация ресурсов грид.

Развитие технологий грид подошло к фазе реализации проектов, направленных на развертывание крупных действующих инфраструктур. Выход в практику принес не только новые задачи, но и нуждается в оценке подходов, которые были разработаны и успешно применялись в пробных экспериментах. В этой связи ценным представляется опыт компаний, выпускающих коммерческие программные продукты для распределенного компьютеринга. Анализу решений, которые дополняют и даже в какой-то степени альтернативны признанным стандартам, посвящена данная работа.

Ключевые слова: грид, распределенный компьютеринг, ресурсная архитектура, кластер, безопасность, диспетчеризация

V.N. Kovalenko, D.A.Koryagin. Organization of Grid resources.

Grid development has approached the phase of realization of projects aimed at deployment of the large-scale operating infrastructures. Realities of practical implementation has not only brought in a whole new set of tasks, but also demanded reevaluation of the methods that were previously developed and successfully tested in pilot experiments. In this respect, experience gained by commercial companies developing software tools for the distributed computing seems to be valuable. Analysis of various solutions that complement and to a certain extent offer an alternative to the acknowledged standards is the subject of this paper.

Key words: GRID, distributed computing, resource architecture, cluster, security, scheduling

Содержание

1. ВВЕДЕНИЕ: ОСНОВНЫЕ ПОЛОЖЕНИЯ ГРИД.....	4
1.1. Концепция грид	4
1.2. Программное обеспечение грид	4
1.3. Ресурсы грид.....	6
2. СПОСОБЫ ОРГАНИЗАЦИИ РЕСУРСОВ ГРИД	8
3. ДВУХУРОВНЕВЫЙ ГРИД	10
3.1. Комплексные узлы: проблемы и решения.....	10
3.2. Политика распределения ресурсов.....	14
4. ОДНОУРОВНЕВЫЙ ГРИД.....	16
4.1. Архитектурные решения одноуровневого грид	17
4.2. Области применения одноуровневого грид	18
4.3. Технологии одноуровневого грид	20
4.3.1. Платформа DCGrid и вопросы безопасности.....	20
4.3.2. Платформа LiveCluster	22
5. ЗАКЛЮЧЕНИЕ	23
6. ЛИТЕРАТУРА.....	24

1. ВВЕДЕНИЕ: ОСНОВНЫЕ ПОЛОЖЕНИЯ ГРИД

1.1. Концепция грид

Концепция грид породила новую модель организации различных форм обработки данных (компьютинга), предложив технологии удаленного доступа к ресурсам разных типов независимо от места их расположения в глобальной сетевой среде. Так, с помощью грид появляется возможность выполнять программные коды на одном или сразу нескольких "чужих" компьютерах, становятся повсеместно доступными хранилища данных со структурированной (базы данных) и неструктурированной (файлы) информацией, источники данных (датчики, инструменты наблюдения) и программно управляемые устройства.

Цель создания грид - интеграция определенного множества пространственно распределенных ресурсов для того, чтобы обеспечить возможность выполнения широкого класса приложений на любой совокупности этих ресурсов, независимо от места их расположения.

В реализации грид представляет собой инфраструктуру, которая состоит из находящихся в разных местах ресурсов, соединяющих их телекоммуникаций (сетевые ресурсы) и взаимосогласованного по всей инфраструктуре связующего (middleware) программного обеспечения (ПО), поддерживающего выполнение дистанционных операций, а также выполняющего функции контроля и управления операционной средой. Грид создается владельцами ресурсов, выделяющими их в общее пользование. Владельцы и потребители, действующие на основании определенных правил предоставления/потребления ресурсов, образуют **виртуальную организацию (ВО)**.

Грид является средой коллективного компьютеринга, в которой каждый ресурс имеет владельца, а доступ к ресурсам открыт в разделяемом по времени и по пространству режиме множеству входящих в ВО пользователей. Виртуальная организация может образовываться динамически и иметь ограниченное время существования.

Таким образом, следуя [1], можно определить грид как пространственно распределенную операционную среду с гибким, безопасным и скоординированным разделением ресурсов для выполнения приложений в динамически образующихся виртуальных организациях.

1.2. Программное обеспечение грид

Концепция грид родилась в научном сообществе, и вполне естественно, что разработки и исследования начальной стадии были направлены на поддержку вычислительно интенсивных задач научно-технического характера. В результате был предложен ряд протоколов: коммуникационный, безопасности и удаленного доступа к вычислительным, файловым, информационным ресурсам. Набор протоколов хотя был и ограниченным, но достаточным для запуска заданий, управления ими, генерации исполнительной

среды, доставки входных и результирующих файлов. Протоколы были поддержаны реализацией двух версий инструментальной системы Globus Toolkit (GT1 и GT2) [2]. Globus Toolkit (GT) и ряд разработанных на его основе программных средств образовали программную составляющую инфраструктуры для нескольких крупных Грид, в том числе DataGrid [3] и GriPhyn [4]. На эти Грид поставлены приложения по обработке результатов экспериментов в области ядерной физики. В связи с большими объемами данных и вычислений интеграция распределенных ресурсов оказалась критически важной и доказала свою полезность.

Однако проблемы организации распределенного компьютеринга не являются специфичными для научно-технической сферы, и разработка новой архитектуры ПО грид велась с учетом интересов производственных приложений, при непосредственном участии крупнейших компаний - производителей компьютерной техники и ПО, таких как IBM, Sun, Hewlett-Packard и Microsoft.

Основные положения предложенного в [5] стандарта архитектуры ПО грид - OGSA (Open Grid Service Architecture) следуют объектно-ориентированной модели и рассматривают в качестве основного объекта грид **службу**. Посредством удаленного обращения к методам службы приложение получает определенный вид обслуживания. Таким образом унифицируются различные функции: доступа к вычислительным ресурсам, ресурсам хранения, базам данных и к любой программной обработке данных.

Архитектура грид-служб решает важнейшую проблему распределенной среды – проблему интероперабельности - путем стандартизации способа описания интерфейсов служб. В этом отношении OGSA опирается на стандарты Web-служб, в частности на SOAP, WSDL и WSI [6]. Дополнительно OGSA частично стандартизирует семантику грид-служб, определяя унифицированный для всех служб набор методов и соответствующих интерфейсов: обнаружения, динамического создания, управления временем жизни, уведомлений и т.д.

В 2003 году вышла первая реализация инструментария, основанная на архитектуре OGSA - Globus Toolkit 3.0 (GT3). В GT3 выделяются две части. Первая – это базовая программная инфраструктура Open Grid Services Infrastructure (OGSI), которая содержит поддержку общих для всех служб методов. Вторая часть включает службы, имевшиеся в GT2 (управления заданиями, передачи файлов, информационного обслуживания и безопасности), но уже в архитектуре OGSA.

Переход на архитектуру OGSA важен, во-первых, поскольку становится возможным поддержка как научных (E-science), так и производственных приложений (E-business), функционирующих в пространственно распределенной среде грид. Распределенные приложения, реализованные в виде грид-служб, обладают четко определенными интерфейсами и могут взаимодействовать между собой по стандартным протоколам. Для

прикладного уровня реализация взаимодействия в большой степени автоматизирована и скрыта от разработчика.

Во-вторых, инструментарий для грид - Globus Toolkit стал расширяемым. Предыдущие его версии содержали ограниченный набор служб (GRAM, GRIS, GISS, GridFTP). Реализации этих служб были практически независимы друг от друга и имели немного общих методов (за исключением методов протокола безопасности Grid Security Infrastructure), поэтому их разработка, модернизация во многом дублировалась и усложнялась. Архитектура OGSA и инфраструктура OGSi образовали общий контекст, в котором создание новых служб существенно упростилось.

1.3. Ресурсы грид

Уже начиная с версии GT2 инструментарий Globus Toolkit стал фактическим стандартом для грид, признанным как научным сообществом, так и ведущими компаниями компьютерной индустрии [7]. Благодаря тому, что GT с самого начала имел и по-прежнему сохраняет статус открытого ПО, к настоящему времени накоплен значительный опыт его применения в крупных проектах. Используя инструментальные средства GT, разными коллективами были разработаны дополнительные службы: репликации файлов, авторизации, диспетчеризации заданий и др.

Существует, однако, круг вопросов, который во многом остается вне связанной с GT линии развития и который можно обозначить как **организация ресурсной составляющей инфраструктуры грид**. ИПМ РАН, совместно с несколькими российскими институтами принимает участие в проекте EGEE (Enabling Grids for E-science in Europe) [8], целью которого является создание общеевропейского грид. Работы по проекту только начинаются, но уже понятно, что наиболее дорогостоящим и трудоемким будет создание и обслуживание именно ресурсной составляющей инфраструктуры.

Стандарт OGSA определяет службы как абстрактные объекты, но не содержит никаких предписаний о способе их реализации. В OGSA не затрагиваются вопросы программной модели служб и исполнительской среды их функционирования, что, конечно, имеет смысл, так как делает стандарт независимым от реализационной платформы.

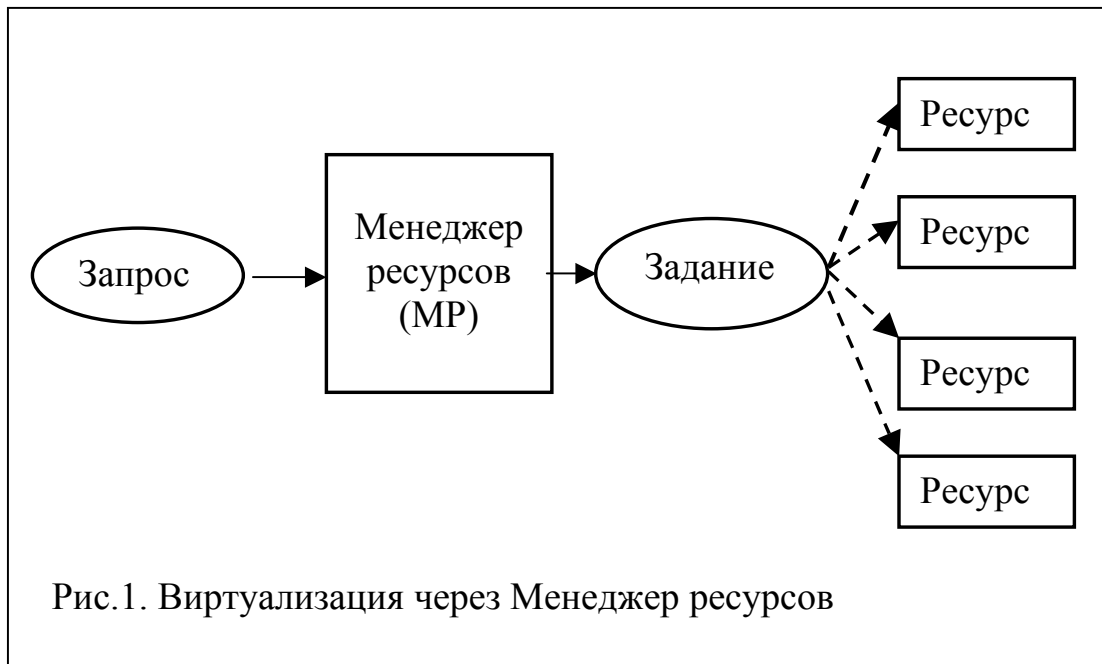
На практике, например в GT3, грид-службы реализуются в компонентных средах - контейнерах, разработанных для Web-служб. Так, на платформе J2EE применяются разные типы контейнеров: EJB, JSP, сервлеты и апплеты. Роль контейнеров – размещение служб, обеспечение жизненного цикла, поддержка безопасности.

Если этих функций контейнера достаточно для Web-служб, то для грид-служб требуется большее - способ реализации этих служб должен обеспечивать виртуализацию ресурсов:

- многопользовательское обслуживание, динамически адаптирующееся к меняющейся нагрузке путем порождения множества экземпляров служб.

- автоматическое распределение ресурсов между экземплярами служб, выполняющих обработку потока запросов.

Этим требованиям отвечает среда исполнения грид-служб, в которой имеется пул ресурсов с единым управлением, осуществляемым менеджером ресурсов (МР). Запрос, поступающий на интерфейс службы преобразуется в форму задания для МР и передается ему по его интерфейсам. Основные функции МР – выделение ресурсов под задания и поддержка их выполнения (рис. 1).



К сожалению, в современных исследованиях вопросам организации ресурсов и управления службами уделяется относительно мало внимания. Тем больший интерес вызывают программные продукты коммерческих компаний, занимающихся разработкой средств для поддержки внутренней информационной инфраструктуры предприятий, инфраструктуры, связывающей предприятия, и инфраструктуры провайдеров услуг. Если исследовательское направление сосредоточено, главным образом, на моделях и протоколах обеспечения интероперабельности пространственно распределенного ПО, то основные достижения коммерческих систем лежат как раз в сфере управления ресурсами.

Ведущие компании IBM, Sun, Hewlett-Packard, Avaki, Oracle и др. заинтересованы в развитии пространственно распределенного компьютеринга и связывают перспективы с переходом на протоколы и архитектуру грид. Общие положения способа построения грид на базе локальных систем управления распределенными ресурсами выглядят следующим образом [9].

- Модель служб OGSA рассматривается как будущий стандарт всей информационной индустрии, на основе которого будут строиться

пространственно распределенные приложения. Через посредство служб приложения получают унифицированный дистанционный доступ к ресурсам ВО.

- Связующее ПО грид “склеивает”, то есть делает доступными потребителям, географически разнесенные, принадлежащие разным административным доменам ресурсные пулы.

- Средства грид для сбора и хранения информации снабжают ВО метаданными о ресурсах, услугах и условиях их предоставления. OGSA специфицирует формат описаний и способ хранения метаданных в реестрах. На основе метаданных работают различные коммунальные службы грид.

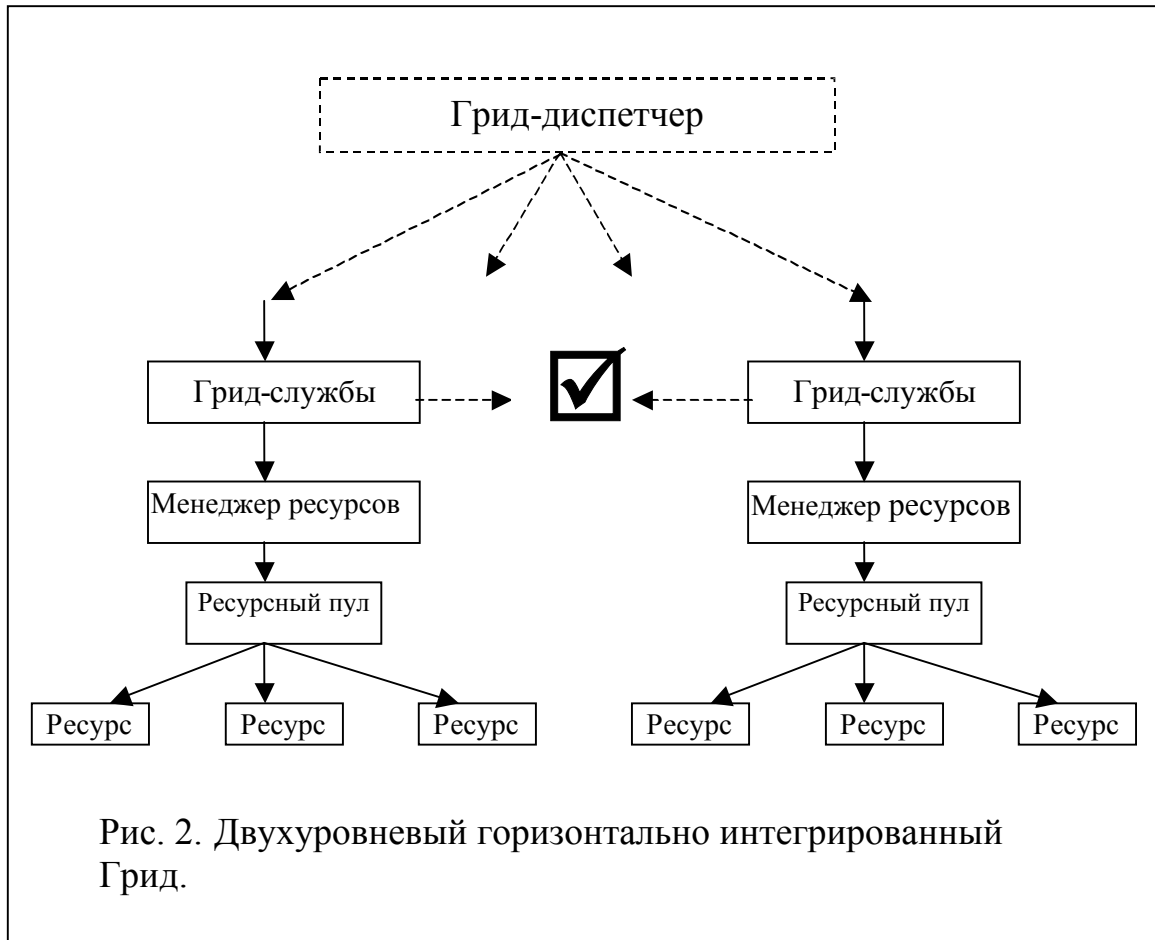
- Защита в ВО базируется на стандарте инфраструктуры безопасности (PKI) грид, основанном на сертификатах X.509. PKI поддерживает однократную регистрацию пользователей, которая действует повсеместно, на всех ресурсных пулах.

2. СПОСОБЫ ОРГАНИЗАЦИИ РЕСУРСОВ ГРИД

Рассмотрим два способа организации ресурсов грид. Первый способ, соответствующий направлению GT, можно назвать двухуровневым (или горизонтально интегрированным). В этой форме грид образуется из совокупности комплексных, то есть содержащих множество компьютеров, узлов. Ресурсы отдельного узла находятся в автономном административном домене, связаны локальной сетью и обычно управляются системой пакетной обработки (СПО), которая играет роль локального МР.

Узел включается в грид через одну одну или несколько машин-шлюзов, на которые устанавливаются грид-службы, и, таким образом ресурсы узла становятся доступны повсеместно. На такой способ организации ресурсов ориентирован GT, в котором поддержаны интерфейсы с СПО PBS [10], Condor [11], LSF [12], SGE [13] и др. Обратим внимание, что виртуализация ресурсов здесь происходит на уровне отдельных узлов грид, а для обеспечения прозрачного доступа ко всем ресурсам ВО предполагается наличие службы глобальной виртуализации – брокера или диспетчера (рис. 2).

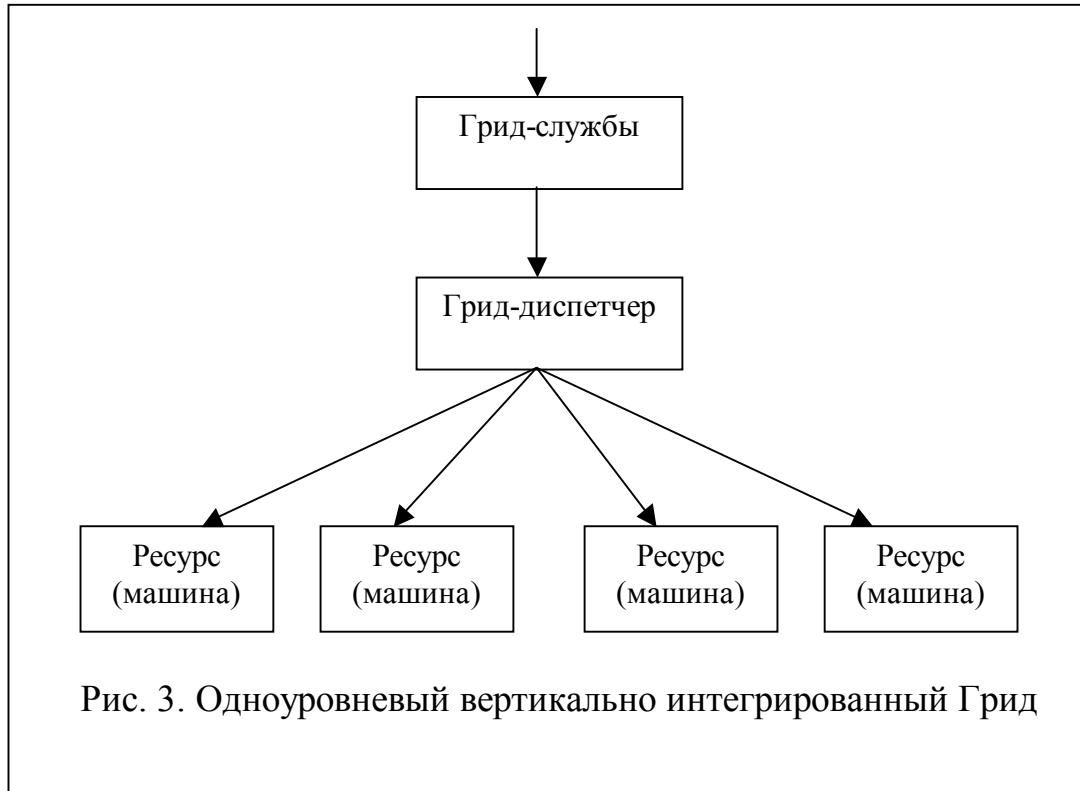
Описанный подход хорош для ВО, которые опираются на развитую локальную инфраструктуру ресурсов. Однако, представляют интерес и ситуации, когда владельцы ресурсов не используют СПО, не могут позволить себе усложнять обслуживание имеющегося парка машин или создают грид на короткое время для совместного выполнения конкретного проекта. В таких обстоятельствах более подходящим может быть второй способ организации грид – одноуровневый (или с вертикальной интеграцией).



В одноуровневой архитектуре ресурсы – пространственно распределенные компьютеры – интегрируются через управляющий центр, который, с одной стороны, представляет собой точку доступа ко всем ресурсам, а, с другой стороны, выполняет функции МР, управляя ресурсами и виртуализируя их (рис. 3).

Одноуровневый подход был предложен в проектах SETI@home [14] и Distributed.net [15], но использовался для решения достаточно специальной по сравнению с целями грид задачи – организации счета отдельных приложений на глобально распределенных ресурсах. Дальнейшее развитие одноуровневого подхода, выразившееся в появлении средств запуска и управления заданиями в продуктах нескольких компаний: Entropia, DataSynapse, United Devices, Paragon Computing, показало возможность его применения в грид, то есть в среде для выполнения множества разных приложений. Следует отметить, что до недавнего времени существовало препятствие для создания грид с помощью коммерческих систем: поскольку все они опирались на частные протоколы дистанционного взаимодействия, область действия построенных на их базе инфраструктур была ограничена корпоративным уровнем. Но положение меняется - перечисленные выше компании в той или иной степени принимали участие в разработке протоколов для архитектуры OGSA и имеют конкретные планы перехода от частных решений к стандартам, утверждаемым грид-сообществом.

Проводя далее сравнение двухуровневого и одноуровневого подхода мы исходим из того, что в перспективе они могут стать взаимодополняющими способами построения грид, а некоторые технологии, разработанные под определенный контекст, представляют ценность и в более широком плане.



3. ДВУХУРОВНЕВЫЙ ГРИД

3.1. Комплексные узлы: проблемы и решения

Наиболее актуальным способом создания мощных многопроцессорных комплексов в настоящее время является кластеризация массово выпускаемых компьютеров, в том числе ПК, с помощью опять же серийного телекоммуникационного оборудования локальной сети (LAN), на основе которого обеспечивается взаимодействие машин.

Если сравнивать кластер и SMP-систему с равным числом процессоров и одинаковой производительностью, то стоимость кластера оказывается на порядок меньше. То есть, с технической точки зрения кластерные системы дают эффективное по соотношению цена/производительность, хорошо масштабируемое решение, пригодное для счета слабо связанных программ, а при использовании высокопроизводительных сетевых технологий типа Myrinet, и программ с интенсивным сетевым взаимодействием. Появились компании, которые поставляют кластерные системы “под ключ” - кластеры становятся таким же рядовым товаром как и отдельные компьютеры.

Обратная сторона перечисленных полезных свойств состоит в том, что, в отличие от систем SMP, кластер – архитектура со слабой интеграцией как

аппаратных, так и программных компонентов. В частности, из-за отсутствия общей памяти на каждый процессорный узел должна быть установлена своя копия ОС. Поэтому, основная проблема кластерных комплексов – проблема обслуживания, трудоемкость которого возрастает с числом процессоров, а с учетом того, что средний размер продаваемых кластеров достиг за последнее время 64-128 процессоров, проблема становится критической.

Известны и широко применяются на практике архитектурные принципы, технические устройства и программные средства, которые позволяют справиться с проблемой “большой системы”, нивелируя разницу между кластером и SMP-сервером [16].

Архитектурно кластер строится в виде множества обрабатывающих узлов и одного головного управляющего узла. Последний выделен тем, что только через него возможен любой внешний доступ к обрабатывающим узлам. Управляющий узел имеет двойную сетевую привязку, будучи связанным через один интерфейс с внутренними узлами, а через другой с внешней открытой сетью. Доступ пользователей из открытой сети к обрабатывающим узлам осуществляется через управляющий узел, на котором располагается программный интерфейс менеджера ресурсов для запуска и управления заданиями.

Что касается обрабатывающих узлов, то хорошим будет решение, когда они не имеют внешних IP-адресов и все традиционные протоколы удаленного доступа на них отключены. Каким образом их тогда можно администрировать? Администрирование, тем не менее, может производиться дистанционно и для всего пула обрабатывающих компьютеров достаточно одного комплекта интерфейсных устройств: клавиатуры, монитора и мыши (KVM – Keyboard, Video, Mouse) при условии, что его можно подключать к разным машинам. Для этого каждый компьютер соединяется с KVM через коммутатор-переключатель с ручным или программным управлением, причем это соединение может быть реализовано как в локальной сети, так и через интернет. Если кластер дополнительно оборудуется аналогичным устройством для дистанционного управления питанием, обеспечивающим включение/выключение и перезагрузку узлов, то удаленное администрирование кластера становится практически полным и может производиться из любой точки сети по надежным и защищенным протоколам.

Основу программной поддержки функционирования кластера как интегрированного центра обслуживания пользователей составляют системы пакетной обработки. Играя роль менеджеров распределенных ресурсов, они предоставляют пользовательский интерфейс для запуска и управления заданиями, поддерживают очереди заданий, осуществляют распределение ресурсов и запуск заданий на них, доставку результатов на рабочие места пользователей.

Но задача обслуживания кластера имеет многосторонний характер и требует, помимо СПО, наличия целого ряда программных инструментов. Одним из существенных аргументов в пользу выбора Linux в качестве базовой

ОС кластера служит то, что для каждой отдельной функции обслуживания можно найти открытое ПО.

Инсталляция и конфигурирование. (Системы SystemImager [17], KickStart [18], CFEngine [19], WebMin [20]). Эти системы выполняют инсталляцию операционных систем и приложений, распределение ПО и обновлений по всем узлам кластера. Все операции осуществляются по сети с минимальным участием администратора на основании структурированных описаний классов узлов. Если, например, кластер однородный, то достаточно определить компоненты ПО, устанавливаемые на управляющий узел и на класс обрабатываемых узлов.

Мониторинг и автоматическая обработка событий. Система Big Brother [21] производит мониторинг оборудования, ОС, памяти и сетевых устройств, отслеживая события, то есть выход параметров за критический порог. Для событий могут быть определены реакции: графическая визуализация, уведомление по электронной почте или телефону (SMS), а также корректирующие воздействия.

Мониторинг производительности. (Ganglia [22], MRTG [23], Mon [24]). В реальном времени производится визуализация данных, построение графиков о состоянии оборудования кластера: процессоров, памяти, дисков, каналов ввода/вывода, сети, а также о состоянии процесса обработки заданий: числе заданий в очередях, загрузке узлов.

Параллельное выполнение команд. (C3 [25], WebMin [20]). Выполнение команд администратора параллельно на всех машинах кластера или на выбранной группе узлов.

Управление терминалами и питанием. Дистанционное включение/выключение/перезагрузка узлов, дистанционное подключение к терминалам узлов. В отличие от перечисленного выше, ПО для этих функций лицензионное и продается поставщиками оборудования.

В принципе, для обслуживания кластера требуются все эти инструменты, но, хотя они и свободно распространяются, нужны существенные усилия для загрузки кодов из разных источников и освоения систем с различным уровнем надежности, поддержки и документированности. Следующий шаг развития средств обслуживания сделан в проектах OSCAR (Open Cluster Group) [26] и ROCKS (NPACI) [27]. В этих проектах предложен пакетный подход: несколько инструментов компилируются для конкретных компьютерных платформ и поставляются вместе.

Состав пакетов формируется исходя из лучших примеров практики. В NPACI ROCKS, например, включены Sun Grid Engine (SGE), Ganglia, MPI [28], прикладная библиотека ATLAS BLAS [29], тесты производительности High-Performance Linpack [30], а также ПО грид – Globus и Condor. ROCKS, сделанный для Red Hat Linux, позволяет загрузить по сети дистрибутивы и автоматически установить их на узлы кластера. Насколько эффективна эта схема было продемонстрировано на конференции Суперкомпьютинг 2003 [31]:

вся установка ПО на машину Sun Fire V60x со 128 узлами заняла два часа, после чего на ней были запущено реальное приложение.

Включение кластера в грид должно быть еще более простым, чем его создание. Здесь также может быть применена идея автоматической установки – совместно с установкой кластера или поверх нее. Средства для этого есть, например, в проекте LCG [32] они основаны на конфигураторе LCFGng [33]. Предлагаемое для узла типовое решение включает следующее ПО: операционную систему (RedHat 7.3), коммунальные службы GRB, GHS, DB, Replica Catalog, службы Globus – GRAM, GHS, СПО OpenPBS, а также ряд собственных служб проекта LCG. Комплект ПО скачивается по сети из CVS на сервер LCFGng, администратор кластера определяет адреса и роли машин, и далее сервер LCFGng конфигурирует по локальной сети каждую машину кластера. Степень автоматизации очень высокая: в результате, получается полностью готовый к работе в грид узел. Кроме того, в процессе функционирования кластера обеспечивается обновление ПО в соответствии с новыми версиями CVS.

При всей полезности пакетированного ПО, этот подход не способен исправить крупный недостаток, присущий собранию систем от разных разработчиков: отсутствие общей архитектуры. Независимые системы опираются на собственную базу с пересекающейся функциональностью, а такая избыточность порождает накладные расходы на их освоение и эксплуатацию.

Компания Platform Computing ставит задачу создания общей программной среды, в которую как модули могут вставляться различные инструменты администрирования и управления. Основная идея модульной архитектуры продуктов Platform Clusterware [34] и Platform LSF [35] состоит в том, чтобы рассматривать кластер в виде расширяемой совокупности Web-служб, обслуживающих как пользовательские запросы по управлению заданиями, так и административные запросы, реализуемые различными подключаемыми инструментами. И пользовательский, и административный доступ к кластеру осуществляется через графический Web-интерфейс, построенный по технологии JavaBeans [36]. Далее входная информация в стандартной форме, посредством XML и SOAP, передается Брокеру служб, который вызывает выполнение того или иного инструмента по его “родным” протоколам.

Следующее архитектурное решение касается устройства обрабатывающих узлов. Здесь применяется модель агентов – упрощенных служб, работающих на узлах и прямо взаимодействующих с нижележащей ОС. Агенты исполняют элементарные функции, так что их могут использовать различные инструменты, такие, например, как службы мониторинга или управления нагрузкой.

Ключевыми являются два агента.

- Информационный агент, который в Platform LSF называется Load Information Manager (LIM), собирает статическую информацию о

конфигурации, а также динамическую информацию о загрузке ресурсов узла. Этот агент расширяем, что позволяет менять состав собираемой информации.

Информация от LIM отдельных узлов передается в реальном времени на головной LIM, агрегируется и становится доступной для всех служб кластера. На основе такой схемы распределенного сбора информации функционирует служба событий, которая мониторирует пороги и показатели загрузки и автоматически применяет конфигурируемые правила, инициируя корректирующие действия или уведомления.

- Агент удаленного выполнения (Remote Execution Service - RES) реализует механизмы управления прикладными программами на узлах: создание исполнительных сред, установку лимитов потребления ресурсов, собственно запуск программ. В LSF этот агент поддерживает последовательные и параллельные программы, интерактивные программы, командные скрипты, выполняемые параллельно на нескольких узлах.

Продукты, подобные Platform LSF, показывают направление движения к созданию модульной и расширяемой базы кластерного ПО необходимой для развития средств обслуживания кластеров, однако появление такой базы тормозится практически полным, за исключением, пожалуй, DRMAA [37], отсутствием стандартов в этой области.

3.2. Политика распределения ресурсов

Являясь средой коллективного компьютеринга, грид должен содержать механизмы разделения ресурсов между независимо работающими пользователями ВО в условиях, когда к отдельным ресурсам имеют доступ большое количество пользователей и их состав может динамично меняться. В связи с этим, в грид возникают два типа коммунальных отношений: 1) между поставщиками ресурсов и пользователями; 2) между самими пользователями.

Отношения первого типа задают правила предоставления/потребления ресурсов. Они воплощаются, с одной стороны, в политике предоставления ресурсов, которую определяют владельцы узлов: кто из пользователей имеет доступ к узлу грид; с какими правами доступа к ресурсам узла должно быть запущено задание; с каким уровнем качества должно обслуживаться задание. С другой стороны, пользователь при запуске задания в **ресурсном запросе** задает требования к объему и производительности ресурсов, а также условия, на которых он хочет их получить. Развитый язык для спецификации политики предоставления ресурсов и условий их потребления Class-Adds [38] применяется в системе Condor и основанной на ней Condor-G. В результате сопоставления политики узла и условий потребления определяется возможность запуска задания на том или ином ресурсном узле грид.

Спецификация политики предоставления ресурсов и ее реализация в конфигурационных файлах является одной из задач администрирования грид, которая решается распределенным образом – каждый узел определяет ее независимо. Следует отметить, что политика в конечном итоге должна быть определена по отношению к каждому пользователю – члену ВО, однако в грид

практически реальный способ - определение политики для групп пользователей с разумным уровнем вложенности группирования.

Отношения второго типа (между пользователями) порождаются вследствие того, что при распределении ресурсов возникают конфликтные ситуации, когда несколько заданий одновременно претендуют на одни и те же ресурсы. В [39], [40] разработаны модели, в которых эти отношения рассматриваются как экономические: пользователи платят за ресурсы деньги, и получает ресурсы тот, кто больше заплатит.

Актуален и такой тип отношений, в которых денежные расчеты не применяются, а ресурсы распределяются на основе соглашений между потребителями. Подобный подход характерен и для корпоративных систем управления ресурсами, которые обслуживают несколько подразделений, занимающихся независимыми проектами. Один из вариантов “справедливого” разделения ресурсов предлагается в SGEEE (Sun Grid Engine Enterprise Edition) [41]. SGEEE представляет собой развитие системы пакетной обработки Sun Grid Engine (SGE) для условий, когда крупная корпоративная среда с распределенными ресурсами используется несколькими независимо работающими группами пользователей. Можно выделить несколько достоинств подхода SGEEE, делающих его применимым для ВО грид:

- возможность иерархического определения политики разделения ресурсов;
- автоматическое проведение политики при динамическом планировании распределения ресурсов;
- учет реального потребления ресурсов.

Определение политики. Определение политики распределения ресурсов заключается в выделении процента (квоты) от общего количества ресурсов каждому пользователю, и в SGEEE выделение квот осуществляется иерархически. На верхнем уровне дерева квот ресурсы делятся между, например, подразделениями предприятия. Пусть их всего два: S1 и S2. Единицей квоты служит доля (share). Администратор предприятия решает выделить 800 долей подразделению S1 и 200 долей - S2. Всего выпущено 1000 долей, соответствующих 100% ресурсов. Ресурсы разделены в пропорции 80% - подразделению S1 и 20% - подразделению S2.

На следующем уровне дерева квот (в примере – уровень пользователей) администратор каждого подразделения делит ресурсы между своими пользователями в рамках квоты подразделения. Если в S1 два пользователя, и они должны получать ресурсы в равных долях, каждый из них получит по 400 долей или по 40% от общего количества ресурсов.

Применение политики. Применение политики обеспечивает, чтобы распределение ресурсов в среднем на достаточном большом промежутке времени соответствовало установленным квотам, учитывая при этом, что активность пользователей меняется, и на каких-то интервалах не обязательно все пользователи запускают задания.

На каждом шаге распределения ресурсов пользователям, которые активны (претендуют на ресурсы) в данный момент, раздаются билеты (ticket).

1. При любом числе активных пользователей количество билетов равно общему количеству долей на верхнем уровне дерева квот.
2. Если активен только один пользователь, то ему выдаются все билеты.
3. Если активны несколько пользователей, то каждому выдается количество билетов, пропорциональное его квоте. Например, если квота пользователя U_1 – 10%, а квота U_2 – 20%, при общем количестве долей 1000, U_1 получает 33,3% (333 билета), а U_2 – 66,7% (667 билетов).

Таким образом, в зависимости от числа активных пользователей, отдельный пользователь может получать ресурсов как больше, так и меньше своей квоты, но SGEEE имеет механизмы компенсации пере- и недопотребления.

Усреднение происходит в результате учета реального количества потребленных ресурсов (КПР) в каждом узле дерева квот, и в случае перепотребления уменьшается количество выдаваемых билетов. КПР вычисляется как интегральная характеристика в скользящем временном окне следующим образом. Вводится интервал полузабывания H_f , так что если в некоторый момент было использовано Q ресурсов, то через этот интервал КПР становится равным $Q/2$.

Второй механизм усреднения – компенсирующий коэффициент для пользователей, которые были временно неактивны (ситуация недопотребления). В течение определенного интервала времени квота таких пользователей умножается на компенсирующий коэффициент. Если пользователь имеет квоту 10% и компенсирующий коэффициент установлен 5, то он на время получает квоту 50%.

Планировщик SGEEE распределяет ресурсы, исходя из соотношения билетов, находящихся в обращении на данный момент. Предположим, что подразделение S_1 располагает 900 билетами и запустило 100 заданий. На каждое задание приходится по 9 билетов. Если подразделение S_2 имеет 100 билетов и только 4 задания, каждое задание будет иметь 25 билетов. Задания подразделения S_2 могут выполняться прежде заданий подразделения S_1 , хотя квота S_1 составляет 90%. Однако, как только задания S_2 используют ресурсов больше полагающейся квоты, механизм учета потребления ресурсов начнет снижать количество выданных S_2 билетов.

4. ОДНОУРОВНЕВЫЙ ГРИД

Начало применению одноуровневой схемы организации пространственно распределенных ресурсов было положено в широко известных проектах SETI@home и Distributed.net. В проекте SETI@home (поиск внеземного разума) исследовательская группа Калифорнийского университета (Беркли) создала Web-сайт, с которого добровольцы могут загрузить и установить на

свой ПК агент, выполняющийся как фоновый процесс (“хранитель экрана”) и анализирующий радиотелескопические данные.

Столь необычный способ был мотивирован наблюдением, что типичный пользователь домашнего ПК задействует не более, чем 10% его мощности, так что агент может занимать холостые циклы, не мешая деятельности хозяина ПК. В течение года после начала проекта в него включилось более 2 миллионов добровольцев из разных стран мира. На волне популярности по той же методике были запущены глобально распределенные приложения по проблематике простых чисел, шифрованию и геному человека.

В развитие разработанных технологий несколько компаний: Entropia, DataSynapse, Parabon Computation и United Devices, выпустили ряд коммерческих программных продуктов, соответственно, DCGrid [42], LiveCluster [43], Frontier [44], GridMP [45], претендующих на роль платформ грид. Подчеркнем отличие начальных проектов по поддержке пространственно распределенных приложений от этих продуктов. Программные средства таких проектов как SETI@home позволяют построить распределенную среду, но только для одного приложения, а для каждого нового нужно устанавливать независимо функционирующие серверы. Пользователей в такой среде фактически нет: подключение к проекту сводится к предоставлению ресурсов. Понятно, что такой ограниченный подход едва ли можно отнести к грид.

Коммерческие же платформы способны поддерживать широкий класс приложений в рамках общей инфраструктуры. Основываясь на их направленности – поддержка функционирования пространственно распределенной операционной среды – и на наличии развитых функциональных возможностей мы относим перечисленные платформы к программным средствам грид (хотя и признавая, что их полноценность в этом качестве требует стандартизации протоколов).

4.1. Архитектурные решения одноуровневого грид

Коммерческие платформы, различаясь в конкретных технологиях, сохранили основные черты подхода, основанного на одноуровневой виртуализации ресурсов. Для всех систем характерно наличие одного центра, который управляет множеством пространственно распределенных обрабатывающих узлов (рис. 3). В отличие от двухуровневого грид, здесь узлы элементарны, то есть каждый узел – это компьютер, на который для подключения в грид устанавливается компактный Агент, выполняющий функции запуска заданий на компьютере, мониторинга заданий и ресурсов, защиты и контактов с управляющим центром (грид-сервером).

Доступ к ресурсам одноуровневого грид возможен только через интерфейсы грид-сервера, который для каждого поступающего на него запроса находит ресурсы на обрабатывающих узлах и инициирует его обработку.

Можно заметить, что одноуровневый грид – это аналог кластерной архитектуры со следующим важным отличием: обрабатывающие узлы могут

быть пространственно распределены и принадлежать разным владельцам. Сопоставление одноуровневого и двухуровневого грид показывает, что они развивают известные технологии локально распределенного компьютеринга в двух взаимодополняющих направлениях:

- Двухуровневый грид пошел по пути развития и стандартизации (OGSA) средств доступа к множеству пространственно распределенных ресурсных пулов;
- Одноуровневый грид привнес технологии построения пространственно распределенных “кластеров”.

Обеспечение унифицированного доступа к ресурсам – важная, но не единственная задача, из тех, которые должны решить технологии грид. Трудно представить себе вариант грид, построенный только на прямом доступе к ресурсам с явным указанием их адресов. Поэтому база двухуровневой архитектуры - GT рассматривается как инструментальный пакет, а для создания грид он должен быть дополнен рядом служб, в частности, брокером или диспетчером, которые обеспечивают виртуализацию ресурсов: возможность их косвенного получения через ресурсный запрос.

В одноуровневом грид виртуализация ресурсов, осуществляемая компонентой грид-сервера – планировщиком, положена в основу архитектуры. В этом отношении одноуровневый грид изначально содержит полное, практически пригодное для коллективной работы решение.

Следующий аспект различий связан с вопросами создания и обслуживания ресурсной составляющей инфраструктуры. Средства доступа к ресурсам GT предполагают, что узел грид – многомашинный комплекс с локальным управлением в виде Менеджера Ресурсов. Когда уже существует развитая локальная инфраструктура, такой подход полностью оправдан и позволяет сохранить сложившиеся принципы организации административных доменов: локальные средства и политики безопасности, управления и пр. Если же опыта работы с кластерными комплексами нет, а ВО образуется на короткое время из машин индивидуальных владельцев, то двухуровневая архитектура будет неоправданно громоздкой.

Одноуровневая инфраструктура проще. Прежде всего, в ней только один центр управления, и, в принципе, услуги по его обслуживанию могут оказываться профессиональными провайдерами. Агенты для обрабатываемых машин – легкие (объем – 1МБ, установка – минута), справиться с ними вполне по силам рядовому пользователю.

4.2. Области применения одноуровневого грид

Появление компаний, специализирующихся на коммерческом ПО пространственно распределенного компьютеринга, показывает, что технологии, бывшие предметом преимущественно научного интереса, активно внедряются в социальную сферу и производство. Этот поворот стимулируется заинтересованностью, проявляемой как государственными организациями

управления, обороны, сферы коммунальных услуг, так и частными компаниями, например, финансовыми и энергетическими. Обнаруживается, что множество совершенно разных прикладных областей: науки о жизни, защита окружающей среды, предсказание погоды и климата, численное моделирование в машино- и авиастроении, криптоанализ, биологическое моделирование, фармацевтика, используя грид, выходят на качественно новый уровень решения важных и трудных задач.

Успешность внедрения грид в различных областях во многом обусловлена тем классом задач, на которые изначально ориентировался одноуровневый подход. Это задачи, обладающие свойством “естественного параллелизма”: их полное решение может быть разбито на множество практически независимых подзаданий, имеющих одну программу обработки и различающихся исходными данными. Таким свойством обладают, например, параметрические исследования, комбинаторные проблемы, моделирование методом Монте-Карло, алгоритмы типа "разделяй и властвуй".

Коммерческие платформы грид имеют простые инструментальные средства, с помощью которых могут быть подготовлены “серийные” задания, осуществляющие полный объем вычислений на различных выборках входных данных. Обработка заданий осуществляется в режиме параллельного запуска совокупности подзаданий, выполняющихся независимо на узлах грид, а результаты всех подзаданий объединяются вместе, формируя согласованный выход. Простота постановки задач, хотя и ограниченного класса, на одноуровневый грид является важным фактором, способствующим успеху технологии.

Демонстрационным примером применения рассматриваемого подхода может служить проект Anthrax (Сибирская язва) [46], который возник в связи с имевшимися в США случаями распространения по почте спор сибирской язвы и был направлен на поиск противоядия. Проект был запущен после того, как была выделена ключевая белковая компонента спор. Вычислительная часть задачи заключалась в том, чтобы произвести на выделенном белке скрининг 3.57 млрд. потенциальных ингибиторов токсина. Задача решалась на специальном варианте платформы Grid MP компании United Devices. В распределенные вычисления было вовлечено около 1.9 млн. серверов и ПК. Высокая точность и качество обеспечивались пятикратным уровнем избыточности при скрининге каждой молекулы. Полный скрининг был закончен за 24 дня. В результате из 3.57 млрд. молекул были отобраны 376,064 как потенциальные кандидаты для разработки противоядия, из них 12,000 – как наиболее перспективные. По свидетельству специалистов Оксфордского университета, если бы эта работа делалась традиционными методами, она бы длилась несколько лет, а не четыре недели.

Что касается форм предоставления услуг, то деятельность компаний не ограничивается производством системных программных средств для грид. Они участвуют в конкретных проектах, разрабатывают и ставят на грид

приложения, а также создают инфраструктуру грид, используя ее для оказания провайдерских услуг.

Базовая платформа компании United Devices - Grid MP Enterprise предназначена для интеграции внутрикорпоративных ресурсов: кластеров, серверов, рабочих станций и ПК. Следующий вариант - Grid MP Alliance позволяет создавать грид, объединяющий нескольких участников-поставщиков ресурсов, и позволяет предоставлять избыточные мощности сторонним потребителям.

United Devices поддерживает также открытую публичную инфраструктуру, действующую в Интернет. Добровольцам предоставляется Агент для свободного подключения к Grid MP Global, сейчас объем ресурсов в нем оценивается компанией в 2 миллиона ПК и серверов.

Другая инфраструктура - Grid MP On Demand, построенная на частной коммуникационной сети компании Gateway, поставляет обрабатывающие ресурсы суперкомпьютерного уровня на платной основе. В ней существуют два типа соглашений по обслуживанию.

- Standard MP On Demand предоставляет географически распределенные компьютеры в режиме разделяемого использования. В инфраструктуру входит свыше 6,000 ПК суммарной мощности более 11 Терафлопс.

- В Premier MP On Demand ресурсы, расположенные в центре обработки данных компании Gateway, выделяются пользователям персонально.

4.3. Технологии одноуровневого грид

4.3.1. Платформа DCGrid и вопросы безопасности

Платформа DCGrid компании Entropia, обладая всеми чертами архитектуры одноуровневого грид, имеет и ряд особенностей. Её назначение – повышение эффективности парка ПК предприятия: простаивающие или недозагруженные вычислительные мощности используются в стиле грид как интегрированные ресурсы для централизованно управляемой обработки заданий.

Инфраструктуру DCGrid составляют один управляющий узел - Сервер и множество обрабатывающих узлов. Сервер реализует функции пользовательского и административного интерфейса, управления заданиями и распределения ресурсов. Административный интерфейс, реализованный через Web-браузер, служит средством управления функционированием системы на основе приоритетов, зависящих от пользователя, проекта, времени суток.

Управляющие компоненты сервера DCGrid решают задачу создания надежной, безопасной, предсказуемой среды в условиях, когда ресурсная составляющая инфраструктуры образуется из ПК, имеющих собственных владельцев, и не является ни надежной (машины могут выключаться и перезагружаться), ни безопасной (посторонние лица могут иметь дистанционный или прямой доступ к машине), ни предсказуемой (в любой момент машина может быть полностью занята владельцем).

Агенты DCGrid на обрабатывающих узлах выполняет функции по: 1) управлению приложениями, включая инициацию/терминирование, доставку файлов, диагностике ошибок; 2) информированию центрального сервера о характеристиках и состоянии узлов.

Для всех платформ грид, которые поддерживают неотчуждаемые от владельцев ресурсы, в том числе и для DCGrid, особенно критичны вопросы безопасности. Мощный механизм “изоляции” приложений в DCGrid направлен на три аспекта безопасности:

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

Выполнение приложения грид происходит в среде виртуальной машины DCGrid - EVM, которая работает параллельно с процессами “хозяина” ПК. EVM [47] представляет собой программную оболочку, которая перехватывает все обращения приложения грид к ОС. Перехват реализуется путем известного технического приема подмены стандартных динамических библиотек (DLL) на библиотеки DCGrid. Таким образом гарантируется полный контроль со стороны EVM всех обращений приложения к ОС, а следовательно и к ресурсам ПК.

Эта технология бинарной модификации приложений разработана только для Windows x86 различных изданий: NT, 2000 и XP, что соответственно сужает класс обрабатывающих машин, с которыми может работать DCGrid. Однако, возможность выполнения любых приложений Windows в DCGrid не ограничена: приложение может быть написано на любом языке, не требуется исходных кодов, перекомпиляции или сборки, адаптация происходит автоматически.

Защита владельца основывается на соглашении о разделении ресурсов между владельцем и грид и регулируется настраиваемой “политикой аренды”. Эта политика определяет предельный уровень внешней (со стороны грид) загрузки таких ресурсов, как процессор, память, диск, количество нитей. EVM производит мониторинг потребления этих ресурсов, и, если предельный уровень превышает, она приостанавливает процессы внешнего приложения.

Защита обрабатываемого компьютера направлена на то, чтобы исключить возможность умышленного или неумышленного изменения (ошибки в программе, вирусы и пр.) приложением грид состояния арендуемого компьютера. Изолируя приложение, EVM не пропускает обращения к ОС, которые могли бы вызывать модификацию регистра, файловой системы и других критических ресурсов. Ограничивается также использование в приложении грид той части прикладного интерфейса Windows, которая позволяет управлять машиной (вход, перезагрузка) и такими устройствами, как дисплей, аудиосистема и сеть.

Защита приложения грид гарантирует целостность результатов и ограничивает доступ к данным, с которыми оно работает. Для этого, во-

первых, файлы данных хранятся в зашифрованном виде. Во-вторых, и для программных файлов, и для файлов данных вычисляется, постоянно хранится, передается вместе с файлом по сети его контрольная сумма.

4.3.2. Платформа LiveCluster

В отличие от компании Entropia компания DataSynapse (специализирующаяся в области финансовых приложений) считает, что ее платформа LiveCluster универсальна и может объединять в грид любые ресурсы: от мейнфреймов и кластеров до серверов и ПК, независимо от того, в каком режиме, выделенном или разделяемом с владельцем, они используются. При простой архитектуре одноуровневого грид, при минимальных накладных расходах на администрирование может быть достигнута высокая надежность обработки и эффективность использования ресурсов.

На грид платформы LiveCluster могут быть поставлены приложения, которые не были разработаны для распределенного выполнения, но для этого требуется их адаптация. Адаптация выполняется с помощью специального прикладного интерфейса (API) LiveCluster Application Enablers, что существенно проще, чем разработка распределенного приложения традиционными средствами типа MPI или PVM, хотя LiveCluster поддерживает и такие приложения. Средствами LiveCluster могут быть подготовлены также сериализуемые приложения (путем определения множества параллельно выполняемых подзаданий).

Особо следует отметить, что приложение может быть поставлено на грид в виде Web-службы. В этом случае LiveCluster виртуализирует обработку запросов к приложению, выделяя под нее те или иные ресурсы грид, а также обеспечивая клонирование обрабатывающих процессов и параллельность обработки. Поскольку соответствующее API основывается на стандартах SOAP, XML и WSDL, это делает платформу максимально приближенной к OGSA.

Для того, чтобы обеспечить максимально высокую, близкую к 100% загрузку при управлении выделенными ресурсами в LiveCluster применяется особый метод “адаптивной” диспетчеризации. Обычные методы выделяют заданию фиксированное количество процессоров в момент запуска, и все они закрепляются за заданием до его окончания. При таком способе невозможно обработать пакет многопроцессорных заданий так, чтобы все наличные процессоры грид были заняты постоянно: какое-то количество будет простаивать, если на них нельзя разместить ни одно из ожидающих в очереди заданий.

Адаптивная диспетчеризация позволяет стартовать заданию даже на одном процессоре, а затем по мере освобождения занимать дополнительные. Необходимо, однако учитывать, что такой метод подходит для ограниченного класса приложений, например, сериализуемых или подготовленных специальным образом MPI-приложений. Кроме того, такая диспетчеризация необходимо базируется на механизме миграции подзаданий, применение

которого работает в сторону понижения общей эффективности системы обработки.

Как архитектурную особенность платформы LiveCluster выделим способ взаимодействия управляющего центра (Server) и агентов на обрабатывающих узлах-компьютерах (Engine). Инициатива принадлежит Engine, который детектирует изменение состояния ресурсов на своем узле и сообщает об этом серверу, а тот реагирует, например, посылкой нового задания. По сравнению с более распространенной схемой, в которой сервер периодически опрашивает состояние ресурсов, этот подход лучше масштабируем, а также позволяет более просто адаптироваться к изменениям: отключению, занятию или добавлению отдельных узлов.

Платформа LiveCluster почти полностью, за исключением фоновых демонов и инсталляционной программы, реализована на Java, и ее компоненты не зависят от ОС, на которую устанавливаются. Распределенные компоненты могут взаимодействовать по протоколам глобальных или локальных сетей, причем могут быть настроены различные транспортные протоколы и режимы шифрования сообщений.

Engine имеет объем примерно 1 Мб, а его установка занимает меньше минуты времени. После того как Engine установлен, не требуется никакого администрирования: сервер автоматически управляет распределением заданий и, кроме того, осуществляет обновление кодов самого Engine.

5. ЗАКЛЮЧЕНИЕ

Появление масштабных инфраструктурных проектов обусловило переход к новой фазе развития ПО грид. Сыгравший исключительную роль Globus Toolkit перестает быть единственным доступным средством для создания грид, но по-прежнему сохраняет значение как стандарт дистанционного взаимодействия и инструментально-базовый пакет для разработки служб.

Новое поколение ПО грид развивается в форме платформ, то есть взаимосогласованных наборов средств, которые со значительно большей полнотой поддерживают развертывание и обслуживание грид, а также организуют функционирование распределенной инфраструктуры как единой операционной среды. Последний аспект проявляется в том что платформы начинают реально решать задачи обеспечения надежности, детерминированности, безопасности, управляемости грид с помощью механизмов диспетчеризации, мониторинга заданий и устройств, учета и протоколирования.

В связи с появлением нескольких альтернативных платформ - помимо упомянутых коммерческих, это и европейские платформы DataGrid, Unicore [48] и американская VDT [49] - на передний план выходит задача обеспечения интероперабельности грид разных ВО, использующих различные исполнительные среды. Важными представляются два следующих вопроса. Первый: каким образом можно обратиться из одного грид к службе другого

грид? И второй, инструментальный вопрос: каким образом службу одной платформы, можно использовать при создании другой?

В работе [1] высказано мнение, что для этого необходима унификация протоколов и интерфейсов доступа к службам. Однако, уже имеющийся опыт создания коммунальных служб показывает, что этого недостаточно: требуется определение базового состава служб и унификация их семантики. Такая “семантическая” стандартизация потребует времени, но, пожалуй, только ее достижение может открыть дорогу для действительно широкого распространения технологий грид.

6. ЛИТЕРАТУРА

- [1]. *Foster I., Kesselman C., Tuecke S.* The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001.
<http://www.globus.org/research/papers/anatomy.pdf>
- [2]. <http://www.globus.org>
- [3]. <http://www.eu-datagrid.org>
- [4]. <http://www.griphyn.org>
- [5]. *Foster I., Kesselman C., J. Nick, Tuecke S.* The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.
<http://www.globus.org/research/papers/ogsa.pdf>
- [6]. *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.
- [7]. <http://www.globus.org/developer/news/20011112a.html>
- [8]. <http://www.eu-egee.org>
- [9]. *Graupner S., Pruyne J., Singhal S.* Making the Utility Data Center a Power Station for the Enterprise Grid, HP Laboratories Palo Alto.
<http://www.hpl.hp.com/techreports/2003/HPL-2003-53.pdf>
- [10]. <http://www.openpbs.org>
- [11]. <http://www.cs.wisc.edu/condor/>
- [12]. <http://www.platform.com/products/LSFfamily/>
- [13]. <http://www.sun.com/software/gridware/sge.html>
- [14]. <http://setiathome.berkeley.edu>
- [15]. <http://distributed.net>
- [16]. Towards “Cluster as Server”: An Integrated Approach to Workload and Systems Management for Compute Clusters, Technical Whitepaper,
http://www.platform.com/pdfs/whitepapers/CW_ClusAsServer_WP.pdf
- [17]. <http://www.systemimager.org>
- [18]. <http://www.redhat.com>
- [19]. <http://www.cfengine.com>
- [20]. <http://www.webmin.com>
- [21]. <http://www.bb4.org>

- [22]. <http://ganglia.sourceforge.net>
- [23]. <http://www.mrtg.jp>
- [24]. <http://www.kernel.org/software/mon>
- [25]. <http://www.csm.ornl.gov/torc/C3>
- [26]. <http://oscar.openclustergroup.org>
- [27]. <http://www.rocksclusters.org/>
- [28]. <http://www.netlib.org/mpi>
- [29]. <http://math-atlas.sourceforge.net>
- [30]. <http://www.netlib.org/benchmark/hpl>
- [31]. <http://www.gridtoday.com/03/1124/102289.html>
- [32]. <http://lcg.web.cern.ch/LCG>
- [33]. <http://www.lcfg.org>
- [34]. http://www.platform.com/pdfs/datasheets/Plt_Clusterware_DS.pdf
- [35]. <http://www.platform.com/products/LSF>
- [36]. <http://java.sun.com/products/javabeans>
- [37]. *Ferstl F., Haas A.* DRMAA interface specification (draft), 2001. http://www-unix.mcs.anl.gov/~schopf/ggf-sched/WG/drmaa_v0.txt
- [38]. <http://www.cs.wisc.edu/condor/classad>
- [39]. *Buyya R., Vazhkudai S.* Compute Power Market: Towards a Market-Oriented Grid, The First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001), Brisbane, Australia, May 16-18, 2001. <http://www.buyya.com/papers/cpm.pdf>
- [40]. *Buyya R., Abramson D., Giddy J., Stockinger H.* Economic Models for Resource Management and Scheduling in Grid Computing, Special Issue on Grid Computing Environments, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, USA, May 2002. <http://www.buyya.com/papers/emodelsgrid.pdf>
- [41]. <http://www.sun.com/software/gridware/sgeee53/wp-sgeee/wp-sgeee.pdf>
- [42]. <http://www.entropy.com>
- [43]. <http://www.datasynapse.com>
- [44]. <http://www.parabon.com>
- [45]. <http://www.ud.com>
- [46]. http://www.ud.com/rescenter/files/cs_dod.pdf
- [47]. *Calder B., Chien A., Wang Ju, Yang Don.* The Entropia Virtual Machine for Desktop Grids, 2003. http://www-cse.ucsd.edu/Dienst/UI/2.0/Print/ncstrl.ucsd_cse/CS2003-0773
- [48]. <http://www.unicore.org/documents/UNICOREPlus-Final-Report.pdf>
- [49]. <http://www.lsc-group.phys.uwm.edu/vdt>