ОРДЕНА ЛЕНИНА
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
им. М.В.Келдыша
Российской Академии Наук

М.К.Валиев, Е.Л.Китаев, М.И.Слепенков

Использование службы директорий LDAP для представления метаинформации в глобальных вычислительных системах

Москва, 1999
Настоящая работа поддержана Российским фондом фундаментальных исследований, гранты № 99-01-00374 и № 99-01-00375

Аннотация

В работе обсуждаются вопросы, связанные с представлением и использованием метаинформации, которая играет важную роль при организации высокопроизводительных вычислений в глобальной вычислительной среде. Дается описание основных классов объектов вычислительной среды, информация о состоянии которых должна отображаться в форме метаданных. Вводится понятие информационной службы метакомпьютера (ИСМ) и перечисляются основные операции, которые эта служба должна поддерживать. Описывается схема реализации ИСМ на основе службы директорий LDAP. Рассматриваются вопросы использования ИСМ с целью обеспечения различных схем оптимального запуска приложений и планирования вычислений в метакомпьютерной среде.

Using LDAP directory service for representation of metainformation in global computing systems

Abstract

The paper discusses a wide range of aspects of representing and using metainformation, which plays a significant role in the high-performance wide-area distributed computing. Main classes of computational grid objects are described which should be represented in the form of metadata. The notion of a metacomputing information service (MIS) is introduced and the basic operations are described which must be supported by such a service. An approach to the implementation of MIS is presented which is based on the LDAP directory service. Several aspects of application of MIS are discussed which are connected with the efficient management of computational tasks and planning of computations in the metacomputing environment.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ **3**

1 ОРГАНИЗАЦИЯ МЕТАДАННЫХ В ГЛОБАЛЬНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ GLOBUS **5**

1.1 Схема директории MDS 6
1.2 Организация сбора и актуализации информации в MDS 13

2 ИСПОЛЬЗОВАНИЕ ИСМ ДЛЯ ПОИСКА ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ И ПЛАНИРОВАНИЯ ВЫПОЛНЕНИЯ ЗАДАНИЙ **19**

ЗАКЛЮЧЕНИЕ **25**

СПИСОК ЛИТЕРАТУРЫ **26**

Введение

В ближайшее время в мире ожидается резкое увеличение производительности глобальных сетей на основе новых телекоммуникационных технологий и применения гигабитных каналов связи. В связи с этим становится возможной постановка задачи создания метакомпьютера - глобальной, географически распределенной вычислительной системы, в которой можно будет объединить ресурсы многих в настоящее время изолированных вычислительных центров.

В появлении такой системы заинтересованы как владельцы вычислительных центров, которые смогут более эффективно использовать имеющиеся вычислительные мощности и, соответственно, получить реальные экономические выгоды, так и пользователи системы, которые смогут при реализации своих ресурсоемких приложений аккумулировать мощности нескольких вычислительных центров.

В системе такого масштаба существенную роль приобретает служба информационного обеспечения (информационная служба метакомьютера, ИСМ), в задачи которой входит предоставление оперативного доступа к различного рода служебной информации (метаинформации), описывающей конфигурацию метакомпьютера, состав и состояние его объектов (вычислительных и сетевых ресурсов).

Главными потребителями метаинформации в глобальной вычислительной системе являются приложения, которые используют эту информацию для обнаружения необходимых ресурсов (например, поиска вычислительного сервера с определенной моделью процессора), выбора оптимального ресурса (например, географически ближайшего вычислительного сервера), планирования вычислений и т.п. Доступ к метаинформации может потребоваться и для реализации различных компонент из состава штатного и нештатного программного обеспечения метакомпьютера (например, для

соединения с хостом необходимо по его имени определить соответствующий протокол доступа и сетевой адрес).

Основные группы объектов глобальной вычислительной системы, информация о которых должна отражаться в форме метаданных - это вычислительные центры (сайты), кластеры вычислителей и отдельные вычислители, сетевое оборудование и каналы связи, пользователи системы, приложения и т.д. Метаданные имеют сложную структуру статической и динамической природы. Например, в стандартное описание вычислителя входят:

- количество и модель процессоров, параметры оперативной и дисковой памяти;
- инсталлированная операционная система и программное обеспечение (трансляторы, вычислительные пакеты, библиотеки подпрограмм и т.д.);
- состояние загруженности вычислителя и история изменения этого состояния (текущая загрузка, свободная дисковая и оперативная память и т.д.) ;
- описание правил использования вычислителя, установленных организацией - владельцем компьютера.

Наиболее сложные проблемы при создании ИСМ связаны с высокой степенью распределенности метаинформации в глобальной, географически распределенной вычислительной системе. Поставщиками метаинформации являются вычислительные центры (сайты метакомпьютера), в задачу которых входит предоставление в определенном формате и последующая актуализация информации о тех имеющихся у них вычислительных и сетевых ресурсах, которые выделяются ими в распоряжение пользователей метакомпьютера. Информация может поступать также и из других источников, например, от специализированных узлов, осуществляющих мониторинг сетей. Таким образом, общее число поставщиков метаинформации в подобной системе может исчисляться тысячами и даже десятками тысяч.

Одним из наиболее развитых в настоящее время инструментов, призванных обеспечить создание распределенных информационных систем в глобальной сети Интернет, является служба директорий LDAP (Lightweight Directory Access Protocol) [JBH98]. Систематизированное описание службы директорий LDAP, а также вопросы применения LDAP в качестве инструментального средства были подробно рассмотрены авторами в работе [ВКС2000].

Поскольку информационная служба метакомпьютера принадлежит именно к классу распределенных систем, то кажется естественной идея использования LDAP в качестве инструментального средства для ее реализации. Этот подход впервые был предложен авторами проекта Globus.

Globus [WWWGlo], [FK97], [CFK98] представляет собой реально действующий прототип глобальной метакомпьютерной системы, объединяющий в настоящее время вычислительные ресурсы более 100

организаций из различных стран мира. Globus ориентирован на поддержку высоко-эффективных распределенных вычислений (high-performance distributed computing, HPDC). Одной из ключевых компонент в системе Globus является информационная служба MDS (Metacomputing Directory Service) [FFK97], базирующаяся на службе директорий LDAP.

В настоящей работе дается подробное описание этого подхода и, одновременно, проводится анализ достоинств и недостатков, связанных с применением LDAP для реализации информационной службы метакомпьютера. В процессе исследования основное внимание уделяется следующим вопросам:

- системе классов, с помощью которых осуществляется представление данных о состоянии вычислительных ресурсов метакомпьютера;
- организации процесса сбора, обновления и поиска метаинформации, описывающей доступные вычеслительные и сетевые ресурсы в глобальной вычислительной системе;
- использованию ИСМ при организации распределенных вычислений в метакомпьютерной среде.

Первый раздел настоящей работы посвящен описанию системы MDS. В разделе 1.1подробно рассматривается схема MDS-директории. Эта схема включает набор классов объектов, необходимых для представления вычислительных и сетевых ресурсов метакомпьютерной системы. В разделе 1.2 дается представление об организации процесса сбора и актуализации информации, реализуемого в MDS. Здесь же описывается процедура подключения к MDS нового метакомпьютерного сайта на примере сайта ИПМ.

Во втором разделе работы обсуждаются основные способы использования ИСМ при запуске и выполнении приложений в среде метакомпьютера, начиная с простейшего случая, когда при запуске задания явно указываются вычислительные ресурсы, на которых оно должно быть размещено, и переходя к более сложным вариантам, которые включают поиск ресурсов в метакомпьютере и планирование динамического размещения заданий.

В заключении делается попытка определить круг нерешенных на сегодня проблем, которые требуется решить для создания полноценной ИСМ.

1 Организация метаданных в глобальной вычислительной системе Globus

Система MDS представляет собой реально действующий прототип метакомпьютерной информационной службы, разработанной и эксплуатируемой в рамках проекта Globus. На сегодняшний день в MDS аккумулирована информация о вычислительных ресурсах метакомпьютера, предоставленных более 100 вычислительными центрами из различных стран

мира. К достоинствам MDS следует отнести, что помимо данных о вычислительных ресурсах как таковых (серверах, рабочих станциях, кластерах), в системе также представлена информация о сетевых ресурсах (конфигурация сети и параметры, описывающие ее производительность). Наличие этой информации позволяет существенно оптимизировать распределенные вычисления, поскольку дает возможность учитывать пропускную способность сети при предварительном планировании размещения вычислительных заданий.

В основе реализации MDS лежит служба директорий LDAP. Для представления метаинформации в Globus организована автономная директория (MDS-директория), которая отличается от обычной директории собственной схемой, расширяющей стандартную схему LDAP набором классов, необходимых для представления элементов глобальной вычислительной системы. Схема MDS будет подробно рассмотрена нами в первой части настоящего раздела.

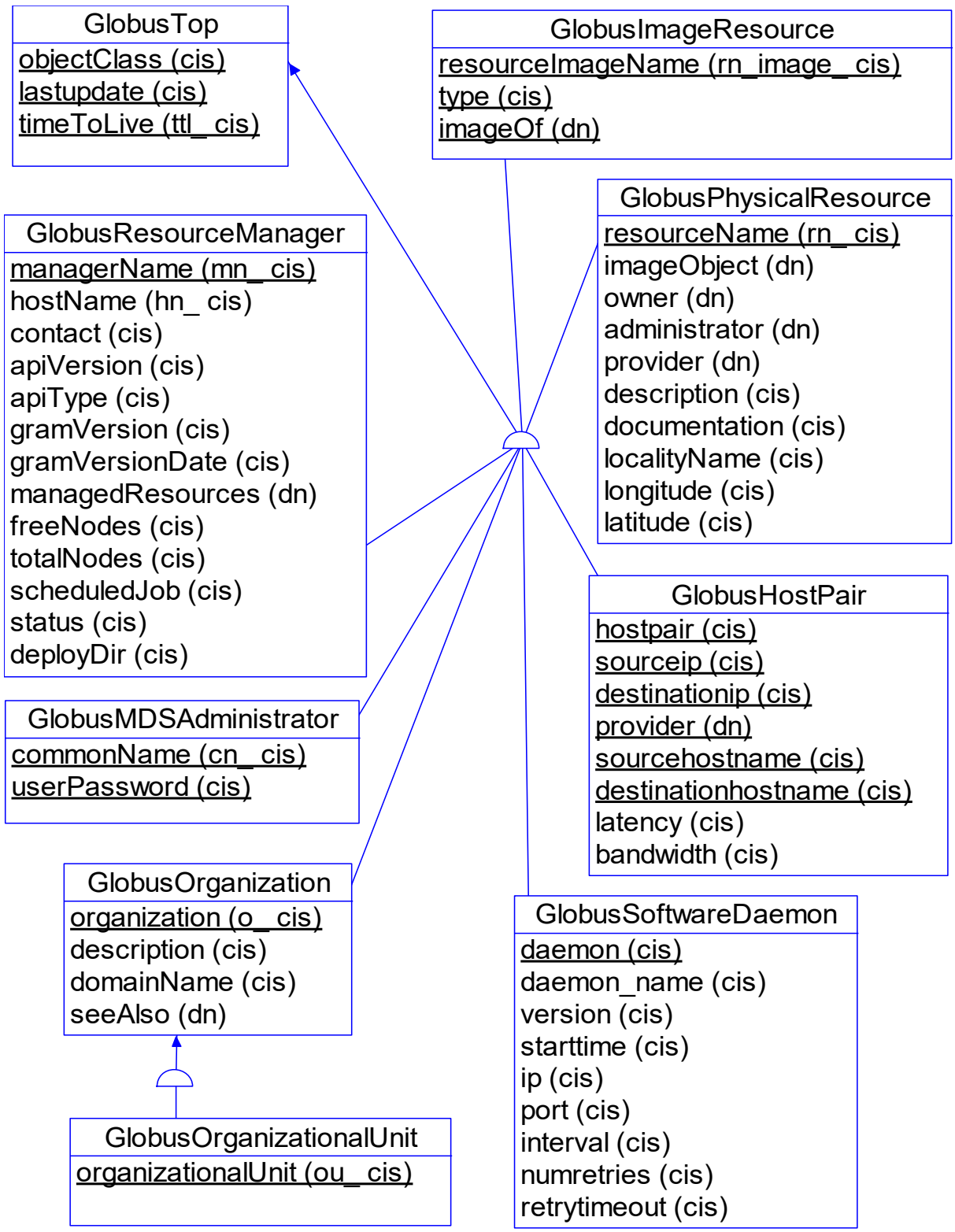
С технической точки зрения любой LDAP-сервер, на котором установлена схема MDS, может использоваться как MDS-сервер. Верхние уровни директории MDS построены по тому же принципу, что и соответствующие уровни LDAP-директории. На первом уровне расположен объект с именем c=US, а на втором - объект o=Globus,c=US. Оба этих объекта являются по сути фиктивными, поскольку не хранят никакой полезной информации. Все реальные объекты, представляющие метаинформацию, подчинены вершине o=Globus,c=US информационного дерева. Такая организация обеспечивает возможность в будущем интегрировать MDS-директорию в глобальную директорию LDAP.

Поскольку система MDS является приложением над LDAP, она наследует все программное обеспеспечение, имеющееся в службе директорий. В частности ее разработчикам не пришлось тратить дополнительные усилия на реализацию операций поиска, модификации данных, аутентификацию и авторизацию пользователей и т.п. Для этих целей в MDS применяются страндартные пользовательские утилиты и API, принятые в LDAP.

Принципиальное значение имеет то, что основной объем информации в MDS собирается и актуализируется в автоматическом режиме. Для этого реализована необходимая инфраструктура, которая базируется на взаимодействии MDS-сервера и комплекса резидентных программ (демонов), которые устанавливаются на вычислителях, подключаемых к системе Globus. Этот аспект системы MDS рассматривается в разделе 1.2, где будет также проиллюстрирована технология подключения к метакомпьютерной системе Globus вычислительных ресурсов новых организаций.

1.1 Схема директории MDS

На представленной ниже схеме приведены основные классы, которые используются в MDS.



Все классы объектов, определяемые в схеме MDS, наследуют атрибуты от абстрактного класса GlobusTop. Класс GlobusTop имеет три обязательных атрибута (обозначенных подчеркиванием): имя класса объекта (objectClass), дата и время последнего обновления объекта (lastupdate) и служебный атрибут, определяющий время жизни объекта (timeToLive, ttl). Два последних атрибута, которые являются обязательными для всех объектов в MDS,

определяют правила *автоматического* удаления объектов из директории. Значение ttl может быть задано либо как constant, тогда на объект не распространяется автоматическое удаление, либо как dd:hh:mm, тогда он будет удален, если величина lastupdate+ttl больше чем текущая дата. Например, если ttl имеет значение 04:00:00, то это означает, что объект будет удален, если он не обновлялся в течение четырех дней.

Для представления организаций и их подразделений используются классы GlobusOrganization и GlobusOrganizationalUnit. Атрибутный состав этих классов аналогичен стандартному, принятому в LDAP для описания организаций (на схеме приведена только часть атрибутов, которые реально используются в MDS). Для представления администраторов MDS используется класс GlobusMDSAdministrator, который содержит атрибуты: общепринятое имя (cn, обычно здесь указывается Directory Manager) и зашифрованный пароль (userPassword).

Подключение вычислительных ресурсов организации к системе Globus требует установки одного или нескольких *менеджеров ресурсов*, которые отвечают за взаимодействие с клиентскими приложениями и выполняют операции резервирования вычислительных ресурсов, запуска заданий и т.п. Существуют менеджеры нескольких типов - штатный менеджер fork (включенный в состав ОС UNIX), а также специализированные менеджеры для различных кластерных систем (pbs, condor и др.). Менеджеры ресурсов представляются в MDS при помощи класса GlobusResourceManager, в котором объявляются следующие атрибуты: имя менеджера (managerName), представляемое строкой в формате DNSимяХоста-ТипМенеджера (например, "grworkst.keldysh.ru-fork" - менеджер с DNS-именем grworkst.keldysh.ru типа fork); имя хоста, на котором установлен менеджер (hostName); адрес, по которому производится соединение с менеджером в формате DNSимяХоста:Порт (contact); ссылки (DN) на обслуживаемые менеджером вычислительные ресурсы (managedResources); общее число обслуживаемых узлов и число свободных узлов (totalNodes, freeNodes); список выполняемых в текущий момент заданий (sheduledJob); а также прочая служебная информация.

В системе Globus реализованы операции мониторинга сети, который осуществляется при помощи демонов, выполняющих периодическое измерение задержки и пропускной способности соединения между парами хостов. Для представления получаемой демонами информации используются объекты класса GlobusHostPair, в которых отражаются: адреса пары хостов IP и DNS (sourceip, destinationip и sourcehostname, destinationhostname); уникальное имя пары, образуемое конкатенацией sourceip и destinationip (hostpair); DN-имя демона, предоставившего информацию (provider), а также полученные демоном результаты измерения (latency - задержка, bandwidth - пропускная способность). В MDS также представляется информация и о самих демонах-измерителях. Для этого используется класс GlobusSoftwareDaemon со следующими атрибутами: имя демона (daemon и daemon_name); время запуска

демона (starttime); IP-адрес и порт, на котором выполняется демон (ip, port); временной интервал, по истечении которого возобновляются измерения (interval); а также настроечная информация (numretries - число попыток соединения, retrytimeout - контрольное время для обрыва соединения).

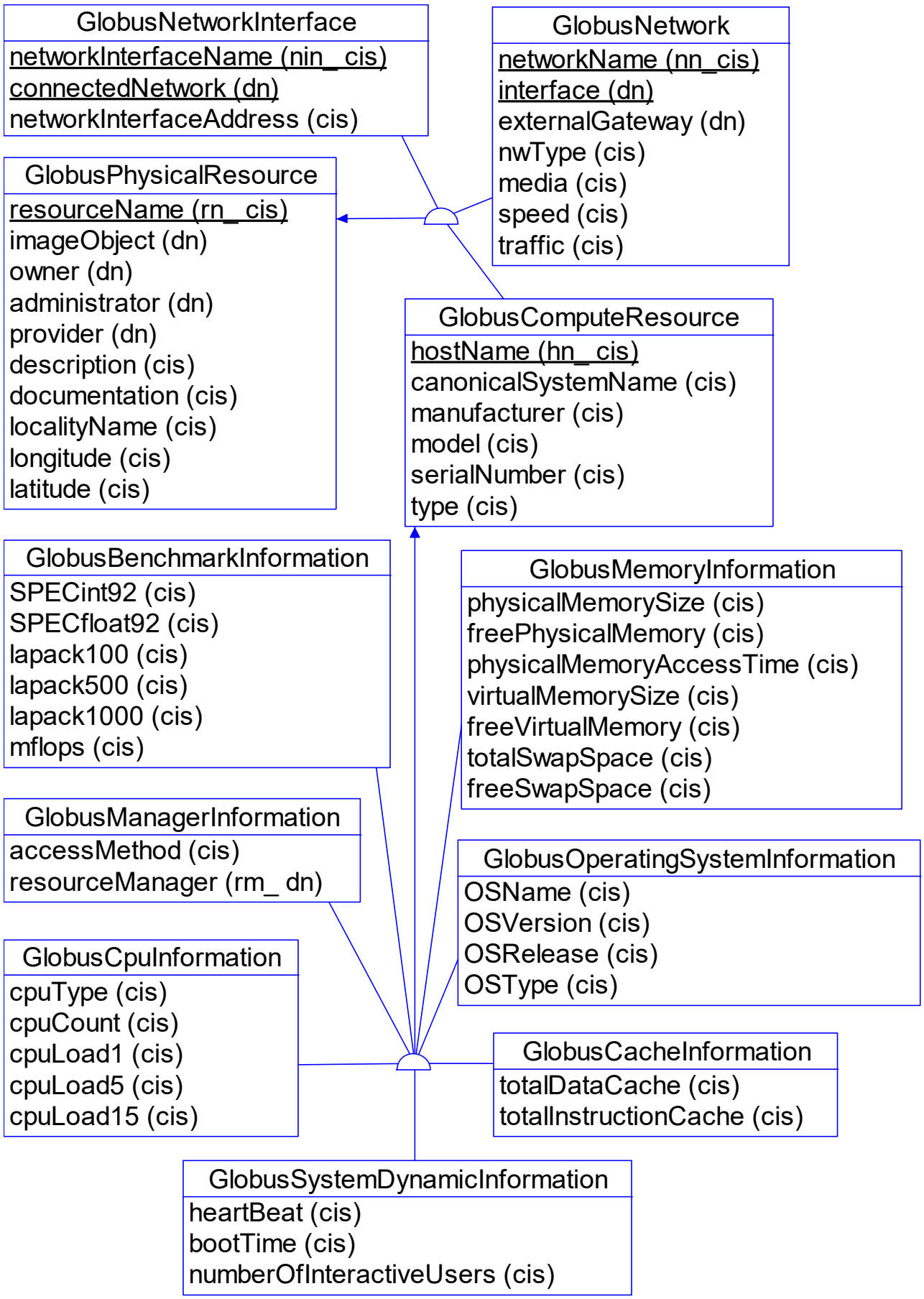
Для представления реальных физических ресурсов в MDS вводится абстрактный класс GlobusPhysicalResource, который является родительским для трех производных классов: GlobusComputeResource - описание вычислительного ресурса, GlobusNetwork - описание сети, GlobusNetworkInterface - описание сетевого интерфейса (сетевой платы) компьютера. В свою очередь от класса GlobusComputeResource порождается набор дочерних классов, описывающих различные аспекты вычислительных ресурсов метакомпьютера, как показано на приведенной ниже диаграмме.

Класс GlobusPhysicalResource имеет следующие атрибуты: имя ресурса (resourceName, например "Host grworkst.keldysh.ru" или "Network 194.226.56.0"); ссылки на организацию-владельца, администратора и провайдера ресурса (owner, administrator, provider); текстовое описание и ссылки на документацию (description, documentation); характеристики географического местоположения ресурса (localityName - штат, город и т.п., longitude - географическая долгота, latitude - широта); а также ссылка на образ ресурса (imageObject). Смысл последнего атрибута будет раскрыт несколько позже.

Класс GlobusComputeResource наследует перечисленные в GlobusPhysicalResource атрибуты и вводит описание хоста с использованием следующих атрибутов: DNS-имя хоста (hostname); каноническое имя системы, которое образуется конкатенацией типа процессора, производителя и версии операционной системы (canonicalSystemName, например, "hppa1.1 hp hpux 10.20"); производитель компьютера (manufacturer, например, hp или sun); модель и серийный номер (model, serialNumber); тип хоста (type, обычно принимает значения workstation или server). Как видно из диаграммы, от класса GlobusComputeResource порождается семь дочерних классов.

Класс GlobusCpuInformation отражает информацию о процессорах, используемых хостом. Этот класс имеет следующие атрибуты: тип процессора (cpuType); количество процессоров (cpuCount); средняя загруженность процессоров за последние 1, 5 и 15 минут (cpuLoad1, cpuLoad5 и cpuLoad15). Характеристика имеющегося у хоста кеша дается в классе GlobusCacheInformation, где отражается общий размер кеша данных и кеша инструкций (totalDataCache и totalInstructionCache).

Характеристика имеющейся оперативной памяти представляется классом GlobusMemoryInformation, где приводится: общий и свободный размеры физической памяти (physicalMemorySize и freePhysicalMemory); время доступа к физической памяти (physicalMemoryAccessTime); общий и свободный размеры виртуальной памяти (virtualMemorySize и freeVirtualMemory);


GlobusNetworkInterface
networkInterfaceName (nin_ cis)
connectedNetwork (dn)
networkInterfaceAddress (cis)
GlobusNetwork
networkName (nn_cis)
interface (dn)
externalGateway (dn)
nwType (cis)
media (cis)
speed (cis)
traffic (cis)
GlobusPhysicalResource
resourceName (rn_ cis)
imageObject (dn)
owner (dn)
administrator (dn)
provider (dn)
description (cis)
documentation (cis)
localityName (cis)
longitude (cis)
latitude (cis)
GlobusComputeResource
hostName (hn_ cis)
canonicalSystemName (cis)
manufacturer (cis)
model (cis)
serialNumber (cis)
type (cis)
GlobusBenchmarkInformation
SPECint92 (cis)
SPECfloat92 (cis)
lapack100 (cis)
lapack500 (cis)
lapack1000 (cis)
mflops (cis)
GlobusMemoryInformation
physicalMemorySize (cis)
freePhysicalMemory (cis)
physicalMemoryAccessTime (cis)
virtualMemorySize (cis)
freeVirtualMemory (cis)
totalSwapSpace (cis)
freeSwapSpace (cis)
GlobusManagerInformation
accessMethod (cis)
resourceManager (rm_ dn)
GlobusOperatingSystemInformation
OSName (cis)
OSVersion (cis)
OSRelease (cis)
OSType (cis)
GlobusCpuInformation
cpuType (cis)
cpuCount (cis)
cpuLoad1 (cis)
cpuLoad5 (cis)
cpuLoad15 (cis)
GlobusCacheInformation
totalDataCache (cis)
totalInstructionCache (cis)
GlobusSystemDynamicInformation
heartBeat (cis)
bootTime (cis)
numberOfInteractiveUsers (cis)

общий и свободный размеры области своппинга (totalSwapSpace и freeSwapSpace).

Для представления динамики изменений состояния вычислителя используется класс GlobusSystemDynamicInformation, в котором отражается: дата и время последней перезагрузки системы (bootTime); дата и время последней проверки, выявившей работоспособность вычислителя (heartBeat); количество пользователей, подключенных в момент последней проверки к вычислителю в интерактивном режиме (numberOfInteractiveUsers).

Характеристики операционной системы, установленной на вычислителе, отражаются в классе GlobusOperatingSystemInformation, где указывается наименование, версия, номер выпуска и тип операционной системы (OSName, OSVersion, OSRelease и OSType). Номинальная производительность вычислителя представляется классом GlobusBenchmarkInformation, где фиксируются характеристики, полученные в результате выполнения стандартных тестов: SPECint92, SPECfloat92, LAPACK, mflops.

Описание вычислительного ресурса завершает класс GlobusManagerInformation, в котором устанавливается ссылка на менеджер ресурсов, обслуживающий данный вычислитель (resourceManager) и, кроме того, здесь можно задать расписание для доступа и резервирования вычислителя (accessMethod).

Как было сказано выше, помимо вычислительных ресурсов в MDS отражается информация о сетевых ресурсах, для чего введены два класса: GlobusNetwork и GlobusNetworkInterface. В классе GlobusNetwork описывается сегмент сети. При этом делается акцент на описание физических характеристик сегмента: идентификатор сегмента (networkName, например, имеющиеся в ИПМ два сегмента можно идентифицировать как 194.226.56.0 и 194.226.57.0); ссылка на вышестоящий внешний сегмент, если такой сегмент представлен в MDS (externalGateway); тип сети (nwType, например, ethernet или ATM); вид кабеля (media, например, cooper или fiber-optic); номинальная пропускная способность сегмента (speed); средняя загруженность сегмента (traffic); ссылки на сетевые интерфейсы компьютеров (объекты класса GlobusNetworkInterface), подключенных к сегменту (interface).

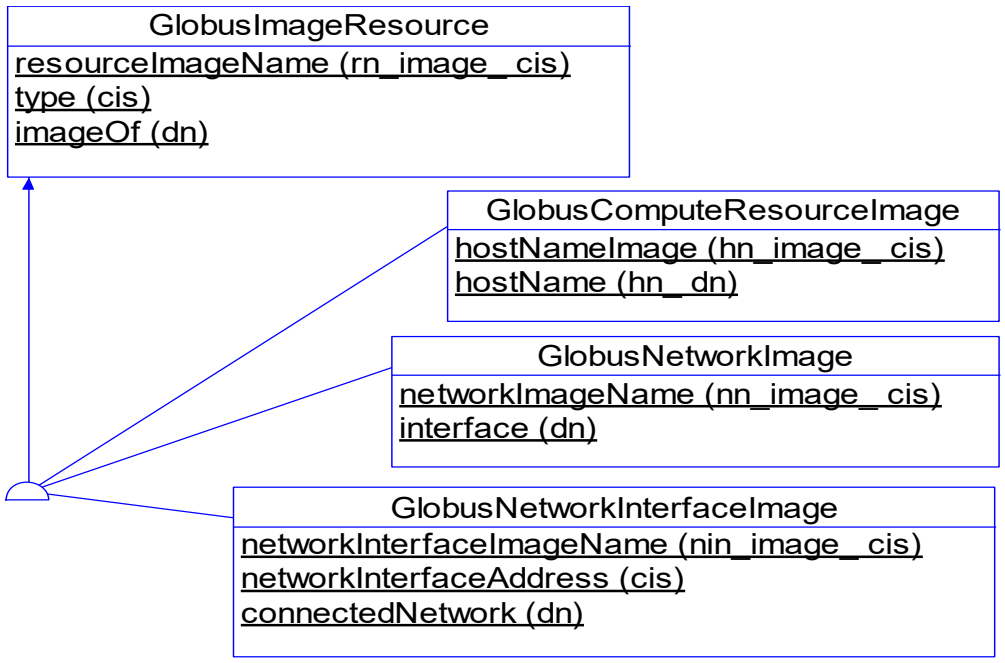
В объектах класса GlobusNetworkInterface отображаются характеристики сетевых плат, через которые компьютеры подключаются в сегмент. Здесь представляется следующая информация: логическое имя порта компьютера, в котором установлена плата (networkInterfaceName, например, lan0 или COM1); ссылка на сегмент сети (connectedNetwork); аппаратный адрес сетевой карты (networkInterfaceAddress, например, в случае Ethernet - 48-битовый адрес карты, устанавливаемый изготовителем).

Помимо физических характеристик сетевых ресурсов (конфигурации и параметров сети), для спецификации которых достаточно описаных выше классов GlobusComputeResource, GlobusNetwork и GlobusNetworkInterface, необходимо также каким-то образом представить информацию логического характера, которая отражала бы какие семейства протоколов (IP, IPX, OSI и

т.п.) поддерживаются в рамках конкретной физической сети. При этом для каждого протокола необходимо зафиксировать систему логических адресов, посредством которых в рамках данного протокола идентифицируются узлы сети.

В принципе, для представления этой информации можно было бы расширить атрибутный состав классов, описывающих физические ресурсы. Например, добавить к классу GlobusNetworkInterface атрибут IPaddress, присутствие значения в котором определяло бы, во-первых, что данный интерфейс поддерживает IP-стек и, во-вторых, само значение сообщало необходимый для соединения IP-адрес. Такое простое решение имеет один принципиальный недостаток. В схеме должен быть перечислен весь спектр используемых сетевых протоколов, причем появление новых неизбежно приводило бы к необходимости расширения схемы, доопределению атрибутного состава классов.

В MDS используется другой, более сложный подход к представлению логической информации о сети, в основе которого лежат идеи, изложенные в [RFC1609]. Физический взгляд на сетевые ресурсы, который реализуется объектами классов GlobusComputeResource, GlobusNetwork и GlobusNetworkInterface, дополняется несколькими логическими взглядами (views). Для представления логических взглядов вводится класс GlobusImageResource (образ физического ресурса), от которого путем наследования порождается три подкласса: GlobusComputeResourceImage, GlobusNetworkImage и GlobusNetworkInterfaceImage, представленные на следующей диаграмме.



Класс GlobusImageResource содержит три обязательных атрибута, которые наследуются всеми подклассами: имя образа ресурса (resourceImageName,

например, "Host Image grworkst.keldysh.ru" или "Network Image 194.226.57.0"); наименование сетевого протокола, который поддерживает данный объект (type, например, IP или IPX); DN-ссылка на соответствующий физический объект, образом которого является данный объект (imageOf).

Класс GlobusComputeResourceImage, используемый для представления образов вычислительных ресурсов, фактически не содержит новых атрибутов, поскольку имя образа вычислителя (hostNameImage, например "Host Image grworkst.keldysh.ru") совпадает с именем образа ресурса, а DN-ссылка на физический хост(hostName) совпадает со ссылкой в imageOf. Класс GlobusNetworkImage, используемый для представления образов сетевых сегментов, обладает атрибутом networkImageName, значение которого совпадает с resourceImageName, а также атрибутом interface, где представляются ссылки на объекты- образы сетевых карт. Класс GlobusNetworkInterfaceImage обладает атрибутом networkInterfaceImageName, значение которого также совпадает с resourceImageName; атрибутом networkInterfaceAddress, который определяет логический адрес узла, соответствующий протоколу; и наконец атрибутом connectedNetwork, который содержит DN-ссылку на соответствующий образ сетевого сегмента.

Из сказанного видно, что между физическими и логическими объектами устанавливаются перекрестные ссылки (атрибуты imageObject у физических объектов и атрибуты imageOf у логических). Описание структуры сети, представленное посредством атрибутов connectedNetwork и interface у объектов NetworkInterface и Network, соответственно, дублируется в объектах-образах. Описанные понятия будут проиллюстрированы в следующей части настоящего раздела.

1.2 Организация сбора и актуализации информации в MDS

В системе Globus предусмотрена достаточно простая технология подключения новых вычислительных ресурсов, к достоинствам которой следует отнести минимизацию трудозатрат административного персонала (администратора центрального MDS-сервера и администраторов, выделенных организациями) на этапе настройки и дальнейшего ведения процедуры сбора метаинформации в MDS-директории. В настоящем разделе мы рассмотрим как производится подключение к Globus ресурсов организации, используя в качестве примера Институт прикладной математики.

На начальном этапе подключения к системе Globus новой организации требуется провести регистрацию, заполнив сооветствующую форму (HTML-форму на Web-сервере [WWWGlo]). В этой форме указываются: наименование организации (полное и краткое), почтовый адрес, телефон, факс, доменное имя (DNS-имя) и пр., а также адрес электронной почты локального администратора, отвечающего в рамках данной организации за подключение ресурсов к системе Globus. Получив такое уведомление от новой организации администратор MDS-сервера производит ее регистрацию в

директории MDS, создавая два новых объекта. Первый из этих объектов представляет информацию об организации и размещается в информационном дереве директории в непосредственном подчинении вершине o=Globus, c=US. Второй объект представляет локального администратора Globus для данной организации и становится дочерней вершиной объекта-организации. Этому объекту приписывается зашифрованный пароль, который используется для входа в директорию и дает право администрировать поддерево, соответствующее данной организации. Приведенные ниже объекты представляют ИПМ и его администратора в MDS после регистрации.

dn: o=KIAM, o=Globus, c=US
objectclass: GlobusOrganization
o: Keldysh Institute for Applied Mathematics
o: KIAM
domainname: keldysh.ru
domainname: kiam.ru
lastupdate: Fri Feb 19 22:20:19 GMT 1999
ttl: constant

dn: cn=Directory Manager, o=KIAM, o=Globus, c=US
objectclass: GlobusMDSAdministrator
cn: Directory Manager
o: Keldysh Institute for Applied Mathematics
o: KIAM
userpassword: {SHA}La2UsBWJSjCcvP5yUUg85G8P3I4=
lastupdate: Fri Feb 19 22:20:19 GMT 1999
ttl: constant

Следующим шагом для подключения к Globus является установка программного обеспечения на компьютере, выделяемом под шлюз (gatekeeper). Именно с этой машины будет в дальнейшем выполняться обновление фрагмента MDS-директории данной организации. Для этого на шлюзовой машине запускается демон обновления MDS, который по истечении определенных интервалов времени отгружает на MDS-сервер обновленное состояние ресурсов данной организации, включенных в метакомпьютер. Отметим, что шлюзовая машина может также использоваться и как вычислитель, доступный пользователям Globus.

Список вычислителей, выделенных организацией в распоряжение метакомпьютерной системы, фиксируется в соответствующем конфигурационном файле, размещенном на шлюзовой машине. Этот файл автоматически модифицируется при подключении новых вычислителей или исключении каких-то из них. При подключении вычислителя на нем устанавливается необходимое программное обеспечение и запускается демон - менеджер ресурса (ResourceManager) соответствующего типа.

Демон обновления MDS, запущенный на шлюзовой машине, выполняет опрос вычислителей из списка, который представлен в конфигурационном файле, и автоматически формирует фрагмент поддерева организации. В этот фрагмент попадают следующие объекты. Для каждого вычислителя создается два объекта. Первый объект, класса GlobusComputeResource, отображает этот вычислитель как физический ресурс, а второй, класса GlobusResourceManager, описывает менеджер ресурса, запущенный на данном вычислителе. Например, рабочая станция grworkst.keldysh.ru представляется в MDS объектами:

dn: hn=grworkst.keldysh.ru, o=KIAM, o=Globus, c=US
objectclass: GlobusComputeResource
objectclass: GlobusOperatingSystemInformation
objectclass: GlobusCacheInformation
objectclass: GlobusMemoryInformation
objectclass: GlobusManagerInformation
objectclass: GlobusCpuInformation
objectclass: GlobusSystemDynamicInformation
objectclass: GlobusBenchmarkInformation
hn: grworkst.keldysh.ru
rn: Host grworkst.keldysh.ru
canonicalsystemname: hppa1.1 hp hpux 10.20
manufacturer: hp
machinehardwarename: hppa1.1
type: workstation
osname: hpux
osversion: hppa1.1 hp hpux 10.20
osrelease: 10.20
ostype: hpux
cpuload1: 3.77
cpuload5: 3.29
cpuload15: 3.40
lastupdate: Fri Apr 2 16:35:02 GMT 1999
ttl: 04:00:00

dn: cn=grworkst.keldysh.ru-fork, o=KIAM, o=Globus, c=US
objectclass: GlobusResourceManager
cn: grworkst.keldysh.ru-fork
mn: grworkst.keldysh.ru-fork
hn: grworkst.keldysh.ru
apiversion: 0.1
apitype: fork
deploydir: /opt/globus
managedresources: hn=grworkst.keldysh.ru,
o=KIAM, o=Globus, c=US

gramversion: 1.15
gramversiondate: 1998/09/1901:13:02
gramsecurity: ssleay
lastupdate: Mon Apr 5 18:48:22 GMT 1999
ttl: 00:01:00

Оба объекта располагаются в информационном дереве в непосредственном подчинении вершины, соответствующей организации (для ИПМ - o=KIAM, o=Globus, c=US). Далее, демон обновления MDS автоматически формирует объекты, отображающие сетевые ресурсы, которые связывают вычислители, перечисленные в конфигурационном файле. В представленном ниже примере приведены объекты, которые были порождены демоном обновления для хоста grworkst.keldysh.ru.

dn: nn=194.226.57.0, o=KIAM, o=Globus, c=US
objectclass: GlobusNetwork
nn: 194.226.57.0
rn: Network 194.226.57.0
imageobject: nn_image=194.226.57.0, o=KIAM, o=Globus, c=US
interface: nin=lan0, hn=grworkst.keldysh.ru,
o=KIAM, o=Globus, c=US
lastupdate: Mon Apr 5 16:54:42 GMT 1999
ttl: 04:00:00

dn: nin=lan0, hn=grworkst.keldysh.ru, o=KIAM, o=Globus, c=US
objectclass: GlobusNetworkInterface
nin: lan0
rn: Interface lan0
imageobject: nin_image=194.226.57.39,
hn_image=grworkst.keldysh.ru,
nn_image=194.226.57.0,
o=KIAM, o=Globus, c=US
connectednetwork: nn=194.226.57.0, o=KIAM, o=Globus, c=US
type: IP
lastupdate: Mon Apr 5 16:54:36 GMT 1999
ttl: 04:00:00

dn: nn_image=194.226.57.0, o=KIAM, o=Globus, c=US
objectclass: GlobusNetworkImage
nn_image: 194.226.57.0
rn_image: Network Image 194.226.57.0
imageof: nn=194.226.57.0, o=KIAM, o=Globus, c=US
type: IP
lastupdate: Mon Apr 5 16:54:44 GMT 1999

ttl: 04:00:00

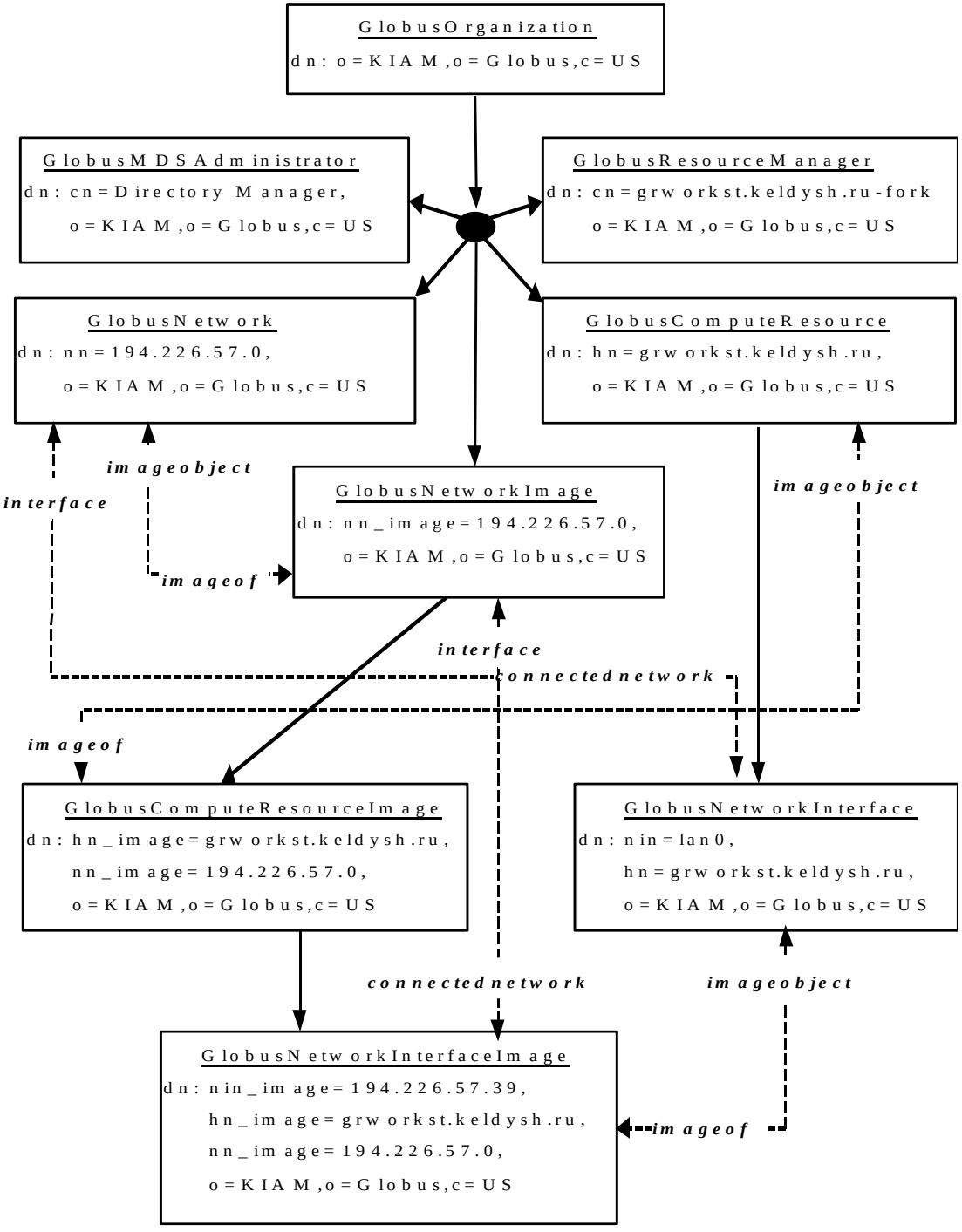
dn: hn_image=grworkst.keldysh.ru, nn_image=194.226.57.0,
o=KIAM, o=Globus, c=US
objectclass: GlobusComputeResourceImage
hn_image: grworkst.keldysh.ru
rn_image: Host Image grworkst.keldysh.ru
hn: grworkst.keldysh.ru
imageof: hn=grworkst.keldysh.ru, o=KIAM, o=Globus, c=US
lastupdate: Mon Apr 5 16:54:33 GMT 1999
ttl: 04:00:00

dn: nin_image=194.226.57.39, hn_image=grworkst.keldysh.ru,
nn_image=194.226.57.0, o=KIAM, o=Globus, c=US
objectclass: GlobusNetworkInterfaceImage
nin_image: 194.226.57.39
rn_image: Interface Image 194.226.57.39
type: IP
imageof: nin=lan0, hn=grworkst.keldysh.ru,
o=KIAM, o=Globus, c=US
networkinterfaceaddress: 194.226.57.39
connectednetwork: nn_image=194.226.57.0,
o=KIAM, o=Globus, c=US
lastupdate: Mon Apr 5 16:54:39 GMT 1999
ttl: 04:00:00

На следующей диаграмме представлена структура фрагмента информационного дерева директории, соответствующего организации ИПМ, в том простом случае, когда к метакомпьютерной системе Globus подключена одна рабочая станция. Из диаграммы легко представить, каким будет результат подключения новых вычислителей ИПМ, если учесть, что подчиненность объектов соответствующих классов, отражаемая в их DN-именах, является предопределенной в MDS.

Как видно из схемы, в непосредственном подчинении вершины, представляющей организацию в информационном дереве, располагаются объекты, отображающие: вычислители (класса GlobusComputeResource), менеджеры ресурсов, установленные на этих вычислителях (GlobusResourceManager), сегменты сети (GlobusNetwork) и образы сегментов сети (GlobusNetworkImage). В подчинении объектов-вычислителей (класса GlobusComputeResource) располагаются объекты, отображающие сетевые карты, установленные в портах вычислителя (класс GlobusNetworkInterface). Последние связываются с объектами-сегментами сети посредством атрибута connectedNetwork (обратная связь устанавливается атрибутом interface). В объекты-образы сетевых сегментов (класса GlobusNetworkImage)

вкладываются образы вычислителей (класса GlobusComputeResourceImage), которым, в свою очередь, подчиняются образы сетевых соединений (класса GlobusNetworkInterfaceImage). Каждый объект-образ связывается с соответствующим объектом, представляющим физический ресурс, при помощи атрибута imageOf (обратная связь устанавливается атрибутом imageObject).



Приведенный пример показывает содержание фрагмента MDS-директории, который формируется автоматически на этапе установки системы Globus. Содержание этого фрагмента может быть далее расширено полезной

информацией, за счет установки дополнительных компонент Globus. Например, на grworkst.keldysh.ru может быть установлен демон, выполняющий тестирование номинальной производительности и определение текущего состояния (загруженности и работоспособности) компьютера, результаты которых дополнят описание соответствующего объекта-вычислителя с помощью означивания атрибутов классов GlobusBenchmarkInformation и GlobusSystemDynamicInformation. Кроме того, можно настроить демоны, выполняющие мониторинг сети, указывая те хосты, соединение с которыми должно измеряться. В результате этого в фрагменте директории появляются объекты класса GlobusHostPair, сообщающие полученную метрическую информацию. Объекты класса GlobusHostPair располагаются в информационном дереве на том же уровне, что и объекты-вычислители.

Наконец, некоторые параметры, которые в принципе не могут быть определены автоматически (например, вид кабеля, задаваемый атрибутом media в классе GlobusNetwork), можно указать вручную. Для этого в системе имеются специальные текстовые файлы-шаблоны, редактирование которых входит в обязанности администратора сайта.

2 Использование ИСМ для поиска вычислительных ресурсов и планирования выполнения заданий

Основными потребителями метаинформации в глобальной вычислительной системе являются приложения, которые используют эти данные для обнаружения необходимых ресурсов, выбора оптимальных ресурсов, планирования вычислений и т.п. В метакомпьютерной среде необходимо поддерживать разнообразные схемы запуска и выполнения приложений, отвечающие различиям в структуре приложений и определенному для них регламенту выполнения, например: приложение может состоять из нескольких заданий, которые требуется разместить на конкретном вычислителе; либо на нескольких вычислителях, но в рамках одного сайта; либо допускается вариант размещения на вычислителях из нескольких сайтов.

В настоящем разделе дается представление об основных способах использования ИСМ для различных схем запуска и выполнения приложений, начиная с простейшего случая, когда при запуске задания явно указываются вычислительные ресурсы, на которых оно должно быть размещено, и переходя к более сложными вариантам, которые включают поиск ресурсов в метакомпьютере (с учетом конфигурации и параметров сети или без их учета) и динамическое размещение заданий. Те аспекты выполнения заданий, которые непосредственно не касаются ИСМ (различные политики планирования, устройство очередей, взаимодействие глобальной среды с локальными кластерами и т.д.) в настоящей работе не обсуждаются. Обзор

многих методов планирования вычислений для метакомпьютерной среды содержится в [Ber98].

Основное применение ИСМ - это планирование распределенных вычислений. При этом планирование нужно понимать в более широком смысле чем обычное планирование на многопроцессорных компьютерах. А именно, планирование в метакомпьютерной среде включает в себя по крайней мере:

- нахождение свободных ресурсов в сети, удовлетворяющих определенным требованиям (производительность, объем памяти, необходимые операционные системы и другое программное обеспечение и т.д.);
- выяснение и оценка коммуникационных возможностей между этими ресурсами;
- распределение компонент приложений (программ и./или данных) по найденным ресурсам.

Обратим внимание, что каждое из этих действий требует значительной информации о ресурсах и конфигурации сети, которую может предоставить ИСМ.

Хотя наиболее значимый эффект от проименения ИСМ достигается в задачах, связанных с поиском и планированием необходимых ресурсов в сети, однако некоторую пользу из ИСМ можно извлечь и в случае, когда адреса тех ресурсов, которые будут использоваться приложением, заранее известны. В частности, такое применение ИСМ заложено в клиентской программе globusrun системы Globus. Программа globusrun имеет два основных варианта вызова:

globusrun [options] -r resource singleRSLstring
globusrun [options] multipleRSLstring.

Рассмотрим сначала первый вариант. В этом случае формируется *одинарный* запрос на выполнение некоторого задания, параметры которого (исполняемая программа, размещение данных и т.д.) задаются в строке singleRSLstring (на языке описания заданий RSL), используя менеджер ресурсов, задаваемый в параметре resource. При этом значением параметра resource может быть так называемая контактная строка некоторого менеджера ресурсов (содержащая имя менеджера, номер порта и контактное имя менеджера) или просто DN-имя менеджера. В первом случае globusrun напрямую передает задание менеджеру ресурса. Во втором случае дополнительная информация о менеджере, которая необходима для передачи ему задания и должна быть включена в его контактную строку, определяется путем обращения к MDS.

Второй вариант формирует *кратный* запрос на одновременное выполнение нескольких заданий на нескольких компьютерах. В этом случае multipleRSLstring фактически содержит несколько одинарных запросов с указанием для каждого из них того менеджера ресурсов, который должен

исполнить этот запрос. При этом, если в запросе используются DN-имена менеджеров, то программа globusrun сначала определяет контактные строки для них (так же как выше, используя MDS) и после этого передает всю информацию модулю DUROC коаллокирования ресурсов, который дальше занимается размещением заданий по вычислителям и их исполнением.

После выполнения исходного размещения заданий и их запуска происходит регулярное обновление информации о текущем состоянии вычислителей и запущенных на них заданиях в MDS. Планировшик или само приложение могут использовать эту динамически актуализируемую информацию для отслеживания состояния заданий и при необходимости принять решение о снятия задания с перегруженного вычислителя и передачи задания другому вычислителю и т.п.

Перейдем к рассмотрению более сложного случая, когда фиксированное заранее выделение необходимых ресурсов невозможно или нежелательно. В этом случае наиболее простым вариантом является ручное планирование вычислений. А именно, пользователь, которому для выполнения своего приложения нужны удаленные ресурсы, запускает команду поиска ldapsearch с указанием параметров нужных ему ресурсов. На основе полученной из MDS-сервера информации пользователь вручную размещает в сети задания с помощью, например, FTP или rcp (или запускает globusrun с соответствующими параметрами RSLstring и/или менеджера ресурсов). Производимая при этом работа, конечно, сильно зависит от типа рассматриваемого приложения. Если приложение не является распределенным, и требуется просто найти для его выполнения свободный компьютер, например, с определенной операционной системой, быстродействием и объемом оперативной памяти, то команда поиска может быть задана в следующем виде:

```
ldapsearch -h mds.globus.org -b “o=globus, c=us”, -s SUBTREE
    “(&(osname=hpux) & (osversion:=hppa1.1 hp hpux 10.20)
      &(osrelease:=10.20)&(ostype= hpux)
      &(freePhysicalMemory >10Mb)& (mflops>100M.)
      &(cpuload1> 0)) “
```

Назначение атрибутов, по которым производится поиск в представленных здесь примерах, описано в предыдущем разделе работы. Результаты поиска будут представлены пользователю в виде списка DN-имен найденных вычислителей, на которых установлена операционная ситема HP UX (указанной версии), имеющих более 10Mb свободной оперативной памяти, с производительностью процессоров более 100Mflops и находящихся в рабочем состоянии (cpuload1> 0).

Другой пример команды поиска выдает для всех пар взаимосвязанных в сети компьютеров (sourcehostname и destinationhostname) значения

пропускной способности и задержки канала между ними, определенные в моменты времени (modifytimestamp), начиная с полуночи 25 октября 1999 г.

```
ldapsearch -b 'o=Globus, c=US' -h mds.globus.org -p 389
  "(& (modifytimestamp>=19991025000000Z)
     (& (sourcehostname=*)
        (destinationhostname=*)))"
  bandwidth latency modifytimestamp
```

Конечно, в обоих случаях в качестве ответа можно получить очень большой объем информации, в которой трудно будет разобраться. Существенное сокращение объема результатов поиска может быть достигнуто за счет указания в поисковом фильтре дополнительных критериев отбора. Например, учитывая, что пользователя интересуют только те вычислители, на которых он реально может запустить свои задания, в запросе полезно было бы использовать уточняющий критерий вида allowedUser = < *имя-пользователя*>. К сожалению, в текущей версии MDS в описании вычислителей не предусмотрен атрибут, перечисляющий пользователей метакомпьютера, которым разрешен доступ. Этот очевидный недостаток видимо будет устранен в последующих версиях системы.

Другая возможность сократить объем результирующей информации связана с сужением базы поиска. Например, указание в предыдущих запросах в качестве базового DN “o=KIAM,o=Globus, c=US” вместо “o=Globus, c=US” сужает область поиска вычислительного ресурса, ограничивая его рамками конкретной организации (в данном случае – ИПМ). Развитием этой идеи является выполнение поиска по заданному пользователем списку организаций, либо поиск по признаку географической близости (например, по организациям России). Поскольку такие операции в общем случае трудоемки для исполнения вручную (требуют нескольких запросов ldapsearch и интегрирования полученных результатов), то для их эффективной реализации создаются соответствующие клиентские программы.

В оставшейся части настоящего раздела будут рассмотрены несколько прототипов клиентских программ, предназначенных для автоматизации процесса запуска и выполнения приложений в среде метакомпьютера, которые включают в себя возможности нетривиального поиска ресурсов, планирования вычислений, мониторинга хода вычислений и т.п.

Одна из таких систем, называемая Graphical Resource Selector (GRS), объявлена авторами проекта Globus [CFK98] (программа установки системы Globus даже содержит параметр включения GRS в установку, однако в распространяемую версию GRS не включена). GRS основана на сопоставлении двух графов G1 и G2, изображающих фрагменты сети. Граф G1 является шаблоном, который определяется пользователем и представляет конфигурацию, требуемую для выполнения приложения. В узлах графа указывается информация о компьютерах: тип, ОС, модуль, который будет

исполняться на данном компьютере, и т.д. На ребрах графа указываются требования на необходимые коммуникационные характеристики между данными узлами: тип протокола, пропускная способность, задержка и т.д. Используя информацию из MDS, система GRS строит граф G2, сопоставимый с G1. В графе G2 отображаются конкретные доступные в данный момент физические ресурсы, на которые можно распределить модули приложения. Граф G2 представляет собой фактический план размещения заданий по вычислителям, на основании которого действует подсистема коаллокирования DUROC.

Как видно, система GRS задумана как универсальная программа запуска распределенного приложения, базирующаяся на достаточно распространенном подходе к представлению приложений в форме графа. Следует обратить внимание, что сопоставление графов, лежащее в основе размещения вычислений в GRS, является довольно сложно реализуемой операцией. В процессе выполнения сопоставления может потребоваться многократное обращение к MDS, соединение (JOIN) полученных результатов, агрегирование результатов и т.п.

Другой подход к автоматизации запуска распределенных вычислений в метакомпьютере связан с построением планировщиков для конкретных приложений или типов приложений. На практике во многих случаях разработчики приложения реализуют собственный планировщик (брокер приложения), отвечающий за запуск и выполнение данного приложения. Использование знания об архитектуре приложения и других его специфических особенностей дает возможность создавать достаточно простые и эффективные брокеры. Такая специализация позволяет существенно более эффективно использовать информацию из ИСМ и более глубоко и многосторонне управлять всем процессом выполнения заданий (по сравнению с системами планирования общего назначения).

Более общий подход состоит в создании планировщиков, ориентированных на приложения определенного типа. Один из важных классов приложений, для которых такой подход является полезным, составляют приложения типа “хозяин-слуги”. В таких приложениях “хозяин” распределяет некоторый набор независимо обрабатываемых задач между “слугами”, результаты работы которых передаются обратно “хозяину” (этот процесс может выполняться в цикле). Независимость обрабатываемых “слугами” задач позволяет более свободно обращаться с ними при планировании.

Одной из известных систем обработки приложений типа “хозяин-слуги” является система Nimrod. Эта система разработана для выполнения вычислительных экспериментов следующего типа. Имеется некоторое параметризованное приложение, которое нужно выполнить для большого числа значений параметров (например, решение некоторой системы дифференциальных уравнений с различными граничными условиями) с последующим сравнением результатов вычислений и их визуализацией.

Система Nimrod предназначена для распределенного выполнения такого вычислительного эксперимента. Однако Nimrod не содержит средств автоматического планирования: она выбирает компьютеры (или кластерные системы) из фиксированного заранее списка (хранимого в файле) и передает им задания. При этом пользователю не гарантируется завершение работы его приложения за какое-то определенное время. Если ожидание ответа превосходит разумные пределы, то пользователь может принудительно снять задание с выполнения.

В настоящее время на базе системы Nimrod разрабатывается система Nimrod-G [AG97], использующая для обнаружения подходящих ресурсов и планирования запуска вычислений средства системы Globus. Пользователь системы Nimrod-G может задавать ограничения на окончательный срок завершения вычислительного эксперимента или на общую стоимость его выполнения. При этом система должна либо выполнить эксперимент в рамках заданных ограничений, либо сообщить пользователю, что при имеющихся ресурсах это невозможно. Эта функция системы обеспечивается с помощью соответствующей информации из ИСМ. Однако, так как значительная часть информации в ИСМ динамична и отражает действительное положение дел в метакомпьютере только с известной долей приближения, перспектива завершения вычислительного эксперимента в рамках данных ограничений не может быть абсолютно гарантирована. Для повышения степени гарантии можно использовать те или иные эвристические приемы.

С другой стороны, динамичность состояния метакомпьютера вызывает необходимость мониторинга (с помощью регулярного обращения к ИСМ) состояния выполнения отдельных задач эксперимента. В частности, в некоторый момент может выясниться, что обстановка на каком-либо из вычислителей (компьютеров или кластеров) изменилась настолько, что дальнейшее выполнение каких-то заданий вызовет выход за пределы критических ограничений на время или стоимость эксперимента. В этом случае задания забираются из вычислителя с тем, чтобы передать их в дальнейшем на другие подходящие вычислители.

Некоторый подход к построению инструментальной системы для создания планировщиков, ориентированных на приложения, предлагается в [BW97].

Можно заметить, что для более успешного планирования вычислений в метакомпьютерной среде важную роль играет такой механизм как предварительное резервирование ресурсов. При использовании этого механизма планировщик не только формирует план размещения заданий по ресурсам, но и гарантирует получение этих ресурсов в соответствии с заранее составленным расписанием. Однако, резервирование в метакомпьютерной среде представляет собой весьма сложную проблему. По сравнению с обычными системами резервирования оно осложняется, в частности, непредсказумостью времени выполнения многих приложений и, соответственно, трудностью предсказания моментов освобождения тех или иных ресурсов. В настоящее время вопросы резервирования исследуются в

рамках проекта системы коаллокирования и резервирования ресурсов GARA. [FKL99], расширяющей возможности системы Globus. В связи с этим также ведутся некоторые исследования по возможности использования сведений об истории выполнения приложений в метакомпьютере для прогнозирования его дальнейшего поведения с целью выработки более оптимальных стратегий выполнения приложений (см., например, [STF99]).

Заключение

Метаинформация играет важную роль при организации высокопроизводительных вычислений в глобальных вычислительных средах. В работе были рассмотрены связанные с этим аспекты представления и использования метаинформации. Были названы основные классы объектов вычислительной среды, информация о состоянии которых должна отображаться в виде метаданных. Введено понятие информационной службы метакомпьютера (ИСМ) и описаны основные операции, которые эта служба должна поддерживать. Рассмотрены вопросы использования ИСМ с целью обеспечения различных схем оптимального запуска приложений и планирования вычислений в метакомпьютерной среде.

Значительная часть работы была посвящена вопросам реализации ИСМ. Показано, что достаточно адекватным инструментом для реализации ИСМ является служба директорий LDAP. Описана система MDS, представляющая собой экспериментальную реализацию ИСМ, выполненную в рамках проекта Globus. Подробно рассмотрена система классов, составляющих схему MDS, а также поддерживаемые в MDS процедуры организации сбора и автоматической актуализации метаинформации.

Рассмотренные в работе вопросы являются частью обширной проблематики, возникающей в рамках задачи создания глобальных распределенных вычислительных систем. В настоящее время эта область активно развивается, привлекая внимание большого числа исследователей. По нашему мнению, к числу наиболее актуальных проблем, связанных с совершенствованием ИСМ, принадлежат следующие.

Небходимо дальнейшее развитие модели директорий в направлении поддержки истории изменений состояния вычислительных ресурсов, что должно позволить строить прогнозы для более эффективного планирования выполнения заданий. В описание ресурса должно быть включено представление о стоимости использования этого ресурса, а также информация о его доступности для конкретных пользователей или групп пользователей.

К недостаткам модели LDAP, которые должны быть устранены в будущем, следует отнести отсутствие четких правил обеспечения корректной эволюции схемы и общей структуры директории, а также соответствующих механизмов синхронизации схемы в случае распределенных директорий. Также необходимо совершенствование языковых конструкций для задания

запросов к директории в сторону большей выразительности и декларативности. При этом для обеспечения эффективной реализации алгоритмов планирования явно не хватает конструкции соединения (типа реляционного JOIN), агрегативных операций (типа COUNT, MIN, MAX и т.д.), а также операций сортировки и группирования (типа ORDER BY и GROUP BY).

В процессе эволюции метакомпьютерных систем возникает необходимость интегрирования в ИСМ информации из внешних источников, например, собираемых службами мониторинга сетей данных. Для обеспечения интеграции подобной внешней информации, должны быть разработаны соответствующие шлюзовые интерфейсы, предоставляющие доступ как к внешним LDAP-директориям, так и к другим типам баз данных. Примером такой интеграции является проект RIB-Globus [WWWRib], направленный на аккумулирование в MDS информации из распределенной реляционной базы данных RIB. В этой базе собрана ценная информация о программном обеспечении, установленном на компьютерах по всему миру.

Еще одним перспективным направлением в метакомпьютинге является дальнейшее совершенствование клиентских рабочих мест, с целью создания дружелюбного пользовательского интерфейса, а также реализации интеллектуальных брокеров и планировщиков, способных эффективно организовать выполнение заданий, оптимизировать стоимостные и временные характеристики, обеспечить устойчивое прохождение заданий в случае сбоев. Такие программы могли бы также непосредственно обращаться за метаинформацией к нескольким источникам, возможно имеющим разные структуры.

Список литературы

[AG97] D.Abramson, and J.Giddy, "Scheduling Large Parametric Modelling Experiments on a Distributed Meta-computer", PCW '97, September 25 and 26, 1997, Australian National University, Canberra,
pp P2-H-1 - P2-H-8.

[Ber98] F.Berman, “High-Performance Scheduling”, a chapter from the book "The Grid - Blueprint for a New Computing Infrastructure", July 1998.
http://www.mkp.com/books_catalog/1-55860-475-8.asp

[BW97] F.Berman, and R.Wolski, “The AppLeS Project: A Status Report”, Proceedings of the 8th NEC Research Symposium, Berlin, Germany, May 1997.
http://www.cs.ucsd.edu/groups/hpcl/apples/pubs/nec97.ps

[CFK98] K.Czajkowski, I.Foster, C.Kesselman, S.Martin, W.Smith, and S.Tuecke, “A Resource Management Architecture for Metacomputing Systems”, In Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing,1998

[FFK97] S.Fitzgerald, I.Foster, C.Kesselman, G.von Laszewski, W.Smith, and S.Tuecke, “A Directory Service for Configuring High-Performance Distributed Computations”, In Proc 6th IEEE Symp. on High Performance Distributed Computing, pp.365-375, IEEE Computer Society Press, 1997

[FK97] I. Foster, and C.Kesselman, “Globus: A Metacomputing Infrastructure Toolkit.”, Intl J. Supercomputer Applications, 11(2):115-128, 1997.

[FKL99] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy, “A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation”, Intl Workshop on Quality of Service, 1999.

[JBH98] H.Johner, L.Brown, F.-S. Hinner, W.Reis, and J.Westman, “Understanding LDAP”, ITSO Redbook SG24-4986-00, June 1998

[JLM99] H.V.Jagadish, L.V.S.Lakshmanan, T.Milo, D.Srivastava, and D.Vista, “Querying Network Directories”, In Proc. ACM SIGMOD Int.Conf. on Management of Data, June 1999

[STF99] W. Smith, V. Taylor, I Foster, “Using Run-Time Predictions to Estimate Queue Wait Times and Improve Scheduler Performance”, Proceedings of the IPPS/SPDP '99 Workshop on Job Scheduling Strategies for Parallel Processing.

[WWWGlo] http://www.globus.org

[WWWRib] http://icl.cs.utk.edu/projects/rib-globus

[ВКС2000] М.К.Валиев, Е.Л.Китаев, М.И.Слепенков, “Служба директорий LDAP как инструментальное средство для создания распределенных информационных систем”, Препринт ИПМ, 2000