

АНАТОМИЯ ГРИД

Создание Масштабируемых Виртуальных Организаций

Ян Фостер Карл Кессельман Стив Тьюке

* * *

THE ANATOMY OF THE GRID

Enabling Scalable Virtual Organizations

Ian Foster^{1,2} Carl Kesselman³ Steven Tuecke¹

¹ Mathematical and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439

² Department of Computer Science, University of Chicago, Chicago, IL 60637

³ Information Science Institute, University of Southern California, Marina del Rey, CA 90292

foster@mcs.anl.gov, tuecke@mcs.anl.gov, carl@isi.edu

Аннотация

Грид-компьютинг возник как новая важная область, отличающаяся от традиционного распределённого компьютеринга своей нацеленностью на инновационные приложения, как правило, связанные с необходимостью разделения крупномасштабных ресурсов, и, в некоторых случаях, на обеспечение возможности высокопроизводительной обработки данных. В данной статье очерчивается границы этой области. Во-первых, рассматривается “грид-проблема”, которая определяется как задача обеспечения гибкого, безопасного и согласованного разделения ресурсов в так называемых *виртуальных организациях* - динамичных объединениях отдельных пользователей, институтов и ресурсов. При такой постановке вопроса мы сталкиваемся с проблемами однозначной аутентификации, авторизации, обнаружения ресурсов и организации доступа к ним, а также рядом других сложных задач. Это и есть класс проблем, на решение которых нацелены грид-технологии. Далее мы представляем расширяемую и открытую *грид-архитектуру*, в которой протоколы, службы, интерфейсы прикладного программирования и инструментарий для разработки программного обеспечения распределены по категориям в соответствии с их ролью в обеспечении разделения ресурсов. Мы описываем требования, которым, по нашему мнению, должны удовлетворять любые подобного рода механизмы, и обсуждаем важность определения компактного набора “*интергрид*”-*протоколов*, поддерживающих интероперабельность между различными грид-системами. Наконец, мы рассматриваем, как грид-технологии соотносятся с другими современными технологиями, включая такие как корпоративная интеграция, провайдер прикладных услуг, провайдер услуг хранения и одноранговый (peer-to-peer) компьютеринг. Мы утверждаем, что грид-концепции и грид-технологии дополняют и внесут большой вклад в эти другие технологические подходы.

Содержание

Аннотация	2
Содержание	3
1 Введение	4
2 Появление виртуальных организаций	6
3 Сущность грид-архитектуры	9
4 Описание грид-архитектуры	11
4.1 Фабрикаты: Интерфейсы локального управления	12
4.2 Связь: Лёгкое и безопасное общение	14
4.3 Ресурс: Разделение отдельных ресурсов	16
4.4 Кооперация: Согласование множества ресурсов	18
4.5 Приложения	21
5 Практика грид-архитектуры	22
6 “В грид-среде”: Необходимость интергрид-протоколов	23
7 Соотношение грид с другими технологиями	24
7.1 World Wide Web	24
7.2 Провайдеры услуг хранения и приложений	24
7.3 Системы корпоративного компьютеринга	25
7.4 Интернет и одноранговый компьютеринг	26
8 Другие перспективы грид	26
9 Заключение	29
Приложение: Определения	30
Библиография	34

1 Введение

Термин “грид” стал использоваться с середины 90-х годов для обозначения некой инфраструктуры распределённого компьютеринга, предлагаемой для обслуживания передовых научных и инженерных проектов. С тех пор были достигнуты значительные успехи в построении такой инфраструктуры (например, [10, 15, 42, 54]), и толкование термина “грид”, по крайней мере, в популярном восприятии существенно расширилось, и стало охватывать всё - от передовых сетевых решений до разработок в области искусственного интеллекта. Возникает сомнение, действительно ли этот термин имеет какое-то реальное содержание и смысл. Действительно ли имеет место отличная от других “грид-проблема”, и поэтому нужны новые “грид-технологии”? Если это так, то в чём сущность этих технологий, и какова сфера их применимости? Несмотря на то, что многочисленные группы проявляют интерес к грид-концепциям и, в значительной степени, имеют общее видение грид-архитектуры, мы пока не достигли консенсуса в ответах на поставленные выше вопросы.

Цель нашей статьи показать, что идея грид действительно мотивируется реальной и конкретной проблемой, и что возникает вполне определённая грид-технологическая база, которая ориентирована на наиболее значимые аспекты этой проблемы. В ходе изложения мы представим детальную архитектуру и план развития современных и будущих грид-технологий. Более того, мы покажем, что несмотря на то, что грид-технологии в настоящее время отличаются от других основных технологических направлений таких, как интернет, корпоративный, распределённый и одноранговый компьютеринг, эти другие технологии могут извлечь значительную выгоду от вхождения в сферу проблем, разрешаемых с помощью грид-технологий.

Реальной и конкретной проблемой, подчёркивающей значимость грид-концепции, является согласованное разделение ресурсов и решение задач в динамичных, многопрофильных виртуальных организациях. Разделение, которое мы имеем в виду, это главным образом не обмен файлами, а скорее прямой доступ к компьютерам, программному обеспечению, данным и другим ресурсам так, как это востребовано рядом возникающих в промышленности, науке и технике стратегий совместного решения задач и посредничества в предоставлении ресурсов. Это разделение обязательно жестко контролируется провайдерами ресурсов и потребителями, ясно и чётко определяющими что разделяется, кому разрешено разделение и условия, на которых разделение выполняется. Объединение отдельных специалистов и/или институтов, определённое такими правилами разделения образует то, что мы называем виртуальной организацией (virtual organization) – ВО.

Приведём следующие примеры ВО:

- провайдеры прикладных услуг, провайдеры услуг хранения, провайдеры квантов вычислений, консультанты, приглашаемые производителем автомобилей для разработки сценария развития событий при проектировании нового производства;
- участники промышленного консорциума, финансирующие создание нового самолёта;
- команды кризисного менеджмента, а также базы данных и системы моделирования, которые используются этими командами при выборе способа действия в связи с непредвиденной ситуацией;

- участники многолетних, крупных международных объединений в области физики высоких энергий.

Каждый из этих примеров иллюстрирует основанный на совместном использовании крупного пула вычислительных ресурсов и данных подход к организации вычислений и решению проблем.

Как показывают эти примеры, ВО чрезвычайно разнообразны по своей цели, масштабу, размеру, продолжительности существования, структуре, общественному и социологическому статусу. Тем не менее, внимательное изучение определяющих технологических требований приводит нас к выявлению некоего широкого набора общих для всех ВО интересов и потребностей. В частности, мы можем отметить, что для разделения необходимо обеспечить:

- чрезвычайно гибкие отношения в широком диапазоне возможных сетевых решений: от схемы “клиент – сервер” до схемы “одноранговая сеть”;
- сложный и высокоуровневый контроль за тем, как используются разделяемые ресурсы, включая средства мелкоструктурного контроля доступа, делегирование и применение локальных и глобальных политик;
- возможности разделения разнообразных ресурсов: от программ, файлов и данных до компьютеров, датчиков и сетей;
- разнообразные по критериям производительности и стоимости пользовательские режимы (от однопользовательского до многопользовательского), предусматривающие решение проблем обеспечения качества обслуживания, планирования, совместной загрузки и учёта использования ресурсов.

Современные технологии распределённого компьютеринга не рассматривают перечисленных вопросов и требований. Например, современные интернет-технологии адресованы к проблемам коммуникации и информационного обмена между компьютерами, но при вычислениях не обеспечивают интегрированных подходов к согласованному использованию ресурсов, находящихся в разных сайтах. Системы бизнес-партнёрства, так называемые B2B [52], нацелены на разделение информации (часто через аппарат централизованных услуг). Подобным же образом действуют технологии виртуальных корпораций, хотя здесь разделение со временем может быть расширено до приложений и физических устройств (например, [8]). Технологии распределённого корпоративного компьютеринга такие, как CORBA и Enterprise Java, обеспечивают разделение ресурсов внутри одной организации. Технология DCE поддерживает безопасное разделение ресурсов между сайтами, но большинство ВО считают эту технологию слишком тяжёлой и не гибкой. Провайдеры служб хранения (Storage Service Providers – SSP) и Провайдеры служб приложений (Application Service Providers – ASP) предоставляют организациям-клиентам доступ к внешним ресурсам памяти и вычислительным мощностям, но только ограниченными способами: например, ресурсы SSP обычно связаны с потребителем через частную виртуальную сеть (Virtual Private Network – VPN). Появляющиеся компании интернет-компьютеринга проводят в международном масштабе [28] поиск простаивающих компьютеров, но до настоящего времени поддерживают только жёстко централизованный доступ к таким ресурсам. Итак, современные технологии либо не охватывают все типы ресурсов, либо не обеспечивают необходимые для создания ВО гибкость и управляемость связями, возникающими при разделении ресурсов.

Вот такова картина перед появлением грид-технологий. Исследования и разработки, выполненные внутри грид-сообщества за последние пять лет, привели к созданию протоколов, служб и инструментариев, которые в полной мере соответствуют проблемам, с которыми мы сталкиваемся при организации масштабируемых ВО. Эти технологии предусматривают:

- решения по безопасности, которые поддерживают управление полномочиями (цифровыми мандатами) и политиками планирования при проведении расчётов на пространственно распределённых ресурсах нескольких организаций;
- протоколы управления ресурсами и службами, которые поддерживают безопасный удалённый доступ к вычислительным мощностям и ресурсам данных, а также коаллокацию (совместное распределение – co-allocation) множества ресурсов;
- протоколы информационных запросов и службы, предоставляющие информацию о конфигурации и статусе ресурсов, организаций и служб;
- службы управления данными, которые осуществляют размещение и транспортировку наборов данных между системами хранения и приложениями.

Будучи сфокусированы на динамическом, межкорпоративном разделении ресурсов, грид-технологии дополняют, а отнюдь не конкурируют с существующими технологиями распределённого компьютеринга. Например, корпоративные системы распределённого компьютеринга могут использовать грид-технологии для разделения ресурсов, находящихся вне пределов организации; в области применения систем ASP/SSP грид-технологии могут быть использованы при создании динамичных рынков ресурсов хранения и вычислительных мощностей и тем самым для преодоления ограничений, присущих современным статичным конфигурациям. Ниже мы более подробно обсудим связи грид с этими технологиями.

В последующем материале данной статьи мы расширим рассмотрения по каждой из этих позиций. Наши цели состоят в том, чтобы (1) внести ясность в понимание сути ВО и грид-компьютеринга для тех, кто не знаком с этой областью информационных технологий; (2) способствовать становлению грид-компьютеринга, как дисциплины, путём введения стандартной терминологии и определения общей архитектурной схемы; и (3) ясно указать, как грид-технологии соотносятся с другими технологиями, объяснив как то, почему существующие технологии до сих пор не решают проблемы грид-компьютеринга, так и то, какую пользу они могут извлечь из грид-технологий.

Мы убеждены, что ВО обладает потенциалом, способным драматически изменить наши методы использования компьютеров при решении задач, во многом подобно тому, как Web изменил наши способы информационного обмена. Так, представленные здесь примеры наглядно иллюстрируют, что для многих разных дисциплин и сфер деятельности необходимость вовлечения в процессы сотрудничества является обстоятельством фундаментальным и не ограничивается наукой, технологиями и бизнесом. Вот почему для широкого применения концепций ВО важны грид-технологии.

2 Появление виртуальных организаций

Рассмотрим следующие четыре сценария:

1. Компания, которой необходимо получить обоснованное решение по вопросу размещения нового производства, запрашивает у некоего провайдера ASP сложную программу финансового прогнозирования, обеспечивая ей при этом доступ к

соответствующим частным историческим сведениям, хранящимся в корпоративной базе данных, размещённой на системах хранения, управляемых неким провайдером SSP. В течение периода выработки решения, сценарии “что-если” проигрываются совместно и интерактивно, даже если руководящие отделения компании, участвующие в подготовке решения, находятся в разных местах. Провайдер ASP сам заключает с **Провайдером квантов времени для вычислений** (Cycle provider – CP) контракты на дополнительные прогоны, связанные с непредвиденными ситуациями, требуя, разумеется, чтобы эти кванты удовлетворяли соответствующим требованиям безопасности и производительности.

2. Промышленный консорциум, образованный для выполнения анализа осуществимости сверхзвукового самолёта нового поколения, проводит цикл высокоточного многоаспектного моделирования всего самолёта. При этом моделировании объединяются в единое целое частные компоненты программного обеспечения, разработанные различными участниками, причём каждый из этих компонентов работает на компьютерах его владельца и имеет доступ к соответствующим базам данных проекта и другим данным, предоставленным консорциуму его участниками.

3. Команда кризисного менеджмента реагирует на чрезвычайную ситуацию, возникшую в результате выброса вредных химических веществ, используя модели локальной погоды и почвы, оценивает распространённость загрязнения, определяет воздействие, оказываемое на местное население, а также и на такие географические объекты, как реки и водохранилища, создаёт (возможно, основанный на моделях химических реакций) краткосрочный план смягчения ситуации, планирует и координирует процессы эвакуации, госпитализации и пр., руководит персоналом, ликвидирующим последствия аварии.

4. Тысячи физиков из сотен лабораторий и университетов всего мира объединились с целью проектирования и создания большого детектора в Европейском центре физики высоких энергий (European high energy physics laboratory – CERN), проведения на нём экспериментов и анализа получаемых данных. При обработке данных экспериментов участники этой программы объединяют в общий фонд компьютерные, накопительные и сетевые ресурсы для создания, так называемой среды “Data Grid”, способной обеспечить анализ петабайтных объёмов данных [21, 40, 49].

Эти четыре примера различаются во многих отношениях: по числу и профессиям участников, видам деятельности, продолжительности и масштабу взаимодействия и по разделяемым ими ресурсам. Но они также имеют много общего, что мы и покажем ниже (см. Рисунок 1).

В каждом примере - непредсказуемое число участников, в той или иной степени ранее связанных друг с другом (а, возможно, и вовсе не связанных), хотят разделять ресурсы для выполнения некоторой задачи. Более того, в данном случае разделение подразумевает нечто большее, чем простой обмен документами (как в “виртуальных корпорациях” [17]): оно может быть связано с прямым доступом к удалённым компьютерам, данным, программному обеспечению, датчикам и другим ресурсам. Например, члены консорциума могут обеспечить доступ к специальному программному обеспечению, данным и/или к пулу их вычислительных ресурсов.

Разделение ресурсов обусловлено: каждый владелец ресурсов устанавливает доступность ресурсов, задаёт ограничения, определяющие когда, где и что может быть сделано. Например, участник ВО от Р (см. Рисунок 1) мог бы позволить партнёрам запускать их службы моделирования только для “простых” задач. Потребители ресурсов,

в свою очередь, могли бы указывать ограничения на свойства ресурсов, к работе с которыми они подготовлены. Например, участник ВО от Q мог бы соглашаться только на объединённые вычислительные ресурсы, сертифицированные как "безопасные". Осуществление таких ограничений требует наличия специальных механизмов для задания политик управления, **аутентификации** (установления идентичности – authentication) потребителя или ресурса, и **авторизации** (предоставления права – authorization) выполнения операции в соответствии с установленными отношениями разделения.

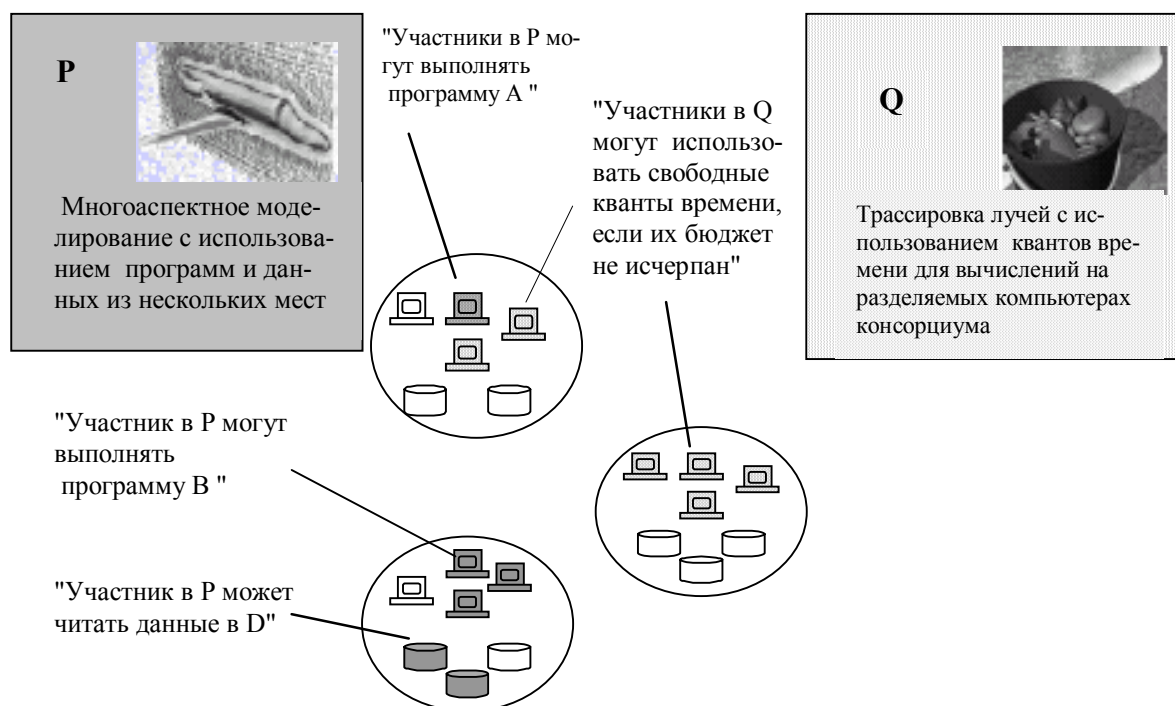


Рисунок 1: Фактически организация может участвовать в одной или нескольких ВО, предоставляя для разделения некоторые или все свои ресурсы. На рисунке показаны (в овалах) три фактически существующие организации и две ВО: P, которая объединяет участников консорциума аэрокосмического проекта, и Q, которая связывает коллег, согласных разделять свободные кванты компьютерного времени, например, при расчётах трассировки лучей. Организация, показанная в левой части рисунка, участвует в P, а показанная справа участвует в Q, а третья является членом как P, так и Q. Политики, регулирующие доступ к ресурсам (резюмируемые в "квотах") разнятся, соответственно, для фактических организаций, ресурсов и ВО, вовлечённых в общий процесс.

Отношения разделения могут со временем динамично изменяться в зависимости от условий применения выделяемых ресурсов, характера разрешённого доступа и состава участников, которым разрешён доступ. И эти отношения вовсе необязательно связаны с необходимостью явного поимённого указания участников, а вполне могут быть неявно определены через политики управления доступом к ресурсам. Например, организация может обеспечить доступ любому, кто докажет, что он "заказчик" или "студент".

Динамичный характер отношений разделения означает, что нам нужны механизмы обнаруживающие и характеризующие сущность связей, которые существуют в данный момент времени. Например, новый участник, вступающий в ВО Q, должен быть в

состоянии определить, к каким ресурсам может быть получен доступ, “качество” этих ресурсов и политики управления доступом.

Отношения разделения часто представляются не просто схемой “клиент-сервер”, а достаточно развитой одноранговой структурой типа “точка – точка”, где провайдеры могут оказаться потребителями, а отношения разделения могут существовать между любым подмножеством участников. Отношения разделения могут объединяться для согласованного использования на многих ресурсах, каждый из которых принадлежит одной из организаций. Например, расчёты, начатые в ВО Q на одном пуле вычислительных ресурсов, могут позже получить доступ к данным или запустить решение подзадач где-либо ещё. В таких ситуациях важной становится способность контролируемые способами делегировать полномочия, это же касается механизмов согласования операций (так называемого, *совместного планирования* – cosheduling) на всём множестве ресурсов.

В зависимости от ограничений, наложенных на разделение, и цели его применения один и тот же ресурс может быть использован разными способами. Например, при одной системе отношений компьютер может быть использован только для прогона определённого блока программного обеспечения, в то время как при другой - его временные кванты могут разделяться для любого счёта. Вследствие отсутствия априорных знаний о том, как некий ресурс может быть использован, показатели производительности, вероятностные характеристики и ограничения (то есть качество обслуживания) могут быть составной частью условий, накладываемых на разделение или использование ресурса.

Эти характеристики и требования определяют то, что мы называем *виртуальной организацией*, концепцию, которая по нашему убеждению становится фундаментальной для многих современных процессов вычислений и обработки данных. ВО позволяют в корне отличным группам организаций и/или отдельным пользователям контролируемо разделять ресурсы, так чтобы они могли сотрудничать при достижении некой общей цели.

3 Сущность грид-архитектуры

Создание, управление и использование динамических, межведомственных ВО, разделяющих ресурсы, требует новой технологии. Мы структурируем наше обсуждение этой технологии в терминах *грид-архитектуры*, которая устанавливает фундаментальные системные компоненты, определяет цели и функции этих компонент и показывает, как эти компоненты взаимодействуют друг с другом.

Определение грид-архитектуры мы начнём с предпосылки, что для обеспечения эффективной деятельности ВО необходимо иметь возможность устанавливать отношения разделения между *любыми* потенциальными участниками. Таким образом, центральной проблемой, требующей разрешения, оказывается *интероперабельность* (взаимодействие различных программных и аппаратных средств – interoperability). В контексте рассмотрения сетевых технологий интероперабельность означает общность протоколов. Поэтому наша грид-архитектура, во-первых, прежде всего, является архитектурой *протоколов*, определяющих базовые механизмы, посредством которых пользователи и ресурсы ВО договариваются, устанавливают, управляют и используют отношения разделения. Основанная на стандартах открытая архитектура способствует расширяемости, интероперабельности, мобильности и совместному использованию общих

программ; стандартные протоколы облегчают определение стандартных служб, которые обеспечивают усовершенствование возможностей. Для поддержки программирования абстрактных конструкций, необходимых при создании удобного в использовании грид, мы можем также разработать, так называемые *Интерфейсы Прикладного Программирования* (Application Programming Interfaces – *API*) и *Инструментарий Разработки Программного обеспечения* (Software Development Kits – *SDK*), определения которых приведены в Приложении. Вместе, эта технология и архитектура составляют то, что часто называется как *промежуточное программное обеспечение* (службы, необходимые для поддержки общего набора приложений в распределённой сетевой среде – “middleware”[3]), хотя мы избегаем использования этого термина здесь из-за его неопределённости. В последующем мы обсудим каждое из высказанных сейчас соображений.

Почему интероперабельность является столь фундаментальной системной возможностью? Дело состоит в том, что мы должны гарантировать формирование отношений разделения между произвольными группами, вступление новых участников динамично и через различные платформы, языки и программные среды. В таком контексте механизмы приносят мало пользы, если они не определены и не реализованы так, чтобы их интероперабельность не лимитировалась границами организаций, политиками управления и типами ресурсов. Без интероперабельности ВО приложения и участники вынуждены устанавливать двусторонние договорённости о разделении, поскольку нет гарантии, что механизмы, используемые между любыми двумя группами могут быть расширены для любых других групп. Без такой гарантии динамичное создание ВО вообще невозможно, а количество типов ВО, которые могут быть сформированы, строго ограничено. Точно также как Web революционизировала разделение информации, предоставив для целей информационного обмена универсальный протокол и синтаксис (HTTP и HTML), нам необходимы стандартные протоколы для повсеместного разделения ресурсов.

Почему протоколы крайне необходимы для интероперабельности? Определение протокола устанавливает, как для реализации заданной дисциплины работы элементы одной распределённой системы взаимодействуют с элементами другой, и структуру информации, передаваемой во время этого взаимодействия. Такая нацеленность на внешние факторы (на взаимодействия), а не на внутренние (на программное обеспечение, характеристики ресурсов) имеет важные прагматические достоинства. ВО имеют тенденцию к постоянному изменению, поэтому механизмы, используемые для обнаружения ресурсов, установления идентичности, определения права доступа и инициализации разделения должны быть гибкими и лёгкими настолько, чтобы договорённости о разделении ресурсов можно было бы быстро устанавливать и изменять. Поскольку ВО дополняют, а не заменяют существующие организации, механизмы разделения не могут требовать существенных изменений в локальных политиках управления и должны позволять отдельным институтам поддерживать предельно жёсткий контроль их собственных ресурсов. Поскольку протоколы определяют взаимодействия между компонентами, а не их реализацию, локальное управление сохраняется.

Почему важны службы? Служба (см. Приложение) определяется исключительно протоколом, посредством которого она общается, и дисциплиной, которую она реализует. Определение стандартных служб – для доступа к вычислительным ресурсам, доступа к данным, обнаружения ресурсов, совместного планирования, репликации данных и так далее – позволяет нам усовершенствовать службы, предлагаемые участникам ВО, а также

абстрагироваться от специфических деталей ресурсов, которые помешали бы разработке ВО приложений.

Почему мы также обсуждаем здесь возможности API и SDK? Конечно, есть нечто более значимое для ВО чем интероперабельность, протоколы и службы. Разработчики должны иметь возможность создавать изоцированные приложения в сложной и динамичной исполнительской среде. Пользователи должны иметь возможность работать с этими приложениями. Работоспособность приложений, их исправность, стоимость разработки и сопровождения – всё это чрезвычайно важные факторы. Стандартные абстракции, API's и SDK's помогают ускорить разработку программ приложений, обеспечивают совместное использование общих программ и улучшают переносимость приложений. API's и SDK's являются дополнением, а не альтернативой протоколам. Без стандартных протоколов интероперабельность может быть достигнута на уровне API только путём использования всюду единой реализации приложения, что невыполнимо во многих заинтересованных ВО, или на основе знания каждой реализацией деталей каждой другой реализации. (Метод *Json*, используемый в [6], предлагает загрузку процедуры протокола на удалённый сайт, что не позволяет обойти это требование).

Итак, кратко, в нашем рассмотрении грид-архитектуры особое значение придаётся, во-первых, идентификации и определению протоколов и служб и, во-вторых, API's и SDK's.

4 Описание грид-архитектуры

При описании грид-архитектуры мы ставим своей целью не полное перечисление всех необходимых протоколов (а также служб, API и SDK), а скорее, формулирование требований для главных классов компонентов. В результате мы представим для обсуждения расширяемую, открытую архитектурную схему, внутри которой могут быть помещены решения ключевых требований ВО. В нашей архитектуре и последующем обсуждении компоненты упорядочены по уровням так, как это показано на Рисунке 2. Компоненты внутри каждого уровня имеют определённые общие характеристики, но могут быть построены на основе возможностей и режимов, обеспечиваемых любым нижним уровнем.

При определении различных уровней грид-архитектуры мы будем следовать принципам “модели песочных часов” (“hourglass model”) [1]. Узкое горло песочных часов устанавливает небольшой базовый набор абстракций и протоколов (например, таких как TCP и HTTP в интернет), на основе которых могут быть отображены многие различные дисциплины (режимы) верхнего уровня (вершина песочных часов), и которые сами могут быть отображены на основе многих различных нижележащих технологий (основание песочных часов). По определению, количество протоколов, установленных горлом должно быть небольшим. В нашей архитектуре горло песочных часов состоит из протоколов *Ресурсов* и протоколов *Связи*, которые облегчают разделение отдельных ресурсов. Протоколы этих уровней сконструированы так, что могут быть реализованы поверх разнообразного ряда типов ресурсов, определённых на уровне *Фабрикатов*. В свою очередь, они могут быть использованы для разработки широкого ряда глобальных служб и специфических прикладных режимов на уровне *Кооперации*, названном так, потому, что именно здесь достигается согласованное (совместное) использование множества ресурсов.



Рисунок 2: Многоуровневая грид-архитектура и её соотношение с Интернет-архитектурой протоколов. Поскольку Интернет-архитектура протоколов простирается от сетевого уровня до прикладного, здесь показано отображение из грид-уровней в Интернет-уровни.

Представленное нами описание архитектуры является высокоуровневым и устанавливает немного ограничений на конструкцию и реализацию. Чтобы сделать это абстрактное обсуждение более конкретным, мы, с целью иллюстрации, кратко охарактеризуем протоколы, определённые в Инструментальном комплексе Globus (Globus Toolkit) [32] и используемые в таких грид-проектах, как NSF National Technology Grid [54], NASA Information Power Grid [42], DOE’s DISCOM [10], GriPhyN (www.griphin.org), NEESgrid (www.neesgrid.org), Particle Physics Data Grid (www.ppdg.net), European Data Grid (www.eu-datagrid.org). Более подробное описание этих протоколов будет представлено в следующей статье.

4.1 Фабрикаты: Интерфейсы локального управления.

Грид-уровень **Фабрикатов** (Fabric layer) предоставляет ресурсы, при разделённом доступе к которым грид-протоколы работают в качестве связующих механизмов: например, вычислительные ресурсы, системы хранения, каталоги, сетевые ресурсы и сенсоры. “Ресурс” может быть логической сущностью, например такой, как распределённая файловая система, кластер компьютеров или распределённый пул компьютеров; в таких случаях применение ресурса может повлечь использование внутренних протоколов (например, протоколов сетевой файловой системы NFS или протоколов управления системными процессами сопровождения кластерного ресурса), но они не имеют отношения к грид-архитектуре.

Компоненты уровня Фабрикатов реализуют локальные, специфические для ресурсов операции, которые выполняются на заданных ресурсах (физических или логических) в результате операций разделения, происходящих на более высоких уровнях. Таким образом, существует тесная и деликатная взаимозависимость между функциями, реализуемыми на уровне Фабрикатов, с одной стороны, и предусмотренными операциями разделения, с другой. Обогащение функциональности уровня Фабрикатов открывает возможность применения более сложных операций разделения; в то же время, если мы установим мало требований к элементам уровня Фабрикатов, то распространение грид-инфраструктуры упрощается. Например, поддержка на этом уровне функции

предварительного резервирования ресурсов делает возможным для служб более высоких уровней удобно агрегировать ресурсы для их совместного планирования, что в противном случае было бы недостижимо. Тем не менее, поскольку на практике мало ресурсов поддерживают встроенное предварительное резервирование, требование предварительного резервирования увеличивает стоимость встраивания в грид новых ресурсов.

Практика подсказывает, что, как минимум, ресурсы должны, с одной стороны, обладать *справочными* (enquiry) механизмами, которые разрешают раскрывать их структуру, состояние и возможности (например, поддерживают ли ресурсы предварительное резервирование), и с другой - механизмами *управления ресурсами* (resource management), осуществляющими некоторый контроль предоставляемого качества обслуживания. В приведённом ниже кратком и неполном перечне охарактеризованы механизмы поддержки ресурсов.

- *Вычислительные ресурсы*: Необходимы механизмы для запуска программ, а также мониторинга и контроля выполнения результирующих процессов. Механизмы управления, позволяющие контролировать выделенные процессам ресурсы, оказываются полезными в качестве аппарата предварительного резервирования. Справочные функции необходимы для получения характеристик аппаратуры и программного обеспечения, а также релевантной информации о текущем состоянии ресурсов, например, какова текущая загрузка и состояние очереди на обслуживание в случае использования ресурсов, управляемых планировщиком.
- *Ресурсы хранения*: Необходимы механизмы для размещения и извлечения файлов. Весьма полезны высокопроизводительные средства пересылки информации (например, схема с чередованием данных – data striping) и подобного рода специальные продукты от независимых разработчиков [56]. Также полезны и механизмы чтения/записи подмножеств файлов, и/или выполняющие выборку удалённых данных, или функции редукции [14]. Механизмы управления, которые позволяют воздействовать на параметры ресурсов, выделенных для передачи данных (объём памяти, производительность дисков, пропускная способность сети, центральный процессор), полезны в качестве механизмов предварительного резервирования. Справочные функции необходимы для определения характеристик аппаратуры и программного обеспечения, а также соответствующей информации для решения вопросов загрузки, например, о доступном объёме памяти или использовании полосы пропускания.
- *Сетевые ресурсы*: Здесь могут оказаться полезными механизмы управления, которые воздействуют на возможности ресурсов (назначение приоритетов, резервирование), выделенных для передач информации по сети. Справочные функции должны быть предусмотрены для определения характеристик сети и её загрузки.
- *Репозитории программ*: Этот особый вид ресурсов хранения нуждается в механизмах управления источником исходных версий программ и объектного кода: например, в такой управляющей системе как CVS.

- *Каталоги*: Для этого специального вида ресурсов хранения нужны механизмы, реализующие обработку запросов к каталогам и операции обновления: например, реляционная база данных [9].
- *Инструментальный комплекс Globus*: Инструментарий Globus был разработан (главным образом) для использования существующих компонентов уровня Фабрикатов, включая поставляемые производителями протоколы и интерфейсы. Тем не менее, даже если поставщик не обеспечивает необходимый для уровня Фабрикатов режим, Инструментарий Globus включает средства для реализации отсутствующей функциональной возможности. Так Инструментарий Globus содержит справочное программное обеспечение для получения информации о структуре и состоянии различных широко распространённых видов ресурсов таких, как компьютеры (версия используемой операционной системы, конфигурация аппаратуры, загрузка [27], ситуация в очереди планировщика), системы хранения (например, о доступном пространстве) и сети (например, текущая и прогнозируемая загрузка [48, 58]) и для упаковки этой информации в формате, который облегчает реализацию протоколов более высоких уровней, особенно уровня Ресурсов. С другой стороны, как правило, управление ресурсами является сферой действия локальных менеджеров ресурсов. Одно исключение из этого правила имеет место в General-purpose Architecture for Reservation and Allocation – **GARA** [33], содержащей, так называемый, слот - менеджер (“slot manager”), который может быть использован для реализации предварительного резервирования ресурсов, не обладающих такой способностью. В других случаях, например, для систем PBS (Portable Batch System - PBS)[51] и Condor [45, 46] разработаны специальные усовершенствования, выполняющие функции предварительного резервирования.

4.2 Связь: Лёгкое и безопасное общение

Уровень Связи (Connectivity layer) устанавливает базовый набор коммуникационных и аутентификационных протоколов, необходимых для выполнения грид-специфических сетевых транзакций. Коммуникационные протоколы обеспечивают возможность обмена данными между ресурсами уровня Фабрикатов. Для обеспечения безопасных криптографических механизмов, используемых при верификации идентичности пользователей и ресурсов, протоколы аутентификации основываются на коммуникационных службах.

Требования, предъявляемые к протоколам уровня Связи, включают решение вопросов передачи информации, её маршрутизации и присваивания имён. Несмотря на существование определённых альтернатив, всё же почти во всех практических ситуациях эти протоколы будут почерпнуты из стека протоколов TCP/IP: конкретно, из уровня интернет (протоколы IP и ICMP), уровня передачи данных (протоколы TCP, UDP), и уровня приложений (протоколы DNS, OSPF, RSVP и др.) многоуровневой архитектуры протоколов интернет [7]. Это, конечно, не означает, что в будущем грид-связи не потребуют новых протоколов, которые будут учитывать конкретные типы сетевых механизмов.

Что касается вопросов безопасности на уровне Связи, отметим, что из-за сложности проблемы безопасности важно, чтобы любые решения, по возможности

основывались на существующих стандартах. Для решения вопросов связи применимы многие из стандартов безопасности, разработанные в контексте пакета протоколов интернет.

Механизмы аутентификации для среды ВО должны обладать следующими свойствами [16]:

- *Однократная регистрация:* Пользователи должны иметь возможность регистрироваться (“log on”) в системе только однажды и затем иметь доступ к множеству грид-ресурсов, определённых на уровне Фабрикатов, без дополнительного вмешательства со стороны пользователя.
- *Делегирование* [35, 40, 45]: Пользователь должен иметь возможность наделять программу способностью выполняться от имени этого пользователя так, что программа может иметь доступ к ресурсам, на которых данный пользователь авторизован. Программа также должна (дополнительно) иметь возможность обусловлено делегировать подмножество своих полномочий другой программе (иногда это называется ограниченным делегированием).
- *Интеграция с различными локальными механизмами безопасности:* Каждый сайт или провайдер ресурсов может применять самые разнообразные локальные механизмы безопасности, включая Kerberos и средства безопасности Unix. грид-службы безопасности должны уметь взаимодействовать с этими разнообразными локальными механизмами. Практически невозможно требовать единообразия локальных механизмов безопасности, а скорее необходимо обеспечить возможность отображения грид-механизмов безопасности на аппарат локальной среды.
- *Отношения доверия, опирающиеся на пользователя:* Для того, чтобы пользователь мог работать с пулом ресурсов, предоставленных многими провайдерами, как с единым целым, система безопасности не должна требовать от каждого провайдера ресурсов согласования или взаимодействия с каждым другим провайдером при конфигурировании среды безопасности. Например, если пользователь обладает правом использования сайтов А и В, то он должен иметь возможность работать с сайтами А и В вместе, не требуя, чтобы администраторы этих сайтов взаимодействовали по данному поводу.

Средства безопасности грид также должны обеспечивать гибкую поддержку коммуникационной защиты (осуществляя, например, контроль степени защищённости, независимую защиту данных для ненадёжных протоколов, поддержку надёжности иных, чем TCP протоколов) и предоставлять владельцу ресурса (stakeholder) средства контроля решений, касающихся авторизации, включая различные механизмы ограничения при делегировании полномочий.

Инструментальный комплекс Globus: Перечисленные выше интернет-протоколы, используются для коммуникации. Протоколы *Инфраструктуры Безопасности грид* (Grid Security Infrastructure - *GSI*) [16,31], базирующейся на технологии открытых ключей, используются для аутентификации, коммуникационной защиты и авторизации. Для решения большинства проблем, перечисленных выше, Инфраструктура GSI надстраивает

и расширяет протоколы **Безопасности Транспортного Уровня** (Transport Layer Security - **TLS**) [26]: в частности, однократной регистрации, делегирования, интеграции с различными механизмами безопасности (включая Kerberos [53]) и опирающиеся на пользователя доверительные отношения. Для идентификации используются сертификаты в формате X-509. Контроль авторизации осуществляется с помощью инструментария, который позволяет владельцам ресурсов через **Обобщённый интерфейс контроля Авторизации и Доступа** (Generic Authorization and Access – **GAA**) интегрировать локальные политики управления. В теперешней версии (v1.1.4) Инструментария богатая поддержка ограниченного делегирования не обеспечена, но была продемонстрирована в прототипах.

4.3 Ресурс: Разделение отдельных ресурсов

Уровень **Ресурсов** (Resource layer) основывается на протоколах коммуникации и аутентификации уровня Связи и определяет протоколы (а также API's и SDK's), обеспечивающие для отдельных ресурсов возможности безопасной инициализации, мониторинга и управления операциями разделения на отдельных ресурсах. Для осуществления доступа и контроля локальных ресурсов процедуры уровня Ресурсов, реализующие эти протоколы, обращаются к функциям уровня Фабрикатов. Протоколы уровня Ресурсов касаются исключительно отдельных ресурсов и поэтому игнорируют вопросы глобального состояния и неделимых операций, применяемых к множеству распределённых ресурсов; такого рода проблемы касаются обсуждаемого ниже уровня Кооперации.

Можно выделить два основных класса протоколов уровня Ресурсов:

Информационные протоколы, используемые для получения сведений о структуре и состоянии ресурса, например, о его конфигурации, текущей загрузке и политики использования (например, в плане стоимости).

Протоколы управления используются для согласования доступа к разделяемому ресурсу, определяя, например, требования к ресурсу (включая, предварительное резервирование и качество обслуживания) и операцию(и), которые должны выполняться, например, такие как создание процесса или доступа к данным. Поскольку протоколы управления отвечают за выполнение отношений разделения, они должны служить “точкой применения политики”, гарантируя, что требуемые протокольные операции согласуются с политикой, в соответствии с которой должно происходить разделение ресурса. Вопросы, которые должны рассматриваться на данном уровне включают учёт использования ресурсов и их оплату. Протокол также может осуществлять и мониторинг статуса и выполнения (например, не завершилась ли) операции.

Несмотря на то, что может быть создано много таких протоколов, надо учесть, что протоколы уровней Ресурсов (и Кооперации) определяют горло нашей модели песочных часов, поэтому следует ограничиться небольшим и целенаправленным набором протоколов. Эти протоколы должны быть выбраны так, чтобы охватить фундаментальные механизмы разделения, действенные для многих различных типов ресурсов (например, различных систем управления локальными ресурсами) и в то же время не ограничивать типы или производительность более высоких протоколов, которые могут быть разработаны.

Перечень желательных функциональных возможностей на уровне Фабрикатов, приведенный в разделе 4.1, указывает основные средства, необходимые на уровне Ресурсов. К этому перечню мы добавим требование “только однократного” определения семантики для многих операций, с надёжным контролем ошибок, фиксирующим когда происходит сбой.

Инструментальный комплекс Globus: Принят небольшой набор, в основном состоящий из стандартизированных протоколов. В частности он включает:

- *грид-Протокол* (поддержки) *Информации о Ресурсах* (Grid Resource Information Protocol - **GRIP**) в настоящее время базируется на *Облегчённом Протоколе Доступа* к (сетевым) *Каталогам* (Lightweight Directory Access Protocol – **LDAP**) и используется для указания стандартной информации о ресурсе и соответствующей информационной модели. Действующий совместно с ним *грид-Протокол Регистрации Ресурсов* (Grid Resource Registration Protocol - **GRRP**), используется для регистрации состояния ресурсов в *Серверах грид-Индексной Информации* (Grid Index Information Servers - **GIIS**), обсуждаемых в следующем разделе.
- Базирующийся на протоколе HTTP *грид-Протокол Доступа к Ресурсам и Управления* (Grid Resource Access and Management - **GRAM**) используется для выделения вычислительных ресурсов, а также мониторинга и контроля вычислений на этих ресурсах.
- Расширенная версия *Протокола Передачи Файлов* (File Transfer Protocol - **GridFTP**) является протоколом управления доступа к данным; расширения включают использование протоколов безопасности уровня Связи, частичного доступа к файлам и управления параллелизмом для высокоскоростных передач информации. Протокол FTP принят в качестве базового протокола передачи данных, поскольку его поддерживают сторонние разработчики, а также потому, что его самостоятельное управление и каналы данных облегчают применение изошрённых услуг.
- Протокол LDAP также используется в качестве протокола доступа к каталогам.

Для каждого из этих протоколов Инструментарий Globus определяет клиентскую часть API's и SDK's систем программирования C и Java. Также для каждого протокола серверная часть SDK's и серверы обеспечивают возможность удобной интеграции в грид различных ресурсов (вычислительных, хранения, сетевых). Например, *грид-Информационная Служба Ресурсов* (Grid Resource Information Service - **GRIS**) реализует функциональные возможности серверной части LDAP с помощью внешних вызовов, позволяющих опубликовать произвольной информации о ресурсах. Важным элементом серверной части Инструментария является, так называемый, “привратник” - (“gatekeeper”), по существу, выдающий подтверждённое GSI “удостоверение” идентичности, которое понятно протоколу GRAM и может быть использовано для диспетчеризации различных локальных операций. Прикладные программные интерфейсы *Обобщённой Службы Безопасности* (Generic Security Services - **GSS** [44]) используются для всех операций аутентификации внутри этих серверов и SDKs, облегчающих замену альтернативных служб безопасности на уровне Связи.

4.4 Кооперация: Согласование множества ресурсов

В то время, как уровень Ресурсов нацелен на взаимодействие с одним ресурсом, расположенный выше уровень рассматриваемой архитектуры содержит протоколы и службы, (а также API's и SDK's), которые не ассоциированы с каким - либо одним заданным ресурсом, а скорее являются глобальными по природе и охвату взаимодействий на всём множестве ресурсов. По этой причине мы называем следующий слой архитектуры уровнем **Кооперации** (Collective layer). Поскольку компоненты уровня Кооперации согласно протоколу песочных часов построены на узком “горле” уровней Ресурсов и Связи, они могут реализовать широкое разнообразие режимов, не предъявляя новых требований к разделяемым ресурсам. Например:

- *Службы каталогов (directory services)* позволяют членам ВО обнаруживать зарегистрированные обобществлённые и/или частные ресурсы. Служба каталогов даёт возможность её пользователям запрашивать ресурсы, указывая их имена и/или атрибуты такие как тип, доступность или загруженность. Для создания каталогов используются протоколы GRRP и GRIP уровня Ресурсов.
- *Службы совместного одновременного распределения, планирования и заказа ресурсов (co-allocation, scheduling and brokering services)* позволяют участникам ВО запрашивать один или больше ресурсов для конкретной цели и планировать решение задач на соответствующих ресурсах. В качестве примеров можно указать системы AppLes [12,13], Condor-G, Nimrod-G [2] и брокер DRM [10].
- *Службы мониторинга и диагностики (monitoring and diagnostics services)* осуществляют мониторинг ресурсов ВО в для выявления отказов, вторжения (“intrusion detection”), перегрузки и т.д.
- *Службы репликации данных (data replication services)* поддерживают управление ресурсами хранения ВО (также возможно, сетевыми и вычислительными) для достижения максимальной производительности при доступе к данным с учётом, например, таких показателей, как время ответа, надёжность и стоимость [4, 40].
- *Системы программирования для grid (grid-enabled programming systems)* дают возможность использовать в грид-среде известные модели программирования. При этом для решения вопросов, касающихся обнаружения и распределения ресурсов, безопасности и других используются различные грид-службы. Например, грид-реализации **Интерфейса Передачи Данных** (Message Passing Interface - **MPI** [29, 34]) и интегрированных сред разработки (manager-worker frameworks) [20, 37].
- *Системы управления заданиями (workload management systems) и инфраструктура взаимодействия (collaboration frameworks)* – также известные, как среды решения задач (problem solving environments – “PSEs”) - предназначены для описания, использования и управления многоэтапными, асинхронными и многокомпонентными потоками работ.
- *Службы обнаружения программного обеспечения (software discovery services)*, используя параметры решаемой задачи находят и выбирают [19] наилучшие для

этой задачи реализацию программного обеспечения и исполнительную платформу. Примерами могут служить системы NetSolve [18] и Ninf [50].

- *Серверы групповой авторизации (community authorization servers)* проводят в жизнь политики группового доступа к ресурсам, выдавая мандаты, которые члены группы могут использовать для доступа к групповым ресурсам. Используя информацию о ресурсах и протоколы управления ресурсами (на уровне Ресурсов) и безопасности, эти серверы обеспечивают глобальную, жёсткую политику обслуживания. Akenti [55] решает некоторые из этих проблем.
- *Службы учёта использования и оплаты ресурсов (community accounting and payment services)* собирают вместе информацию о ресурсах для учёта их использования, платежах за ресурсы и/или ограничения на использование ресурсов членами группы.
- *Службы взаимодействия (collaboration services)* поддерживают согласованный синхронный или асинхронный обмен информацией внутри потенциально больших групп пользователей. Примерами могут служить разработки CAVERNsoft [25, 43], Access Grid [22], а также коммерческие системы программного обеспечения.

Эти примеры иллюстрируют широкое разнообразие протоколов уровня Кооперации и служб, применяемых на практике. Заметим, что в то время, как протоколы уровня Ресурсов по своей природе должны быть общими и широко распространёнными, протоколы уровня Кооперации охватывают спектр от решений общего характера до исключительно специальных или зависящих от конкретной предметной области, причём последние, возможно, интересны только внутри конкретных ВО.

Набор функций может быть реализован либо в виде постоянно существующих служб соответствующими протоколами, либо в виде SDK's (с ассоциированными API's), спроектированными для использования в приложениях. В обоих случаях, их реализация может быть построена на протоколах уровня Ресурсов (или протоколах уровня Кооперации) и API's. Например, на Рисунке 3 показано совместное размещение API и SDK на уровне Кооперации (средний ряд), которое использует протокол управления уровня Ресурсов для операций с нижележащими ресурсами. Выше этого мы определяем протокол совместного резервирования и реализуем службу совместного резервирования, которая, обращаясь к API совместного размещения, применяет этот протокол для выполнения операций совместного размещения и, возможно, обеспечивая дополнительную функциональность, например, авторизацию, нечувствительность к сбоям и регистрацию. После этого приложение может использовать протокол службы авторизации для запроса сквозного сетевого резервирования.

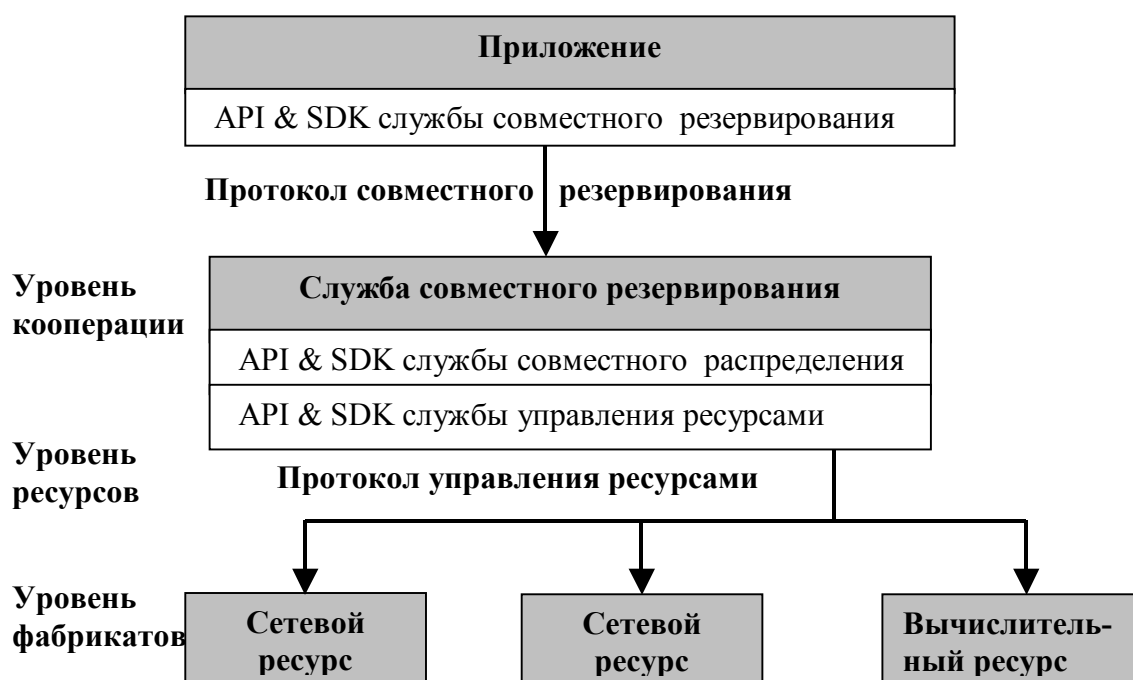


Рисунок 3: Для обеспечения приложений необходимыми функциональными возможностями протоколы уровней Кооперации и Ресурсов, службы, API и SDK могут быть объединены различными способами

Компоненты Кооперации могут быть разработаны по заказу определённой группы пользователей, ВО или конкретного приложения. Например, в виде SDK, который реализует связанный со спецификой приложения протокол, или службы совместного резервирования для заданного комплекса сетевых ресурсов. Другие компоненты Кооперации могут быть более общецелевыми, например, служба репликации, которая управляет международным комплексом систем хранения для многих групп, или служба каталогов, сконструированная для обнаружения других ВО. Вообще, чем масштабней цель группы пользователей, тем важнее, чтобы протоколы компонент уровня Кооперации и API’s базировались на стандартах.

Инструментальный комплекс Globus: В дополнение к представленным ранее в этом разделе примеров служб, многие из которых построены на Globus-протоколах уровней Кооперации и Ресурсов, мы упомянем *Службу Мета Каталогов* (Meta Directory Service – *MDS*), которая вводит *грид-Серверы Индексов Информации* (Grid Information Index Servers - *GIISs*), обеспечивающие возможность рассмотрения подмножества ресурсов с различных точек зрения, информационный протокол LDAP, используемый для доступа к ресурсным GRIS’s при получении сведений о состоянии ресурса, и GRRP, применяемый при регистрации ресурса. Для поддержки управления в среде грид используются каталоги репликаций и служб управления [4]. Служба интерактивного репозитория (“MyProxy”) обеспечивает безопасность хранения для программных агентов, действующих от имени пользователя. Библиотека совместного распределения DUROC обеспечивает SDK и API для совместного распределения ресурсов [24].

4.5 Приложения

Последний уровень обсуждаемой грид-архитектуры – уровень **Приложений** (Applications layer) - содержит пользовательские программные приложения, которые применяются в среде ВО. Рисунок 4 иллюстрирует, как грид-архитектура представляется с точки зрения прикладного программиста. Приложения конструируются в терминах обращений к службам, определённым на любом уровне архитектуры. На каждом уровне чётко определены протоколы, которые обеспечивают доступ к нескольким полезным службам: управлению ресурсами, доступа к данным, обнаружения ресурсов и так далее. Также на каждом уровне для интерфейсов API's может быть установлено, какая реализация (идеально, обеспеченная инструментариями SDK's от сторонних фирм) протокола обмена информацией посылает соответствующим службам сообщения для выполнения желаемых действий.

Необходимо подчеркнуть, что “Приложения”, представленные на Рисунке 4 на одном уровне, на практике обращаются к очень сложным средам и библиотекам

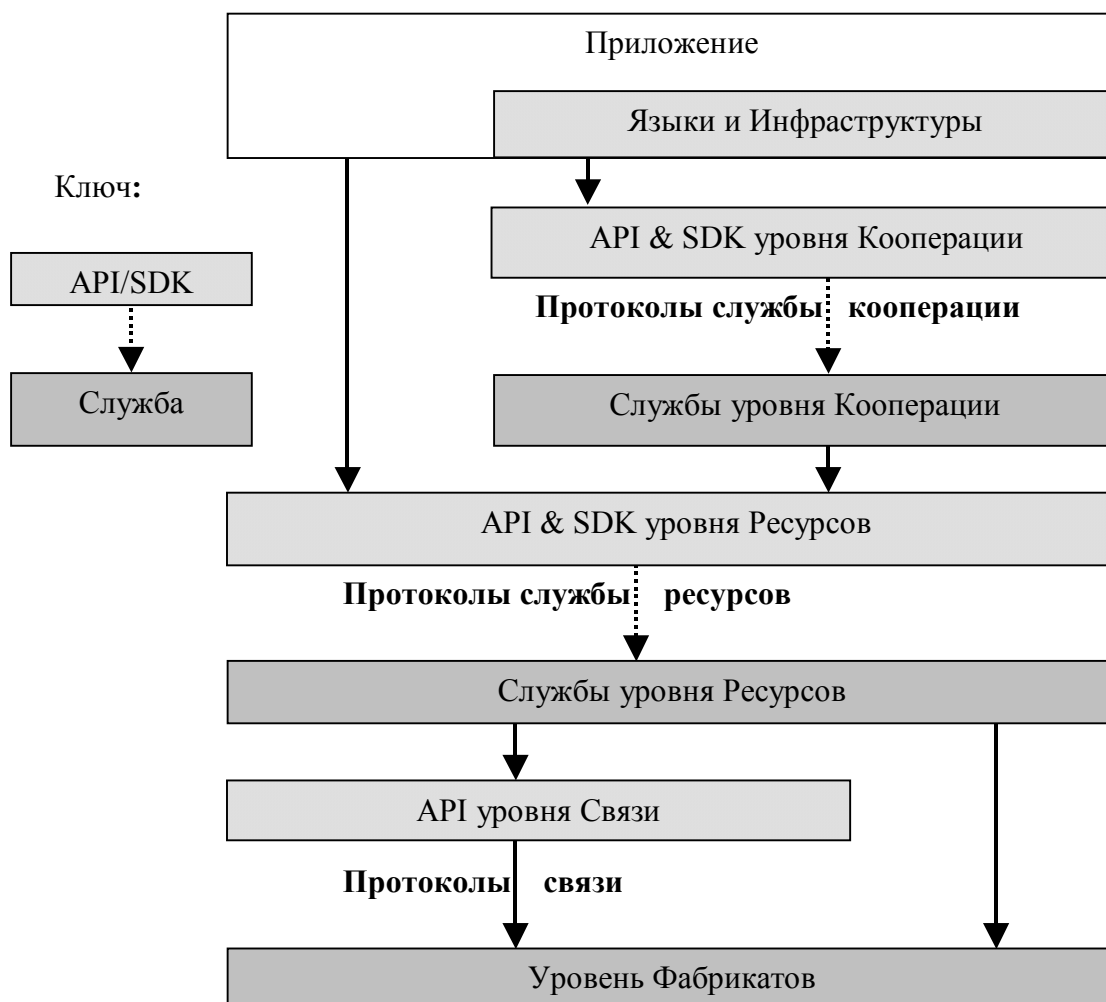


Рисунок 4: Прикладные программные интерфейсы (API's) разработаны с помощью инструментов для разработки программного обеспечения (SDK's), которые в свою очередь используют грид-протоколы для взаимодействия с сетевыми службами, предоставляющими функциональные возможности конечному пользователю. SDK's более

высокого уровня могут обеспечивать функции, которые непосредственно не отражаются в конкретном протоколе, но могут объединять операции протокола с обращениями к дополнительным API's, а также осуществлять локальную реализацию функций. На рисунке сплошные линии представляют прямые обращения; прерывистые линии – взаимодействия протоколов.

(например, таким, как Common Component Architecture [5], SciRun [19], CORBA [35, 47], Legion [38], Cactus [11], системам управления потоками работ [15] и другим средствам с развитой внутренней структурой, которая будучи показана на нашем рисунке, многократно увеличила бы его размеры. Эти системы в свою очередь могут сами определять протоколы, службы и/или APIs (например, Simple Workflow Access Protocol [15]). Однако такие проблемы выходят за рамки данной статьи, которая посвящена только наиболее фундаментальным протоколам и службам, необходимым в грид-среде.

5 Практика грид-архитектуры

Мы используем два примера, чтобы проиллюстрировать, как на практике функционирует грид-архитектура. Таблица 1 показывает службы, которые могли бы быть использованы для выполнения многоаспектного моделирования и приложений, разделяющих кванты вычислительных ресурсов (см. Рисунок 1). Базовые элементы уровня Фабрикатов в каждом из примеров одни и те же: компьютеры, системы хранения и сети. Кроме того, для обеспечения связи и безопасности каждый ресурс переговаривается со стандартными протоколами уровня Связи и протоколами уровня Ресурсов для наведения справок, выделения памяти и управления. Выше этого, каждое приложение использует некую смесь универсальных и в значительной мере проблемно-ориентированных служб уровня Кооперации.

В случае трассировки лучей мы предполагаем, что приложение рассчитано на использование вычислительной системы, обладающей высокой пропускной способностью [46]. Для того, чтобы в среде ВО управлять выполнением большим количеством почти независимых задач, такая система должна вести информационный учёт наборов активных и ожидающих запуска задач, обнаруживать для каждой задачи местоположение соответствующих ресурсов, организовывать загрузку и использование таких ресурсов, обнаруживать и реагировать на различного рода отказы и так далее.

Таблица 1: грид-службы, использованные при конструировании двух приложений Рисунка 1.

	Многоаспектное моделирование	Трассировка лучей
Кооперация (проблемно-зависимая)	Объединитель решателей, архиватор распределённых данных	Контрольная точка, управление заданиями, восстановление от сбоев, подготовка выполнения заданий
Кооперация (групповая)	Обнаружение ресурсов, брокеры ресурсов, системный мониторинг, авторизация участников, отзыв сертификатов	
Ресурсы	Доступ к вычислениям, доступ к данным, доступ к	

	информации о структуре, состоянии и производительности системы
Связь	Коммуникация (IP), служба имен(DNS), аутентификация, авторизация, делегирование
Фабрики	Системы хранения, компьютеры, сети, репозитории программ, Каталоги

Разработка такой системы в контексте нашей грид-архитектуры использует как зависящие от предметной области службы уровня Кооперации (динамическая контрольная точка, управление пулом задач, преодоление сбоев), так и универсальные службы этого уровня (посредничество, репликация данных для исполняемых и общих входных файлов), а также стандартные протоколы уровней Ресурсов и Связи. Проект Condor-G является первым шагом к достижению этой цели.

В случае приложения, используемого при многоаспектном моделировании, на самом верхнем уровне имеют место проблемы совершенно иного характера. Некоторые прикладные инфраструктуры (например, CORBA, CCA) могут использоваться для конструирования приложения из их различных компонентов. Мы также считаем, что в данном случае необходимы механизмы для обнаружения соответствующих вычислительных ресурсов, резервирования времени на этих ресурсах, для организации загрузки и использования, для обеспечения доступа к удалённому хранилищу данных и так далее. Опять же, и здесь может быть использован ряд проблемно-зависимых служб уровня Кооперации (например, объединитель решателей задач, архиватор распределённых данных), но основные базовые средства будут те же, что и в случае трассировки лучей.

6 “В грид-среде”: Необходимость интергрид-протоколов

Наша грид-архитектура устанавливает требования, предъявляемые к протоколам и интерфейсам API's, которые обеспечивают возможность разделения ресурсов, служб и программ. В других отношениях она не ограничивает технологии, которые могли бы быть использованы для реализации этих протоколов и интерфейсов. Действительно, достаточно легко определить множество конкретизаций ключевых элементов грид-архитектуры. Например, мы можем сконструировать на уровне Связи как протоколы, базирующиеся на возможностях систем Kerberos и PKI, так и, благодаря средствам GSS-API (см. Приложение), получить доступ к этим механизмам безопасности через те же самые API. Тем не менее, грид, сконструированный с применением этих различных протоколов, не будет интероперабельным и не сможет разделять наиболее важные службы, по крайней мере, без шлюзов. В связи с этим обеспечение долговременного успеха грид-компьютинга требует, чтобы мы выбрали и довели до широкого распространения единственный пакет протоколов для уровней Ресурсов и Связи и, в меньшей степени ограничения, для уровня Кооперации. Во многом подобно тому, как базовый пакет интернет-протоколов обеспечивает возможность различным вычислительным сетям взаимодействовать и обмениваться информацией, эти “интергрид”-протоколы (так мы будем их называть) обеспечивают возможность различным организациям взаимодействовать, обмениваться ресурсами или разделять их. Про Ресурсы, которые общаются (переговариваются) с помощью таких протоколов можно сказать, что они находятся “В грид-среде”. Стандартные интерфейсы API's также оказываются чрезвычайно полезными, когда речь идёт о необходимости разделения для грид-программ. Установление этих интергрид-

протоколов и интерфейсов API's выходит за рамки данной статьи, хотя Инструментальный комплекс Globus предоставляет средства, которые в настоящее время имеют определённый успех.

7 Соотношение грид с другими технологиями

Концепция контролируемого, динамического разделения ресурсов между членами ВО настолько фундаментальна, что вполне возможно высказать соображение о том, что технологии подобные грид несомненно должны уже быть широко распространены. Тем не менее, в то время как потребность в таких технологиях действительно широко распространена, на практике в достаточно представительном круге приложений мы встречаемся только с примитивными и неадекватными решениями проблем, связанных с созданием ВО. Короче говоря, современные подходы к распределённой обработке данных не обеспечивают построение общей инфраструктуры, которая удовлетворяла бы требованиям, выдвигаемым ВО. Сами же по себе грид-технологии отличаются тем, что обеспечивают общий подход к разделению ресурсов. В связи с этим можно отметить многочисленные благоприятные возможности применения грид-технологий.

7.1 World Wide Web

Распространённость Web-технологий (например, стандартных протоколов – TCP/IP, HTTP, SOAP и других, а также таких языков как HTML и XML) делает их привлекательными в качестве платформы для конструирования ВО систем и приложений. Тем не менее, в то время как эти технологии великолепно решают вопросы поддержки интерактивного взаимодействия клиентского браузера с Web-сервером, что и является основой организации Web, они не вполне достаточны для реализации тех весьма разнообразных моделей интерактивного взаимодействия, которые встречаются в контексте деятельности ВО. Например, сегодняшние Web-браузеры обычно используют для аутентификации пользователя протокол TLS, но не поддерживают функции однократной регистрации или делегирования.

Для интеграции грид с Web-технологиями можно предложить следующие совершенно очевидные решения. Например, включение в Web-браузеры обеспечиваемых GSI средств однократной регистрации для протокола TLS, позволило бы осуществлять однократную регистрацию на множестве Web-серверов. Возможности делегирования, предусмотренные в GSI, позволили бы браузеру клиента делегировать полномочия Web-серверу так, чтобы этот сервер в дальнейшем мог действовать от имени клиента. В свою очередь такие решения существенно упростят использование Web-технологий при создании “ВО- порталов”, которые обеспечат тонким клиентам интерфейсы со сложными ВО приложениями. WebOS предусматривает решение некоторых из этих проблем [57].

7.2 Провайдеры услуг хранения и приложений

Провайдеры прикладных услуг, провайдеры услуг хранения и подобные им обслуживающие компании обычно предлагают использовать специфические, деловые и инженерные приложения из имеющихся в их распоряжении источников (например, в случае обращения к ASP провайдерам) и внешние хранилища информации (при обращении к SSP провайдерам). В этих случаях заказчик подписывает договор на обслуживание, в котором определяется доступ к специальной конфигурации аппаратно-програмных ресурсов. Безопасность при этом должна быть обеспечена путём

использования технологий *Виртуальных Частных Сетей* (Virtual Private Network – *VPN*), позволяющей корпоративной сети (intranet) заказчика охватить ресурсы управляемые системами ASP или SSP от имени заказчика. Другие SSP системы предлагают услуги разделения файлов, и в таких случаях доступ обеспечивается средствами HTTP, FTP, или WebDAV с применением пользовательских идентификаторов и паспортов, также списков контроля доступом, управляющих доступом к ресурсам.

С позиции перспективы развития ВО эти технологии представляются низкоуровневым инструментарием разработчика. VPN и статическое конфигурирование весьма затрудняют достижение многих возможностей разделения, присущих ВО. Например, использование VPN приводит к тому, что обычно ASP-приложение не имеет возможности получить доступ к данным, расположенным в хранилище, управляемом отдельной SSP-системой. Сходная ситуация имеет место и при попытке динамического реконфигурирования ресурсов внутри обособленной ASP- или SSP-системы, что фактически удаётся осуществить крайне редко. Разделение загрузки между несколькими провайдерами, что является обычным решением в индустрии энергоснабжения, пока неизвестно в сфере предоставления услуг провайдерами. Основная проблема здесь состоит в том, что VPN не является ВО: она не может быть расширена динамически с тем, чтобы охватить другие ресурсы и не обеспечивает удалённого провайдера ресурсов какими-либо средствами управления разделением её ресурсов.

Интеграция грид-технологий в среду ASP- и SSP-систем позволит существенно расширить набор возможностей этих систем. В частности, стандартные грид-службы и грид-протоколы могут быть использованы для того, чтобы достигнуть разделения аппаратных и программных ресурсов. Заказчик мог бы при этом договориться о предоставлении ему на основе SLA конкретных аппаратных ресурсов и затем использовать ресурсные протоколы грид для того, чтобы динамически выделять те аппаратные ресурсы, которые необходимы именно для выполнения того или иного конкретного приложения. Механизмы гибкого делегирования и управления доступом к ресурсам позволили бы заказчику напрямую и эффективно осуществлять прогон его приложений на компьютере ASP и безопасный доступ к данным через SSP-систему, а также объединять ресурсы из множества ASP- или SSP-систем со своими собственными ресурсами, когда это диктуется сложностью решаемой задачи. Инфраструктура однократной регистрации позволит динамически связывать множество доменов безопасности, если это действительно необходимо, для обеспечения сложных приложений. Чрезвычайно важны грид-механизмы управления ресурсами и протоколы, осуществляющие учёт их использования и оплаты, которые необходимы при динамическом предоставлении и резервировании ресурсов (например таких, как объём памяти, полоса пропускания и пр.).

7.3 Системы корпоративного компьютеринга

Все технологии корпоративных разработок такие, как CORBA, Enterprise Java Beans, Java 2 Enterprise Edition и DCOM, представляют собой системы, сконструированные для создания с их помощью распределённых приложений. Они поддерживают стандарты интерфейсов к ресурсам, механизмы удалённого вызова, службы торговли для выявления ресурсов и поэтому существенно облегчают решение проблемы разделения ресурсов внутри одной организации. Тем не менее, эти механизмы не соответствуют специфическим требованиям ВО, перечисленным выше. Улаживание вопросов, связанных с разделением ресурсов, при этом, как правило, решается в некой

статической форме и, кроме того, ограничивается рамками одной организации. Основным видом интерактивного взаимодействия здесь оказывается общение в паре клиент-сервер, а не координированное использование множества ресурсов.

Эти рассуждения наводят на мысль том, что грид-технологии могут сыграть определённую роль внутри корпоративного компьютеринга. Например, в случае с CORBA, мы могли бы сконструировать такой *Брокер Запросов Объектов* (Object Request Broker - *ORB*), который использует механизмы GSI для разрешения межведомственных проблем безопасности. Мы также могли бы разработать *Адаптер Мобильных Объектов* (Portable Object Adaptor), который взаимодействует с грид-протоколом управления ресурсами относительно доступа к ресурсам, распределённым по всей ВО. Мы смогли бы сконструировать такие базирующиеся на грид службы Именования и Торговли, которые использовали бы протоколы информационного обслуживания грид для опроса источников информации, распределённых между крупными ВО. В каждом случае использование протоколов грид обеспечивает усовершенствование той или иной функции (например, междоменной безопасности), открывает возможность для взаимодействия с другими (non-CORBA) клиентами. Аналогичные рассуждения могут быть проведены относительно Java и Jini. Например, протоколы и средства реализации Jini приспособлены для небольших наборов аппаратуры. Среда “Grid Jini”, использующая грид-протоколы и грид-службы, позволила бы применять абстракции Jini в крупномасштабной, межведомственной среде.

7.4 Интернет и одноранговый компьютеринг.

Одноранговый (peer-to-peer) компьютеринг, (например, применяемый в системах разделения файлов таких, как Napster, Gnutella и Freenet [23]) и интернет-компьютеринг (применяемый, например, системами SETI@home, Parabon, и Entropia) являются более общими (чем клиент-сервер) подходами к разделению программных и вычислительных структур, которые упоминали, характеризуя суть ВО. В связи с этим они имеют много общего с грид-технологиями. Фактически, мы видим, что на сегодня техническая составляющая работ в этих подходах практически не пересекается. Одна из причин в том, что разработчики однорангового и интернет-компьютеринга до сих пор всецело сосредоточивались на решениях вертикальной интеграции по принципу “печной трубы” (“stovepipe”), а не на определении общих протоколов, которые допускали бы разделение инфраструктуры и интероперабельность. (Это, конечно, общая характеристика новых рыночных ниш, в которых участники всё ещё надеются сохранить монополию). Другая причина состоит в том, что спектр форм разделения, требуемых в различных приложениях достаточно ограничен, например, разделением файлов без контроля доступа или разделяемыми вычислениями с помощью центрального сервера.

По мере того, как эти приложения становятся всё более сложными, а потребность в интероперабельности всё более очевидной, мы будем свидетелями интенсивной сходимости интересов однорангового, интернет- и грид-компьютеринга [31]. Например, технологии однократной регистрации, делегирования и авторизации могут быть важны, когда потребуется организовать взаимодействие служб разделения вычислений и данных, а политики управления доступом к индивидуальным ресурсам станут более искушёнными.

8 Другие перспективы грид

Представленная в данной статье перспектива грид и ВО, разумеется, не является единственно приемлемой. Ниже мы кратко рассмотрим (с критическими репликами) несколько альтернативных перспектив (указаны курсивом).

Грид – это следующая генерация Интернет. Грид не является альтернативой интернет: это скорее всего набор дополнительных протоколов и служб, построенных на базе протоколов и служб интернет, для поддержки разработки и использования приложений в крупномасштабной среде обработке данных и вычислений. Любой ресурс, который “является частью грид”, также, по определению, “является частью Сети”.

Грид – это источник свободных квантов времени для вычислений. Грид-компьютинг не предполагает неограниченный доступ к ресурсам. Грид-компьютинг – это, прежде всего, управляемое разделение ресурсов. Владельцы ресурсов, как правило, будут стремиться применять такие политики, которые ограничивают доступ в соответствии с принадлежностью к той или группе участников ВО, их кредитоспособностью и т.д. Поэтому учёт использования ресурсов важен, а в грид-архитектуре должно быть предусмотрено встраивание ресурсов и протоколов, обеспечивающих ведение информации о загрузке и стоимости ресурсов, а также соответствующую обработку этой информации, когда принимается решение о предоставлении (или нет) конкретного ресурса.

Грид требует распределённой операционной системы. С этой точки зрения программное обеспечение грид должно предусматривать такие службы операционной системы, устанавливаемые на каждой используемой в ВО системе, которые обеспечивают для грид, те же возможности, какие предоставляет операционная система отдельного компьютера: а именно, прозрачность в отношении размещения, именования, безопасности и т.д. Другими словами, с этой точки зрения программное обеспечение грид рассматривается как средство создания некой виртуальной машины. Тем не менее, мы считаем, что такой взгляд противоречит нашим основным целям: широкому распространению и интероперабельности. Мы убеждены, что подходящая модель – это просто набор интернет-протоколов, который обеспечивает широкий спектр ортогональных служб, направленных на уникальные проблемы, возникающие в сетевой среде. Высокая степень физической и административной гетерогенности, имеющей место в среде грид, означает, что традиционная прозрачность недостижима; с другой стороны, это открывает возможность достижения соглашения о стандартных протоколах. Архитектура, предлагаемая в данной работе, преднамеренно открыта, а не зафиксирована: она определяет компактный и минимальный набор протоколов, который должен понимать ресурс, для того, чтобы быть включённым в грид; вне этого набора он воспринимается только для обеспечения инфраструктуры, внутри которой может быть задано много способов его использования.

Грид требует новых моделей программирования. Программирование в грид-среде порождает проблемы, которые не встречаются при использовании последовательных или параллельных компьютеров, например такие, как множество административных доменов, новые виды отказов и большие разбросы в производительности. Тем не менее, мы убеждены, что эти проблемы не главные и что базовая модель программирования при этом фундаментально не изменяется. Так в одних рабочих контекстах абстракция и инкапсуляция могут уменьшить сложность и повысить надёжность. А в других случаях желательно разрешить возможность конструирования широкого разнообразия высокоуровневых абстракций и не сосредотачиваться на частном подходе. Так, например,

разработчику, который верит, что универсальная модель распределённой разделяемой памяти поможет упростить создание грид-приложения, следует реализовать эту модель в терминах грид-протоколов, расширив или заменив эти протоколы, если они действительно неадекватны поставленной цели. Аналогично, разработчику, который полагает, что все грид-ресурсы должны представляться пользователям в виде объектов, потребуется просто создать “объектно-ориентированный API” в терминах грид-протоколов.

Грид делает ненужными высокопроизводительные компьютеры. Сотни, тысячи и даже миллионы процессоров, которые могут быть объединены внутри ВО, представляют собой источник исключительной вычислительной мощности, если их удаётся использовать в некой удобной форме. Это вовсе не означает, что традиционные высокопроизводительные компьютеры устарели. Для решения многих проблем необходимы сильно связанные компьютерные конфигурации с низкими временами ожидания и высокой пропускной способностью; грид-компьютинг может значительно увеличить, а не уменьшить спрос на такие системы, облегчая доступ к ним.

9 Заключение

Мы провели в этой статье краткое обсуждение грид-проблемы, суть которой мы определяем, как управляемое и скоординированное разделение ресурсов и использование их в динамично масштабируемых виртуальных организациях. Мы также рассмотрели как основные аспекты, так и требования, предъявляемые к грид-архитектуре, выделили важнейшие функции, необходимые для обеспечения разделения ресурсов внутри ВО, и определили ключевые отношения между этими различными функциями. Наконец, мы обсудили некоторые детали того, как грид-технологии соотносятся с другими важными технологиями.

Мы надеемся, что терминологическая база и структура, введённые в этой статье, окажутся полезными для формирующегося грид-сообщества, улучшат понимание нашей проблемы и послужат основой общего языка для описания принимаемых решений. Мы также надеемся, что проведенный нами анализ поможет установить связи между разработчиками грид и сторонниками родственных технологий.

Дискуссия в этой статье также поднимает ряд важных вопросов. Как подобрать соответствующие интергрид-протоколы, обеспечивающие интероперабельность среди грид-систем? Какие службы должны быть резидентными (а не дублироваться каждым приложением) для создания приемлемого грид? И что собой представляют ключевые API's и SDK's, которые должны предоставляться пользователям для ускорения разработки и распространения грид-приложений? У нас есть своё мнение по этим вопросам, но ответы явно требуют дальнейших исследований.

Приложение: Определения

Здесь мы даём определение четырём терминам, которые являются фундаментальными при рассмотрении данной статьи, но часто неправильно понимаются и используются, а именно: протокол, служба, SDK и API.

Протокол. *Протокол* – это набор правил, которые применяются в крайних точках телекоммуникационной системы во время обмена данными. Например:

- интернет-Протокол (IP) определяет процедуру ненадёжной передачи пакетов.
- Протокол Управления Передачей (TCP), основывающийся на (IP) определяет процедуру надёжной передачи данных.
- Протокол Безопасности Транспортного Уровня (TLS) определяет правила, обеспечивающие конфиденциальность и целостность данных, передаваемых между двумя коммуникационными приложениями. Он применяется выше некоего надёжного транспортного протокола, например, такого как TCP.
- Облегчённый протокол доступа к (сетевым) каталогам (LDAP) основывается на TCP и определяет протокол типа запрос-ответ, применяемый для опроса состояния удалённой базы данных.

Важное свойство протоколов состоит в том, что они допускают множество реализаций: для обеспечения коммуникационного процесса необходимо только, чтобы в двух крайних точках применялся один и тот же протокол. Таким образом, стандартные протоколы фундаментальны для обеспечения *интероперабельности* в среде распределённого компьютеринга.

Кроме того, определение протокола мало что говорит о режиме работы объекта, применяющего протокол. Например, определение протокола FTP указывает формат сообщений, используемый при переговорах о передаче файлов, но не поясняет как объект, получающий файлы, должен ими управлять.

Как показывают вышеприведенные примеры, протоколы могут быть определены в терминах других протоколов.

Служба. Служба – это создаваемый в сети объект, который обеспечивает некоторую специфическую возможность, например, передавать файлы, создавать процессы или верифицировать права доступа. Служба определяется в терминах протокола, который применяется для взаимодействия с ней, и реакции, ожидаемой в ответ на различные протокольные сообщения (то есть, “служба = протокол + реакция”). Определение службы может допускать разнообразие реализаций. Например:

- FTP-сервер применяет Протокол Передачи Файлов и поддерживает доступ к набору удалённых файлов в режимах чтения и записи. Одна из реализаций FTP сервера может просто писать и читать с локального диска сервера, в то время как другая может писать и читать из системы массовой памяти, автоматически компрессируя и декомпрессируя файлы во время доступа. С позиции уровня

Фабрикатов реакции этих двух серверов в ответ на запрос доступа (или поисковый запрос) очень различны. С позиции же клиента этой службы реакции неразличимы; хранение в файле и последующий поиск в том же файле будут приводить к одинаковым результатам, независимо от того какая реализация службы используется.

- LDAP-сервер применяет LDAP-протокол и обеспечивает ответы на запросы. Одна из реализаций LDAP-сервера может при ответах на запросы использовать информацию из базы данных, в то время как другая может отвечать на запросы, динамически применяя *Простой Протокол Управления Сетью* (Simple Network Management Protocol – *SNMP*), для создания необходимой информации на лету.

Служба может быть или не быть резидентной, способной выявлять отказы и/или восстанавливаться после определённых сбоев; выполняться в привилегированном режиме и/или быть реализована в виде распределённой конфигурации, обеспечивающей широкую масштабируемость. В тех случаях, когда возможно использование нескольких экземпляров, очень важными оказываются механизмы обнаружения (*discovery*), которые позволяют клиенту выяснить свойства конкретного экземпляра службы.

Заметим также, что можно определить различные службы, применяющие один и тот же протокол. Например, в Инструментальном комплексе Globus как служба репликации каталога, так и информационная служба используют протокол LDAP.

API. Интерфейс прикладного программирования (API) определяет стандартный интерфейс (например, набор обращений к подпрограммам или объектов и вызовов методов в случае объектно-ориентированного API) для активизации определённого комплекта функций. Например:

- API Обобщённой Службы Безопасности (Generic Security Service - GSS [44]) определяет стандартные функции для верификации идентичности участников коммуникационных процессов, шифрования сообщений и т.д.
- API Интерфейса Передачи Сообщений (Message Passing Interface – MPI [39]) определяет для нескольких языков программирования стандартные интерфейсы функций, используемых при передаче данных между процессами в параллельной вычислительной системе.

API может задавать множество языковых привязок или использовать Язык Определения Интерфейсов (Interface Definition Language). Язык может быть традиционным языком программирования таким, как C или Java, или может быть интерфейсом оболочки. В последнем случае API ссылается на особое описание аргументов командной строки для данной программы, входных и выходных потоков программы и статус завершения программы. Обычно API будет устанавливать стандартную дисциплину работы, но может допускать множество реализаций.

Важно понимать отношения между API's и протоколами. Определение протокола ничего не говорит об API's, которые могут быть вызваны внутри программы для генерации протокольных сообщений. Один протокол может иметь много API's; один API может иметь множество реализаций, которые предназначены для разных протоколов.

Короче говоря, стандартные API's обеспечивают переносимость; стандартные протоколы обеспечивают интероперабельность.

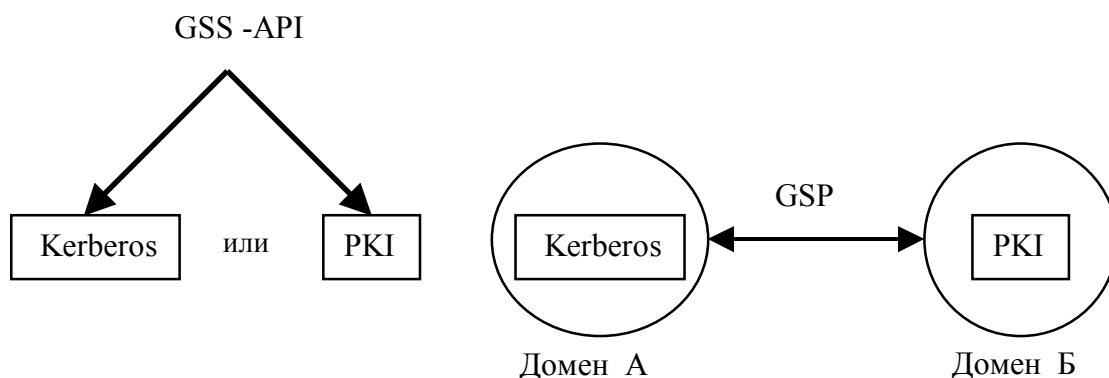


Рисунок 5: В левой части рисунка иллюстрируется использование API для разработки приложений, которые могут применяться либо в среде открытого ключа, либо в среде Kerberos. В правой части рисунка показано использование протоколов (грид-протоколов безопасности, обеспечиваемых Инструментальным комплексом Globus) для создания интероперабельности между средами открытого ключа и Kerberos.

Например, для GSS-API определены как открытые ключи, так и Kerberos связывания. Поэтому программа, которая использует GSS-API вызовы для операций аутентификации, может без изменений работать или в среде с открытым ключом *или* в среде Kerberos. С другой стороны, если мы хотим, чтобы программа выполнялась одновременно и в среде с открытым ключом *и* в среде Kerberos, то нам нужен стандартный протокол, который будет поддерживать интероперабельность этих двух сред. Смотри Рисунок 5.

SDK. Термин инструментарий для разработки программного обеспечения (Software Development Kits – SDK) обозначает комплект автоматически связываемых с приложением или вызываемых из него программ, обеспечивающих реализацию определённых функций. Обычно SDK реализует API. Если API допускает множество реализаций, то будет множество SDK's для этого API. Некоторые SDK's обеспечивают доступ к службам через конкретный протокол. Например:

- Версия OpenLDAP включает в себя SDK LDAP-клиента, в составе которого есть библиотека функций, используемых в C и C++ приложениях для обработки запросов к LDAP-службе.
- Пакет программ JNDI представляет собой Java SDK, содержащий функции, которые могут быть использованы для обработки запросов к LDAP-службе.
- Различные SDK's применяют интерфейс GSS-API, соответственно использующий протоколы TLS и Kerberos.

Может существовать множество SDK's, например, разработанных многими производителями, которые применяют конкретный протокол. Более того, для протоколов, ориентированных на схему клиент-сервер, могут быть реализованы индивидуальные клиентские SDK's для использования приложениями, которые хотят получить доступ к службе, и серверные SDK's для использования разработчиками служб, желающими реализовать специально заказанные режимы работы служб.

SDK не обязательно “говорит” на каком-нибудь протоколе. Например, SDK, который обеспечивает вычислительные функции, может действовать абсолютно локально и не нуждается в применении каких-либо служб при выполнении своих операций.

Библиография

1. Realizing the Information Future: The Internet and Beyond. National Academy Press, 1994. <http://www.nap.edu/readingroom/books/rtif/>.
2. Abramson, D., Sasic, R., Giddy, J. and Hall, B. Nimrod: A Tool for Performing Parameterized Simulations Using Distributed Workstations. In Proc. 4th IEEE Symp. on High Performance Distributed Computing, 1995.
3. Aiken, R., Carey, M., Carpenter, B., Foster, I., Lynch, C., Mambretti, J., Moore, R., Strasner, J. and Teitelbaum, B. Network Policy and Services: A Report of a Workshop on Middleware, IETF, RFC 2768, 2000. <http://www.ietf.org/rfc/rfc2768.txt>.
4. Allcock, B., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S., Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. In Mass Storage Conference, 2001.
5. Armstrong, R., Gannon, D., Geist, A., Keahey, K., Kohn, S.,McInnes, L. and Parker, S. Toward a Common Component Architecture for High Performance Scientific Computing. In Proc. 8th IEEE Symp. on High Performance Distributed Computing, 1999.
6. Arnold, K., O'Sullivan, B., Scheifler, R.W., Waldo, J. and Wollrath, A. The Jini Specification. Addison-Wesley, 1999. See also <http://www.sun.com/jini>.
7. Baker, F. Requirements for IP Version 4 Routers, IETF, RFC 1812, 1995. <http://www.ietf.org/rfc/rfc1812.txt>.
8. Barry, J., Aparicio, M., Durniak, T., Herman, P., Karuturi, J., Woods, C., Gilman, C., Ramnath, R. and Lam, H., NIIP-SMART: An Investigation of Distributed Object Approaches to Support MES Development and Deployment in a Virtual Enterprise. In 2nd Intl Enterprise Distributed Computing Workshop, 1998, IEEE Press.
9. Baru, C., Moore, R., Rajasekar, A. and Wan, M., The SDSC Storage Resource Broker. In Proc. CASCON'98 Conference, 1998.
10. Beiriger, J., Johnson, W., Bivens, H., Humphreys, S. and Rhea, R., Constructing the ASCI Grid. In Proc. 9th IEEE Symposium on High Performance Distributed Computing, 2000, IEEE Press.
11. Benger, W., Foster, I., Novotny, J., Seidel, E., Shalf, J., Smith, W. and Walker, P., Numerical Relativity in a Distributed Environment. In Proc. 9th SIAM Conference on Parallel Processing for Scientific Computing, 1999.
12. Berman, F. High-Performance Schedulers. In Foster, I. and Kesselman, C. eds. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 279-309.
13. Berman, F., Wolski, R., Figueira, S., Schopf, J. and Shao, G. Application-Level Scheduling on Distributed Heterogeneous Networks. In Proc. Supercomputing '96, 1996.

14. Beynon, M., Ferreira, R., Kurc, T., Sussman, A. and Saltz, J., DataCutter: Middleware for Filtering Very Large Scientific Datasets on Archival Storage Systems. In Proc. 8th Goddard Conference on Mass Storage Systems and Technologies/17th IEEE Symposium on Mass Storage Systems, 2000, 119-133.
15. Bolcer, G.A. and Kaiser, G. SWAP: Leveraging the Web To Manage Workflow. IEEE Internet Computing, 85-88. 1999.
16. Brunett, S., Czajkowski, K., Fitzgerald, S., Foster, I., Johnson, A., Kesselman, C., Leigh, J. and Tuecke, S., Application Experiences with the Globus Toolkit. In Proc. 7th IEEE Symp. on High Performance Distributed Computing, 1998, IEEE Press, 81-89.
17. Butler, R., Engert, D., Foster, I., Kesselman, C., Tuecke, S., Volmer, J. and Welch, V. Design and Deployment of a National-Scale Authentication Infrastructure. IEEE Computer, 33(12):60-66. 2000.
18. Camarinha-Matos, L.M., Afsarmanesh, H., Garita, C. and Lima, C. Towards an Architecture for Virtual Enterprises. J. Intelligent Manufacturing.
19. Casanova, H. and Dongarra, J. NetSolve: A Network Server for Solving Computational Science Problems. International Journal of Supercomputer Applications and High Performance Computing, 11(3):212-223. 1997.
20. Casanova, H., Dongarra, J., Johnson, C. and Miller, M. Application-Specific Tools. In Foster, I. and Kesselman, C. eds. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 159-180.
21. Casanova, H., Obertelli, G., Berman, F. and Wolski, R., The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid. In Proc. SC'2000, 2000.
22. Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. J. Network and Computer Applications, 2001.
23. Childers, L., Disz, T., Olson, R., Papka, M.E., Stevens, R. and Udeshi, T. Access Grid: Immersive Group-to-Group Collaborative Visualization. In Proc. 4th International Immersive Projection Technology Workshop, 2000.
24. Clarke, I., Sandberg, O., Wiley, B. and Hong, T.W., Freenet: A Distributed Anonymous Information Storage and Retrieval System. In ICSI Workshop on Design Issues in Anonymity and Unobservability, 1999.
25. Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C. Grid Information Services for Distributed Resource Sharing, 2001.
26. Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W. and Tuecke, S. A Resource Management Architecture for Metacomputing Systems. In The 4th Workshop on Job Scheduling Strategies for Parallel Processing, 1998, 62--82.
27. Czajkowski, K., Foster, I. and Kesselman, C., Co-allocation Services for Computational

Grids. In Proc. 8th IEEE Symposium on High Performance Distributed Computing, 1999, IEEE Press.

28. DeFanti, T. and Stevens, R. Teleimmersion. In Foster, I. and Kesselman, C. eds. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 131-155.

29. Dierks, T. and Allen, C. The TLS Protocol Version 1.0, IETF, RFC 2246, 1999.
<http://www.ietf.org/rfc/rfc2246.txt>.

30. Dinda, P. and O'Hallaron, D., An Evaluation of Linear Models for Host Load Prediction. In Proc. 8th IEEE Symposium on High-Performance Distributed Computing, 1999, IEEE Press.

31. Foster, I. Internet Computing and the Emerging Grid. Nature Web Matters, 2000.
<http://www.nature.com/nature/webmatters/grid/grid.html>.

32. Foster, I. and Karonis, N. A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems. In Proc. SC'98, 1998.

33. Foster, I. and Kesselman, C. The Globus Project: A Status Report. In Proc. Heterogeneous Computing Workshop, IEEE Press, 1998, 4-18.

34. Foster, I. and Kesselman, C. (eds.). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999.

35. Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S. A Security Architecture for Computational Grids. In ACM Conference on Computers and Security, 1998, 83-91.

36. Foster, I., Roy, A. and Sander, V., A Quality of Service Architecture that Combines Resource Reservation and Application Adaptation. In Proc. 8th International Workshop on Quality of Service, 2000.

37. Frey, J., Foster, I., Livny, M., Tannenbaum, T. and Tuecke, S. Condor-G: A Computation Management Agent for Multi-Institutional Grids, University of Wisconsin Madison, 2001.

38. Gabriel, E., Resch, M., Beisel, T. and Keller, R. Distributed Computing in a Heterogeneous Computing Environment. In Proc. EuroPVM/MPI'98, 1998.

39. Gannon, D. and Grimshaw, A. Object-Based Approaches. In Foster, I. and Kesselman, C. eds. The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1999, 205-236.

40. Gasser, M. and McDermott, E., An Architecture for Practical Delegation in a Distributed System. In Proc. 1990 IEEE Symposium on Research in Security and Privacy, 1990, IEEE Press, 20-30.

41. Goux, J.-P., Kulkarni, S., Linderth, J. and Yoder, M., An Enabling Framework for Master-Worker Applications on the Computational Grid. In Proc. 9th IEEE Symp. on High Performance Distributed Computing, 2000, IEEE Press.

42. Grimshaw, A. and Wulf, W., Legion -- A View from 50,000 Feet. In Proc. 5th IEEE Symposium on High Performance Distributed Computing, 1996, IEEE Press, 89-99.
43. Gropp, W., Lusk, E. and Skjellum, A. Using MPI: Portable Parallel Programming with the Message Passing Interface. MIT Press, 1994.
44. Hoschek, W., Jaen-Martinez, J., Samar, A., Stockinger, H. and Stockinger, K., Data Management in an International Data Grid Project. In Proc. 1st IEEE/ACM International Workshop on Grid Computing, 2000, Springer Verlag Press.
45. Howell, J. and Kotz, D., End-to-End Authorization. In Proc. 2000 Symposium on Operating Systems Design and Implementation, 2000, USENIX Association.
46. Johnston, W.E., Gannon, D. and Nitzberg, B., Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. In Proc. 8th IEEE Symposium on High Performance Distributed Computing, 1999, IEEE Press.
47. Leigh, J., Johnson, A. and DeFanti, T.A. CAVERN: A Distributed Architecture for Supporting Scalable Persistence and Interoperability in Collaborative Virtual Environments. *Virtual Reality: Research, Development and Applications*, 2(2):217-237. 1997.
48. Linn, J. Generic Security Service Application Program Interface Version 2, Update 1, IETF, RFC 2743, 2000. <http://www.ietf.org/rfc/rfc2743>.
49. Litzkow, M., Livny, M. and Mutka, M. Condor - A Hunter of Idle Workstations. In Proc. 8th Intl Conf. on Distributed Computing Systems, 1988, 104-111.
50. Livny, M. High-Throughput Resource Management. In Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 311-337.
51. Lopez, I., Follen, G., Gutierrez, R., Foster, I., Ginsburg, B., Larsson, O., S. Martin and Tuecke, S., NPSS on NASA's IPG: Using CORBA and Globus to Coordinate Multidisciplinary Aerospace Applications. In Proc. NASA HPCC/CAS Workshop, NASA Ames Research Center, 2000.
52. Lowekamp, B., Miller, N., Sutherland, D., Gross, T., Steenkiste, P. and Subhlok, J., A Resource Query Interface for Network-Aware Applications. In Proc. 7th IEEE Symposium on High-Performance Distributed Computing, 1998, IEEE Press.
53. Moore, R., Baru, C., Marciano, R., Rajasekar, A. and Wan, M. Data-Intensive Computing. In Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 105-129.
54. Nakada, H., Sato, M. and Sekiguchi, S. Design and Implementations of Ninf: towards a Global Computing Infrastructure. *Future Generation Computing Systems*, 1999.
55. Novotny, J., Tuecke, S. and Welch, V. Initial Experiences with an Online Certificate Repository for the Grid: MyProxy, 2001.

56. Papakhian, M. Comparing Job-Management Systems: The User's Perspective. IEEE Computational Science & Engineering, (April-June) 1998. See also <http://pbs.mrj.com>.
57. Sculley, A. and Woods, W. B2B Exchanges: The Killer Application in the Business-to-Business Internet Revolution. ISI Publications, 2000.
58. Steiner, J., Neuman, B.C. and Schiller, J., Kerberos: An Authentication System for Open Network Systems. In Proc. Usenix Conference, 1988, 191-202.
59. Stevens, R., Woodward, P., DeFanti, T. and Catlett, C. From the I-WAY to the National Technology Grid. Communications of the ACM, 40(11):50-61. 1997.
60. Thompson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K. and Essiari, A. Certificate-based Access Control for Widely Distributed Resources. In Proc. 8th Usenix Security Symposium, 1999.
61. Tierney, B., Johnston, W., Lee, J. and Hoo, G. Performance Analysis in High-Speed Wide Area IP over ATM Networks: Top-to-Bottom End-to-End Monitoring. IEEE Networking, 1996.
62. Vahdat, A., Belani, E., Eastham, P., Yoshikawa, C., Anderson, T., Culler, D. and Dahlin, M. WebOS: Operating System Services For Wide Area Applications. In 7th Symposium on High Performance Distributed Computing, July 1998.
63. Wolski, R. Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service. In Proc. 6th IEEE Symp. on High Performance Distributed Computing, Portland, Oregon, 1997.