# Функциональные возможности и методы реализации программного обеспечения Грид (часть 2)[1]

В.Н. Коваленко

**Аннотация.** Появление распределённых ресурсных инфраструктур, созданных на основе концепции грида, открыло новые возможности для решения важных задач в различных прикладных областях. Однако, решая вопрос об использовании технологий грида, важно учитывать фактическое состояние программного обеспечения в этой сфере. В статье предлагается систематизированное изложение функциональных возможностей наиболее распространённых, проверенных временем программных средств для создания грид-инфраструктур и разработки приложений.

**Ключевые слова**: грид, программное обеспечение, Globus Toolkit, gLite, функциональное описание.

## 4. Комплекс gLite

Подход к созданию программного обеспечения грида, выраженный в архитектуре OGSA, ставя задачу поддержки функций, которые типичны для различных распределённых систем, не предполагает определения исчерпывающего их набора. Состав таких функций невозможно установить априорно, и именно поэтому вопросы инструментальности архитектуры – стандартизация и композиционность – имеют первостепенную важность. Программное обеспечение, соответствующее OGSA, должно не только отражать современное представление о требуемой функциональности, но и способствовать ее расширению в форме новых служб и механизмов. Globus Toolkit можно квалифицировать как прототип платформы грида, но следует иметь в виду, что в нем зафиксированы главные архитектурные принципы, однако состав реализованных в нем служб не является окончательным или полным.

Движущей силой в направлении развития функциональных возможностей выступает опыт создания и использования гридов, чем и объясняется интерес к комплексу gLite, анализу которого посвящена работа автора [43]. Комплекс gLite разработан в проекте EGEE для поддержки крупномасштабной грид-инфраструктуры, которая функционирует в течение последних 5 лет и которую в круглосуточном режиме используют несколько тысяч пользователей.

В связи с необходимостью обеспечения производственного режима, проектирование gLite было изначально направлено на повышение уровня функций работы с ресурсами и обеспечения технологических свойств. При этом gLite в большой степени отражает основные принципы грида и опирается на Globus Toolkit как основу для реализации. В gLite также использованы результаты многих исследовательских проектов: Condor [44], DataGrid [45], DataTag [46], GriPhyN [47], iVDGL [48], – а ряд разработанных в этих проектах программных компо-

[1] Работа выполнена при поддержке фонда РФФИ (грант № 06-07-89111-а), программы фундаментальных исследований Президиума РАН, гранта Президента РФ для ведущих научных школ НШ-2139.2008.9.

нентов включен в gLite с той или иной степенью доработки.

Как и почти все современное программное обеспечение грида, gLite находится в стадии развития и значительно менее стандартизирован, чем Globus Toolkit версии 4. В частности, в работе с ресурсами он опирается на Globus Toolkit версии 2, то есть не придерживается стандартов Web-служб, хотя и ведутся работы по его переводу на новую базу.

### 4.1. Система управления заданиями

Служба GRAM, входящая в состав Globus Toolkit, реализует базовый уровень интеграции компьютерных ресурсов – операции удаленного запуска и управления заданиями. В этих операциях должен быть явно указан адрес того ресурсного центра, на котором должна производиться обработка задания, что предполагает либо жесткую связь приложения с определенным ресурсным центром, либо участие пользователя в его выборе (для этого он может воспользоваться информационной службой). Такого рода ограничения мало приемлемы в гриде большого масштаба, и система управления заданиями WorkLoad Management System (WLMS) [49], входящая в комплекс gLite, дает решение по автоматическому выбору ресурсов. Этот аспект управления заданиями мы будем называть *планированием*.

WLMS выполняет планирование в контексте общего процесса обработки заданий – *диспетчеризации*, которая включает функции, поддерживаемые и службой GRAM:

- доставку исполняемых файлов и входных файлов на исполнительные ресурсы;
- контроль выполнения задания;
- по окончании задания доставку результирующих файлов на серверы хранения (в частности, в точку его запуска).

Сценарий обработки заданий при наличии диспетчера выглядит следующим образом: все запросы запуска задания направляются по адресу WLMS, которая путем планирования определяет исполнительный ресурс и от имени субъекта, выдавшего запрос, передает на него задание (Рис.3).

#### 4.1.1. Способ автоматического выбора исполнительных ресурсов

Выбор ресурсов в WLMS производится отдельно для каждого задания путем сопоставления описания задания с состоянием ресурсов грида, которые содержатся в информационной базе.

Данные о состоянии ресурсов собираются и оперативно обновляются в результате мониторинга ресурсов, выполняемого информационной службой gLite. Схема информационной базы (Glue-схема [50]) ориентирована на представление интегральных характеристик ресурсных центров и предполагает однородность входящих в состав каждого центра компьютеров. Элементы схемы определяют такие данные, как общее количество процессоров и их быстродействие, платформа, объем оперативной и внешней памяти исполнительных компьютеров. Динамические характеристики пред-

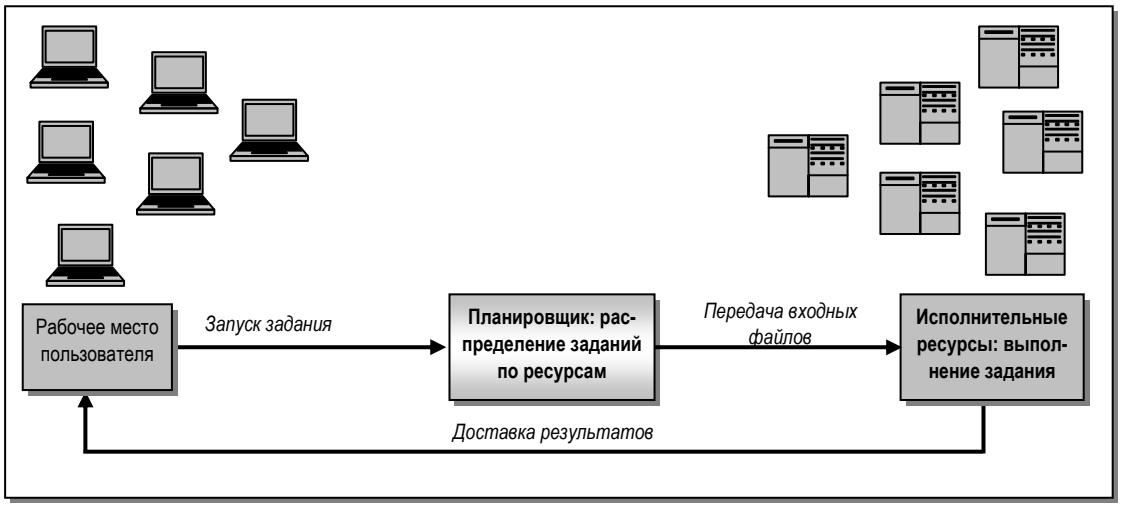


Рис. 3. Сценарий обработки запросов при автоматическом выборе ресурсов

ставлены числом заданий в очереди, количеством свободных и занятых процессоров и т.д. Данные, содержащиеся в Glue-схеме, позволяют выполнять планирование с точностью до ресурсного центра, который в gLite называется Computing Element (CE), то есть определяется не конкретный исполнительный компьютер, а некоторый CE.

Как принято в гриде, планирование (и диспетчеризация в целом) осуществляется исходя из формального описания задания. В WLMS для этого служит язык JDL (Job Description Language) [51], основанный на аппарате ClassAdd (Classified Advertisements) [52]. Аппарат ClassAdd представляет собой общий способ декларации требований к характеристикам объектов в виде пар: <атрибут> = <значение>, причем состав атрибутов может быть произвольным. Опора на ClassAdd позволила применить в качестве механизма выбора ресурсов алгоритм Matchmaking, ведущий начало из системы Condor. Язык JDL определяет конкретный набор атрибутов, которые используются для описания заданий в WLMS. Пример описания задания приведен на Рис. 4.

```
[Type = "Job";
JobType = "Normal";
Executable = "myexe";
StdInput = "myinput.txt";
StdOutput = "message.txt";
StdError = "error.txt";
InputSandbox = {"/users/pacini/example/myinput.txt",
"/users/pacini/example/myexe"};
OutputSandbox = {"message.txt", "error.txt"};
Requirements = other.GlueCEInfoLRMSType == "PBS";
Rank = other.FreeCPUs;]
```

Рис. 4. Пример файла описания задания

Среди приведенных в примере атрибутов можно выделить три основные группы, которые определяют:

- используемые в задании файлы (атрибуты Executable, StdInput, StdOutput, StdError, InputSandbox, OutputSandbox),
- способ выбора ресурсов (Requirements, Rank),
- тип задания (Type, JobType).

На основе первой группы атрибутов осуществляется доставка входных/выходных файлов задания.

Выбор исполнительных ресурсов управляется двумя атрибутами: требованиями (Requirements) и рангом (Rank). Требования определяют подмножество ресурсов, которые подходят для выполнения задания. Ранг выражает предпочтения на множестве подходящих ресурсов.

Значением атрибута Requirements является булевское выражение, в котором в качестве переменных выступают элементы Glue-схемы (в записи им предшествует префикс “other.”). Задание может быть распределено на CE, если значение выражения, вычисленное по текущему состоянию CE, равно true. Пример:

```
Requirements = other.GlueCEInfoLRMSType == "PBS" &&
other.GlueCEInfoTotalCPUs > 2 && Member ("IDL1.7",other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

Это выражение задает требование, чтобы выбираемый CE имел в качестве локального менеджера систему PBS и в нем было не меньше 2 процессорных единиц. В выражении используется также функция classAd – Member, которая возвращает true, если на компьютерах CE установлена программа "IDL1.7".

Атрибут ранжирования ресурсов Rank позволяет определить, какие CE являются предпочтительными для выполнения задания. Он задается арифметическим выражением, зависящим от значений элементов состояния CE. Задание будет направлено на CE, если: оно удовлетворяет требованиям, заданным в атрибуте Requirements, и имеет наивысшее значение атрибута Rank. Обычно этот атрибут употребляется для минимизации времени ожидания задания в очереди CE. Один из возможных вариантов задания Rank:

```
Rank = other.GlueCEPolicyMaxRunningJobs – other.GlueCEStateRunningJobs;
```

Здесь атрибут GlueCEPolicyMaxRunningJobs представляет максимальное число заданий, которые могут одновременно выполняться на CE, GlueCEStateRunningJobs – число выполняющихся в нем заданий.

Другой способ (применяемый по умолчанию):

```
Rank = -other.GlueCEStateEstimatedResponseTime;
```

Указанный здесь элемент состояния представляет собой оценку времени ожидания задания в очереди локального менеджера ресурсов

до начала выполнения. Такой способ определения Rank кажется самым предпочтительным, однако оценка времени ожидания может быть дана в WLMS очень приблизительно.

При подборе ресурсов учитывается также политика, регламентирующая права доступа к ним. Для этого диспетчер заданий взаимодействует с сервером виртуальной организации VOMS.

### 4.1.2. Типы обрабатываемых заданий

Следующая новация gLite – поддержка со стороны WLMS нескольких типов заданий. Формально тип задается в описании задания двумя атрибутами: Type и JobType (Рис. 4).

Особой интерес представляют составные задания, для которых атрибут Type имеет значение “DAG”(directed acyclic graph of dependent jobs) [53]. Составные задания применяются для организации параллельно-последовательной обработки данных, состоящей из ряда этапов, часть из которых зависима по входным данным от результатов других этапов. Зависимость этапов проявляется в том, что некоторые из них могут выполняться параллельно, а другие не могут начать выполнение прежде, чем закончатся их предшественники. Этому соответствует модель составного задания – ациклический граф с ориентированными ребрами, в котором узлы обозначают задания. Ребро графа, направленное из узла A в узел B, означает, что задание B не может быть выполнено раньше задания A (Рис. 5).

В текстовой форме описание составного задания представляется JDL-файлом специального вида, которая включает описание задания в целом, описания его простых составляющих и зависимости. Структура JDL-файла выглядит следующим образом.

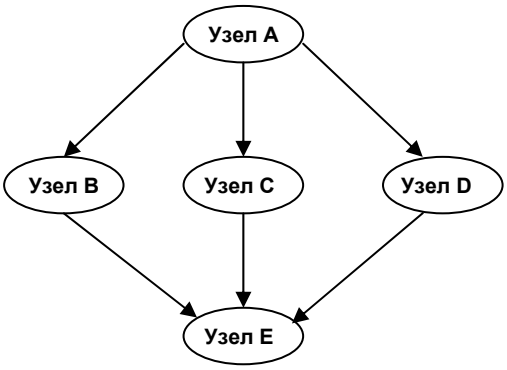


Рис.5. Пример DAG составного задания

Обработкой составных заданий управляет специальная компонента WLMS – DAG Manager (Dagman), которая контролирует завершение составляющих и осуществляет запуск новых, если все задания, от которых они зависят, выполнены.

Специальным случаем составных заданий являются сериализуемые задания (JobType = "partitionable"). Их составляющие – подзадания с одним и тем же исполняемым файлом, который требуется выполнить многократно с различными входными данными. В этом случае задания не зависят друг от друга и имеется компактный способ записи JDL-файла – в нем описывается одна сериализуемая составляющая и две другие, выполняющие пред- и постобработку.

При запуске сериализуемого задания генерируется DAG-задание, состоящее из требуемого числа серийных заданий, а также заданий пред- и постобработки. Задание предобработки выполняется первым и от него зависят все серийные заданий. Задание постобработки зависит от всех серийных, в его функцию входит сборка полученных результатов.

```
[
Type = "dag";
<Общие атрибуты составного задания>
nodes = [
                <Имя составляющей> = [description = [атрибуты составляющей]];
          <Имя составляющей> = [description = [атрибуты составляющей]];
…
dependencies = <зависимость узлов>;];
];
```

Помимо составных WLMS реализует также управление следующими типами заданий:

- интерактивными (атрибут JobType имеет значение “Interactive“); задание такого рода поддерживает связь с точкой запуска;
- параллельными (JobType=“MPICH”), то есть требующими для выполнения нескольких процессоров;
- с контрольными точками (JobType=“Checkpointable”); такие задания сохраняют промежуточные состояния и при необходимости их можно перезапустить не с начала, а с какого-то из запомненных состояний.

Перечисленные типы могут сочетаться друг с другом, например, JobType= {“Checkpointable”, “MPICH”}.

#### 4.1.3. Обеспечение технологических свойств

Обработка задания, выполняемая системой WLMS, включает три последовательных этапа, выполняемые отдельными компонентами.

1. Прием запросов – осуществляется службой WMProxy, реализованной в соответствии со спецификациями Web-служб.
2. Определение исполнительных ресурсов – функция планировщика Workload Manager.
3. Передача задания в ресурсный центр – производится компонентой CondorC по протоколу службы GRAM [54] системы Globus Toolkit. В случае составных заданий в управлении запуском составляющих участвует Dagman.

На всех этапах распределенной обработки заданий действуют механизмы, соответствующие технологиям обеспечения технологических свойств распределенной среды.

- Ведется мониторинг переданных в CE заданий. CondorC периодически опрашивает службу управления заданиями GRAM, установленную в ресурсных центрах, выясняя состояние задания, и перезапускает его на другие ресурсы в случае сбоя, не вызванного ошибками самого задания. Используя ту же информацию, Dagman осуществляет запуск частей составного задания в соответствии с графом зависимостей.
- Программа Log Monitor (LM) наблюдает за лог-файлами CondorC, перехватывая существенные, то есть изменяющие состояние заданий события.
- Служба протоколирования Logging&Bookkeeping (LB) агрегирует информацию от различных компонент WLMS и сохраняет описания событий, происходящих с заданием. Все компоненты WLMS содержат код, поставляющий информацию в LB. Пользователь может узнать состояние задания, опрашивая с помощью команд интерфейса службу LB. Кроме того, он может подписаться на получение уведомлений о каких-либо определенных изменениях состояния, например, об окончании задания.
- Служба учета – DataGrid Accounting System (DGAS) собирает данные о количестве полученных заданием ресурсов.

### 4.2. Управление файловыми данными

Управление данными основывается в gLite на тех же принципах стандартизации дистанционного доступа, что и в Globus Toolkit, но решает задачи виртуализации доступа к файлам и объединения ресурсов хранения с большей полнотой.

Технология объединения ресурсов хранения позволяет создать общее поле постоянной памяти из множества распределенных устройств хранения различных типов: допускаются отдельные диски, дисковые массивы, ленточные устройства массовой памяти (MSS). Устройства хранения структурированы подобно вычислительным: устройства из одного административного домена объединяются в ресурсный центр Storage Element (SE), который управляется локальным менеджером. Совокупность SE образуют грид хранения данных – доступ к каждому SE осуществляется по унифицированным интерфейсам и протоколам, и память устройств SE может использоваться потенциально любым пользователем грида для размещения файлов.

Технология виртуализации, с помощью которой образуется единое пространство адресации файлов, основана на концепции глобальной файловой системы: после того как файл помещается в грид, он становится доступен пользовательским программам, где бы они ни выполнялись, примерно так же, как в локальной среде одного компьютера – по имени и посредством стандартных операций. Поддерживается иерархическое структурирование пространства име-

нования, то есть помимо собственно файлов в нем могут создаваться директории. Поэтому, хотя файловая система одна на всех пользователей, может быть обеспечена уникальность присваиваемых пользователем имен – логических имен файлов LFN.

#### 4.2.1. Средства работы с файлами

Работа с глобальной файловой системой осуществляется с помощью набора средств ввода/вывода gLite I/O, которые представлены, во-первых, в форме прикладного программного интерфейса, во-вторых, в виде утилит, вызываемых через интерфейс командной строки. В составе операций можно выделить три группы: операции с директориями, операции передачи файлов, операции удаленного доступа к файлам.

Операции работы с директориями позволяют:

- устанавливать права доступа (chmod);
- создавать, переименовывать и удалять директории (mkdir, mv, rmdir);
- выдавать состав директорий (ls).

Операции передачи файлов реализованы в трех утилитах: glite-put, glite-get, glite-rm, которые, соответственно, помещают файл, хранящийся в файловой системе компьютера (локальный файл), в грид, копируют из грида в локальный файл, удаляют файл. Команды выглядят следующим образом:

glite-put <localfilename> lfn://<lfn>

glite-get lfn://<lfn> <localfilename>

glite-rm lfn://<lfn>

Здесь <localfilename> – имя локального файла, который помещается в грид под именем <lfn> (в операции put) или в который помещается файл из грида. Семантически помещение файла в грид означает, что производится его копирование в некоторый SE. Выбор места размещения файла определяется конфигурационным файлом, в котором задается “ближайшее” SE, однако, по-видимому, в дальнейшем будут использоваться более изощренные стратегии.

Перечисленный набор файловых операций и операций с директориями решает проблему доступа к файлам из приложений. Соответствующая дисциплина состоит в том, что файлы постоянно хранятся в гриде, а перед началом выполнения приложения копируются (операция get) в локальную файловую систему. Далее доступ из приложения к файлам может осуществляться обычными средствами, и, таким образом, не требуется адаптации кода приложения к условиям грида.

В некоторых случаях может оказаться более предпочтительным вариант непосредственного удаленного доступа к файлу, хранящемуся в гриде на SE. При такой дисциплине требуется модификация приложения: оно должно быть собрано с клиентской библиотекой gLite I/O, которая включает операции read/write/open/close/lseek стандарта POSIX.

В дополнение рассмотренных средств, для массовой передачи файлов в gLite используется служба FTS (File Transfer Service), аналогичная RFT системы Globus Toolkit.

#### 4.2.2. Поддержка виртуализации файлов и репликация

Возможность работы с файлами по их логическим именам предполагает наличие механизма обнаружения их мест хранения. Этот механизм реализуется службой каталогов – LCG File Catalog (LFC), которая также поддерживает репликацию файлов. Технология репликации заключается в хранении нескольких экземпляров (реплик) файлов на разных SE, благодаря чему можно уменьшить нагрузку на сеть, “приблизив” в метрике сетевых ресурсов место хранения к месту регулярного использования файла.

Задача обнаружения файла, решаемая LFC, состоит в том, чтобы по логическому имени LFN найти “физическое” имя SURL вида: <sfn | srm>://<SE_hostname>/<path>, где <sfn|srm> – обозначение альтернативных протоколов доступа, SE_hostname – адрес SE, на котором он хранится, а path, идентифицируя файл в локальной среде SE, зависит от организации этого SE (протокола доступа). Если поддерживается репликация, одному LFN может соответствовать произвольное количество идентификаторов SURL, адресующих SE, на которых хранятся реплики одного файла, так что отношение LFN<-> SURL имеет ранг один ко многим.

Связь между LFN и SURL поддерживается службой каталогов LFC. При помещении файла средствами glite I/O в грид (операция put) производится не только его пересылка на определенный элемент хранения SE, но и регистра-

ция идентификаторов в каталоге. Каталог также неявно корректируется при выполнении операций переименования и удаления. В операциях доступа выбор той или иной реплики осуществляется автоматически по принципу статически определяемой “ближайшей”.

Помимо определения месторасположения файлов служба каталогов решает задачу управления пространством именования: именно в каталоге хранится структура директорий, состав файлов и права доступа.

### 4.2.3. Организация ресурсов хранения

В glite предложен способ организации ресурсов хранения на основе абстракции элемента памяти – Storage Element (SE). SE включает одно или несколько расположенных в одном домене устройств хранения, образующих ресурсный центр. Вся совокупность устройств SE управляется локальным менеджером, поддерживающим размещение файлов и операции над ними. Примерами локальных менеджеров являются Disk Pool Manager (DPM) [55] – для множества дисков, dCache [56] – для дисковых массивов, CASTOR [57] – для систем массовой памяти с кеширующими дисками и ленточным хранилищем. В соответствии с принципами архитектуры грида удаленный доступ к памяти SE осуществляется через стандартизованные интерфейсы трех служб SE. Первые две:

- служба передачи файлов, поддерживающая протокол GridFTP,
- служба доступа к файлам с внешним интерфейсом gLite I/O, – осуществляют транзит запросов, производя отображение протоколов грида во внутренние протоколы локальных менеджеров (Рис. 6).

Третья служба – Storage Resource Management (SRM) предоставляет стандартизированный интерфейс для управления устройствами хранения, выполняя такие функции как:

- перемещение файлов между дисками и другими носителями,
- резервирование памяти,
- подготовка данных для передачи по определенному протоколу.

Отметим, что набор поддерживаемых протоколов является расширяемым, и поддержка двух: gLite I/O и GridFTP, является минимальным требованием. В перспективе интерфейс SRM должен поддерживать более широкий состав операций, таких как выделение квот, управление временем жизни файлов, закрепление файла на диске, чтобы он не перемещался на память второго уровня.

В соответствии с описанной организацией SE схема обработки запроса, к примеру, на открытие файла посредством gLite I/O выглядит так (Рис. 7):

- Клиентская библиотека получает от приложения в качестве параметра LFN файла и передает его службе gLite I/O (1).
- Производится авторизация операции посредством службы File Authorization Service,

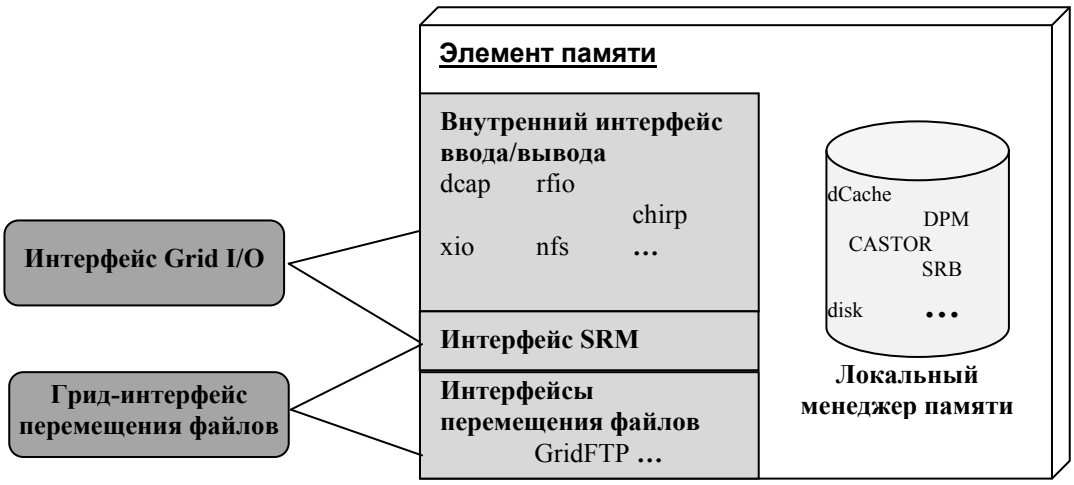


Рис. 6. Грид-интерфейсы элемента памяти SE

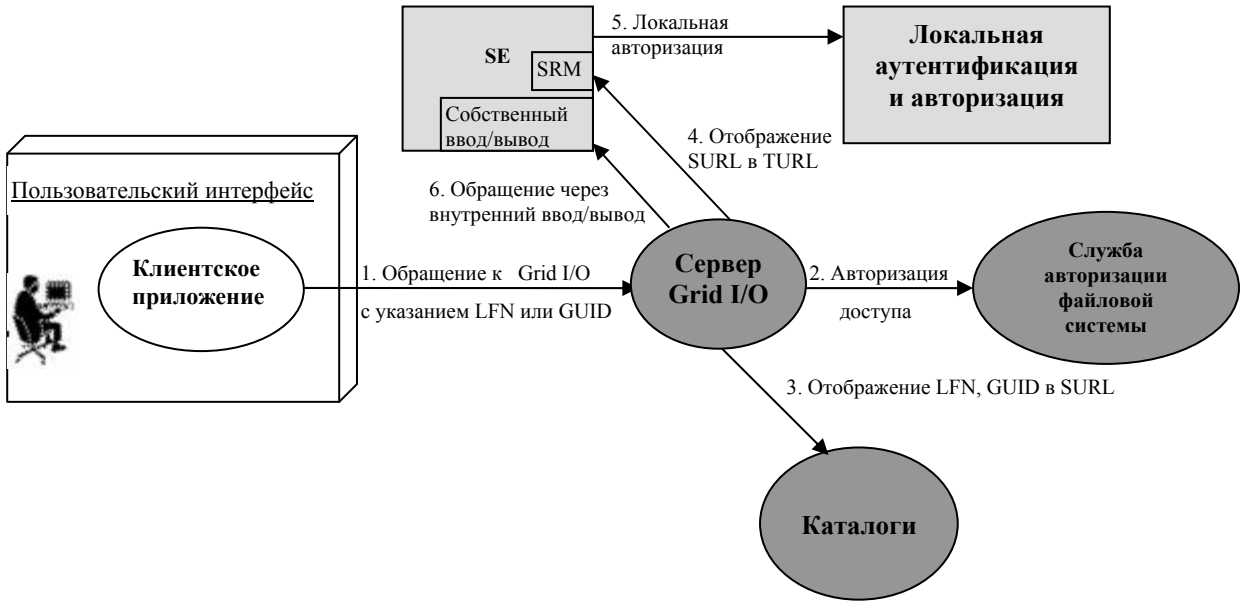


Рис. 7. Схема выполнения файловых операций

и на основе пользовательского сертификата проверяется, имеет ли право пользователь на выполнение операции (2).

- Путем обращения к каталогу глобальное имя файла разрешается в идентификатор SURL, в котором уже определено конкретное SE (3).
- Служба gLite I/O обращается к SRM этого SE, получая в ответ локальный идентификатор TURL (Transport URL), соответствующий внутреннему протоколу локального менеджера (4, 5).
- Производится обращение к интерфейсу POSIX I/O (6), который запускает внутренний протокол для выполнения операции.

### 4.3. Информационная система

В комплексе gLite информационная система играет вспомогательную роль, осуществляя сбор, хранение и поиск сведений о деятельности пользователей и состоянии ресурсов. Эти сведения используются в других компонентах комплекса, включая управление заданиями и данными. Фактически параллельно существуют две информационные системы, причем каждая обслуживает свой состав данных. Первая – Globus Monitoring and Discovery Service версии 2 (MDS2) [58] базируется на OpenLDAP (открытой реализации протокола Lightweight Directory Access Protocol, LDAP). Мы будем рассматривать второй вариант, построенный с помощью системы R-GMA (Relational Grid Monitoring Architecture) [59] в соответствии с архитектурной моделью мониторинга грида GMA (Grid Monitoring Architecture) [60].

Возможности этой модели представляются применимыми в широком классе приложений информационного характера, позволяя решать такие задачи как:

- распределенное хранение структурированной информации в множестве баз данных и поиск по интегральному информационному массиву;
- поставка в оперативном режиме информации от распределенных источников в базу данных (мониторинг);
- поставка информации от распределенных источников в приложения.

Первая задача подобна той, которая решается традиционными системами управления базами данных (СУБД). В этом плане R-GMA является СУБД реляционного типа, следуя положениям реляционной модели. В базе данных информация хранится в форме таблиц, которые состоят из множества кортежей (строк). База данных определяется схемой, описывающей структуру каждой таблицы: состав атрибутов и их типы. Работа с информацией ведется с помощью подмножества языка SQL, совместимого со стандартом SQL92. В числе основных

операций: вставка строки (Insert), извлечение множества строк по поисковому критерию (Select), определение таблицы (CreateTable).

Особенность R-GMA в том, что это СУБД для распределенных баз данных, в которых различные реляционные таблицы или части одной и той же таблицы могут храниться на разных ресурсах, а наличие общего централизованного хранилища хотя и возможно, но не обязательно. Части базы данных, физически располагающиеся на одном ресурсе, будем называть фрагментами. Заметим, что в R-GMA отдельный фрагмент локально управляется собственной СУБД (MySQL).

Несмотря на распределенность базы данных, при использовании она выступает как единое целое: имеет общую схему и поисковые операции выполняются по всему объединенному информационному массиву. Механизм виртуализации, необходимый для обеспечения доступа к информации независимо от места ее расположения, опирается на Реестр – компоненту R-GMA, в которой в дополнение к схеме содержатся описания состава данных отдельных фрагментов, что позволяет отрабатывать запросы по всем релевантным фрагментам. В связи с этим механизмом базу данных, поддерживаемую R-GMA, называют виртуальной.

Другая выполняемая R-GMA функция – мониторинг: помимо вопросов хранения и поиска, традиционных для локальных информационных систем, в ней решаются указанные выше задачи 2 и 3 оперативного сбора и доставки информации. R-GMA предлагает средства, с помощью которых:

- может быть создана виртуальная БД и отдельные фрагменты;
- могут быть установлены связи между источниками информации и фрагментами БД, причем один источник может поставлять данные множеству фрагментов и один фрагмент может получать данные от многих источников;
- в виртуальную БД могут быть включены фрагменты, созданные отличными от R-GMA средствами;
- поставка данных от источников может осуществляться не только в БД, но и в приложения.

### 4.3.1. Принципиальная схема оперативного сбора и доставки информации

И задачи поиска, и задачи поставки информации решаются в R-GMA единообразно – в основе лежит схема взаимодействия множества распределенных программных компонент Поставщиков и Потребителей данных (Рис. 8). Поставщик представляет собой компоненту,

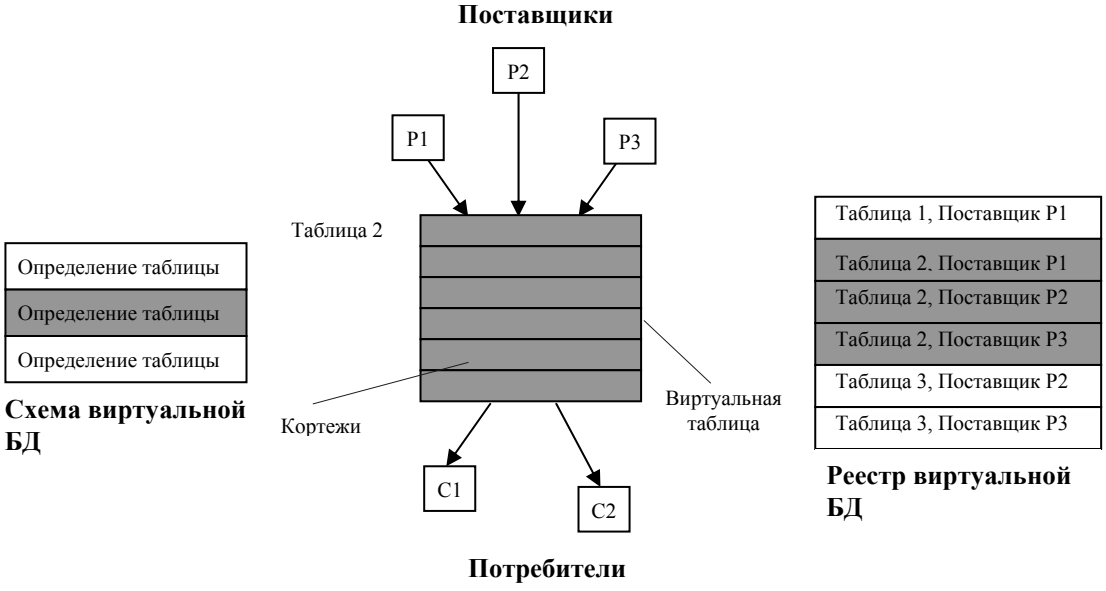


Рис. 8. Основные компоненты R-GMA

которая принимает информацию и вводит ее в виртуальную БД. Как будет показано далее, существует несколько вариантов, откуда информация поступает Поставщику, один из них – поставка от источников данных. Поставщик делает поступающие данные доступными Потребителям. Для этого Поставщик объявляет состав данных, которые он способен поставлять, декларируя одну или несколько таблиц (точнее, некоторую выборку столбцов таблицы), определенных в Схеме виртуальной БД.

Чтобы получать данные, Потребитель не обращается к какому-то конкретному Поставщику, а публикует поисковый SQL-запрос Select Where, получая в ответ удовлетворяющие запросу данные от разных Поставщиков. Интерпретация запроса, включающая определение списка релевантных Поставщиков и выдачу им частичных поисковых запросов, поддерживается автоматически программными компонентами Реестр и Потребитель.

Способ взаимодействия Поставщиков с Потребителями применим не только для поиска, таким же образом выполняется и поставка данных. R-GMA является системой мониторинга, в связи с чем она расширяет реляционное представление данных еще в одном аспекте. Поставщики способны хранить историю – кортежи, которые имеют одинаковые ключи, но относятся к разным моментам времени поставки. При вводе к кортежам автоматически добавляется временная метка (time stamp). Предполагается, что кортежи хранятся в течение ограниченного периода времени (retention period), а по его истечении автоматически удаляются. При наличии истории возможны несколько режимов возврата результатов поисковых запросов: Latest – выдача удовлетворяющей запросу информации из самых свежих кортежей, History – выдача информации из всех кортежей, поступивших за заданный период времени, Continuous – постоянная передача информации из кортежей, поступающих в течение заданного периода.

Результаты поисковых запросов заносятся во временный буфер (память кортежей) Потребителей, откуда могут быть считаны прикладным кодом с помощью специальной операции (Pop). Прикладным кодом, подключаемым к R-GMA на стороне Потребителя, может быть, например, пользовательский интерфейс, визуализирующий поступающие данные, либо программа, записывающая их на постоянное хранение в БД. Таким образом, описанная схема реализует цепочку доставки: источник - Поставщик - Потребитель - приложение, причем связи между Поставщиками и Потребителями имеют избирательный характер и могут устанавливаться динамически. В этом плане R-GMA выступает как средство адресной поставки данных от множества находящихся в разных местах Поставщиков к множеству распределенных Потребителей.

### 4.3.2. Типы программных компонентов R-GMA

В развитие рассмотренной схемы взаимодействия Поставщиков и Потребителей модель мониторинга определяет три разновидности Поставщиков – первичный (Primary Producer), вторичный (Secondary Producer) и по запросу (On-Demand Producer) (Рис. 9). Они различаются главным образом тем, откуда получают данные.

Первичный Поставщик получает данные непосредственно от источников. Под источником понимается прикладной код, который порождает данные и вводит их в виртуальную БД операцией Insert, выдаваемой Поставщику. Весьма важно, что в архитектуре мониторинга отдельный Первичный поставщик обслуживает ограниченный круг “ближайших” источников. Из этого следует, что состав собираемых таким Поставщиком кортежей одной таблицы неполон – он ограничен кругом связанных с ним источников, в то время как другие кортежи той же таблицы порождаются “чужими” источниками. Поэтому первичные Поставщики предназначены не для постоянного, а лишь для временного хранения фрагментов БД.

Функция постоянного хранения возлагается на вторичных Поставщиков, с помощью которых может быть организована многоуровневая распределенная структура, в которой в одном фрагменте БД содержатся таблицы с данными от нескольких других Поставщиков. Вторичный Поставщик, несмотря на название, совмещает роли и Потребителя, и Поставщика. Он получает информацию не от источников, а от любых других Поставщиков, выдавая им поисковый запрос, то есть использует

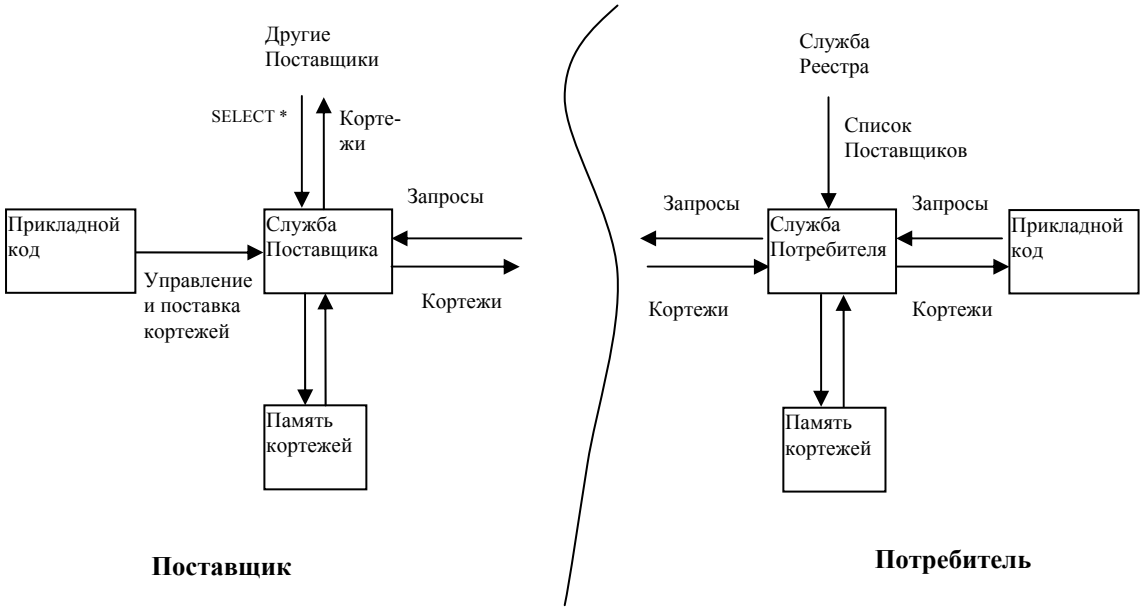


Рис. 9. Схема поставки информации от Поставщиков к Потребителям

технические средства Потребителя. С другой стороны, вторичный Поставщик способен поставлять свои данные другим Потребителям, в том числе другим Вторичным поставщикам – для этого он декларирует состав своих таблиц.

Вторичный Поставщик имеет память кортежей, но она уже предназначена для долговременного хранения и выступает как полноценный фрагмент БД. Более того, информация, поступающая Вторичному Поставщику, заносится в этот фрагмент автоматически, какого-либо дополнительного прикладного кода не требуется.

Основное назначение Вторичного Поставщика – архивация информации: он создает реальные таблицы из виртуальных, собирая кортежи, которые источники заносят в виртуальную БД. Вторичный Поставщик может хранить произвольный набор таблиц, изначально публикуемых разными Поставщиками. Это позволяет ему отвечать на поисковые запросы, включающие реляционную операцию Join между хранимыми у него таблицами. Помимо того, осуществляемая им републикация информации дает возможность дублировать данные, повышая надежность хранения и уменьшая нагрузку на других Поставщиков.

Последний тип Поставщика – по запросу – является средством интеграции внешних по отношению к R-GMA баз данных. В нем в качестве источника данных выступает прикладной код, роль которого – прием, преобразование и выполнение поисковых запросов от Потребителей в БД произвольной организации.

### 4.4. Средства поддержки функционирования грида

В условиях распределенной инфраструктуры и децентрализованной обработки запросов особым образом должны решаться вопросы обеспечения надежности, мониторинга процессов, обслуживания больших пользовательских коллективов. В gLite сделан существенный шаг в этом направлении, что дает основание выделить в его составе отдельную группу средств поддержки функционирования грида и обеспечения необходимых технологических свойств. В то же время следует отметить, что в реализации многих из этих средств не используются стандарты Web-служб, которые могли бы составить базу для соответствующих механизмов. Однако имеющаяся в gLite поддержка технологических свойств показывает направление развития этой группы технологий, определяя их функционально. Отметим также, что в полной мере технологии поддержки функционирования применяются пока лишь в контексте системы WLMS, хотя перспективы их использования в системе управления данными и информационной системе достаточно очевидны.

#### 4.4.1. Служба протоколирования процесса обработки заданий

Вычислительные задания обрабатываются в гриде распределенно различными компонентами: пользовательским интерфейсом, составляющими WLMS и ресурсного центра CE. В этих условиях возникает проблема информирования пользователей о ходе выполнения принадлежащих им заданий. Служба LB (Logging&Bookkeeping service) [61] решает эту проблему:

- предоставляя средства для агрегирования диагностических сообщений, относящихся к каждому отдельному заданию, в виде единого протокола обработки;
- обеспечивая хранение протоколов и предоставление их по запросам;
- посылая уведомления об изменении состояния задания.

Формирование протокола обработки происходит в результате того, что все компоненты WLMS посредством прикладного интерфейса LB регистрируют существенные события, происходящие с заданием (помещение в очередь, определение исполнительных ресурсов, начало выполнения и т.п.). Каждое событие фиксируется с набором атрибутов, одним из которых является уникальный идентификатор задания.

В конечном счете события собираются на сервере хранения протоколов, но реализуется это в два этапа. На первом этапе событие поступает локальному демону locallogger, который выполняется в непосредственной физической близости от источника события, что позволяет избежать каких-либо сетевых проблем при передаче. Демон записывает событие в файл и возвращает источнику подтверждение об успешности операции.

На втором этапе за доставку события отвечает другой демон interlogger. Получая событие либо прямо от демона locallogger, либо, в случае его падения, с диска, он пересылает его в конечную точку назначения – серверу хранения (bookkeeping server), причем надежность на этом этапе обеспечивается механизмом повторной передачи. Можно заметить, что в целом двухэтапный способ доставки сообщений соответствует схеме, применяемой в R-GMA: демоны LB играют роль Поставщика, а сервер хранения – Потребителя.

В грид-инфраструктуре может быть несколько серверов хранения, однако события всех заданий одного пользователя поступают только на один из них (он определяется статически в конфигурации). Сервер хранения выполняет несколько функций. Во-первых, обрабатывая поступающие “сырые” события, он формирует обобщенное представление о состоянии задания, в котором содержатся такие атрибуты, как описание задания (JDL-файл), адрес исполнительного CE, код завершения и прочее. Во-вторых, сервер хранения предоставляет интерфейс Web-служб для получения информации и о состоянии задания (команда job-status), и обо всех событиях (job-logging-info).

В-третьих, сервер хранения поддерживает подписку на получение уведомлений об изменении состояния задания. Для подписки клиент регистрируется на сервере хранения, задавая условия получения уведомлений (идентификатор задания, тип события). Уведомления доставляются принимающему клиенту на стороне пользователя асинхронно тогда, когда происходит событие, указанное в условиях регистрации (например, окончание задания). Этот способ слежения за заданием имеет преимущества, позволяя избежать многократного повторения запросов о состоянии с одним и тем же результатом и снизить тем самым нагрузку на сервер.

#### 4.4.2. Служба учета потребления ресурсов

Входящая в состав gLite служба учета – DataGrid Accounting System (DGAS) [62] представляет собой средство сбора информации об использовании ресурсов пользователями грида. Информационной единицей в этой системе выступает запись использования (Usage Records), которая создается для каждого обработанного задания и содержит такие данные:

- идентификатор задания,
- ресурсный центр, в котором оно выполнялось,
- использованные при выполнении ресурсы,
- время запуска на счет, поступления в очередь локального менеджера, окончания.

Сбор этих данных осуществляет демон учета, работающий в среде CE и взаимодействующий с его локальным менеджером. Механизм сбора выглядит следующим образом. Когда задание начи-

нает выполняться на исполнительном компьютере, его стартовый скрипт создает в специальной директории файл, содержащий имя задания, его идентификатор и адрес сервера, в который должна быть направлена формируемая запись использования. Обнаруживая такой файл, демон учета ищет учетную информацию о задании в лог-файле локального менеджера. Когда задание заканчивается, вся учетная информация выбирается из этого лог-файла и присоединяется к файлу, созданному вначале.

Собираемые записи использования хранятся в децентрализованной инфраструктуре, образованной множеством серверов, на которых установлена служба Home Location Register (HLR). Организация инфраструктуры хранения имеет в DGAS особенности, вызванные следующим обстоятельством.

Обслуживание запросов на выдачу информации в инфраструктуре из множества серверов существенно упрощается, если информация распределена между ними таким образом, что все данные, относящиеся к одному пользователю, сконцентрированы на одном из них. Именно так обстоит дело в DGAS: каждый пользователь приписан к своей “домашней ” службе HLR. Однако DGAS предназначена для обслуживания не только пользователей, но и провайдеров ресурсов. Если пользователи заинтересованы в получении сведений лишь о своих заданиях, то провайдерам нужны данные обо всех заданиях, которые были обработаны на их ресурсах. При ориентации только на пользовательское обслуживание провайдеры были бы вынуждены опрашивать каждый сервер.

В связи с этим в инфраструктуре хранения поддерживаются два вида служб HLR: для пользователей и для провайдеров. Записи использования, собираемые компонентами DGAS на CE, сначала поставляются в одну из возможных служб HLR, а затем дублируются ею в другую. Поддерживается несколько сценариев поставки:

- сначала в пользовательский HLR.

CE =>User HLR =>Resource HLR

- сначала в HLR провайдеров.

CE => Resource HLR => User HLR

- только в HLR провайдеров.

CE => Resource HLR

Представляет интерес еще одна служба в составе DGAS – служба расценок (Price Authority – PA). Эта служба поддерживает базу данных цен ресурсов и хранит историю их изменения. Цены в ней могут устанавливаться вручную администратором, но могут также и вычисляться программно по различным алгоритмам, например, по алгоритму торгов. В PA реализован механизм встраивания алгоритмов установки цен в форме динамической библиотеки.

Наличие расценок ресурсов в службе PA, с одной стороны, и количества потребляемых заданиями ресурсов в записи HLR, с другой, дают возможность вычислять стоимость выполнения заданий. Таким образом, реализованные в DGAS службы позволяют перейти к экономическим методам управления гридом на основе расчетов между пользователями и провайдерами.

#### 4.4.3. Безопасность и поддержка виртуальных организаций

Вопросы безопасности решаются в gLite на основе Инфраструктуры публичных ключей PKI [63], использовавшейся в версии Globus Toolkit 2. Основные положения PKI заключаются в следующем.

Объекты грида (пользователи и службы) располагают цифровым удостоверяющим документом – сертификатом стандарта X.509, который заверен Сертификационным центром (Certificate Authority). В качестве ключевого атрибута сертификат содержит уникальный идентификатор владельца Distinguished Name (DN). Наличие сертификата гарантирует надежную взаимную аутентификацию участников дистанционного взаимодействия, которая осуществляется программным способом и не требует участия пользователя.

Механизм делегирования прав открывает возможность выполнения операций программными компонентами от имени пользователя. Такая потребность возникает: например, запуск задания на CE должен производиться от имени владельца задания, но для этого компонента WLMS, выполняющая дистанционный запуск, должна предъявлять удостоверение владельца. Надежный способ делегирования полномочий состоит в том, что на основе базового сертификата генерируется временный сертификат (так

называемый прокси-сертификат) с коротким сроком действия, который и используется для контактов от имени пользователя.

Наибольший интерес представляют разработанные в gLite решения по второму аспекту безопасности – авторизации. Содержание авторизации заключается в том, что при входе пользователя в систему (в данном случае, в ресурсный центр) определяются его полномочия на выполнение определенных действий (выполнение приложения, доступ к файлу или таблице базы данных). В результате авторизации производится отображение полномочий на некоторый локальный профиль в конкретной среде безопасности, в Unix-системах – в экаунт пользователя.

Базовый механизм авторизации, ведущий начало от первых версий Globus Toolkit, основан на gridmap-файле – списке, в котором перечисляются имена DN всех имеющих доступ к данному ресурсу пользователей грида и соответствующие локальные регистрационные экаунты. Такой простой способ предполагает, что соглашения между пользователями и провайдерами – где и с какими правами пользователь может получать ресурсы – устанавливаются на персональной основе, что противоречит принципам грида и в реальных условиях невозможно.

Грид предполагает, что за организацию работы пользователей и их взаимоотношения с провайдерами отвечают виртуальные организации (ВО), которые определяют полномочия пользователей и заключают соглашения с провайдерами об их обслуживании. Первой попыткой улучшения базового механизма авторизации стало централизованное ведение списков пользователей грида виртуальной организацией. Список публикуется на LDAP-сервере, откуда каждый провайдер может его получить и с помощью утилиты mkgridmap сгенерировать gridmap-файл. Такое решение имеет два недостатка. Во-первых, состав пользователей подвержен частым изменениям и безопасность может быть обеспечена только в том случае, если gridmap-файл отражает эти изменения максимально быстро, так что все провайдеры должны регулярно (например, раз в сутки) обновлять список пользователей с сервера. Во-вторых, задача определения персональных прав пользователей в этой схеме по-прежнему возлагается на каждого провайдера.

В gLite предложена реализованная несколькими программными средствами процедура, которая автоматически определяет и проводит в жизнь локальную политику по авторизации пользователей на основе формального описания их полномочий.

**Формальное описание полномочий**. Аппарат авторизации опирается на формальное описание полномочий пользователя ВО тремя атрибутами [64]:

- группа ВО, к которой принадлежит пользователь,
- его роль внутри группы,
- возможности выполнения операций.

Совокупность этих атрибутов составляет так называемое “Fully Qualified Attribute Names” (FQANs) [65].

**Получение авторизационной информации**. Служба Virtual Organization Membership Service (VOMS), представляющая собой грид-интерфейс к реляционной базе данных, хранит авторизационные атрибуты пользователей ВО и выдает их по запросам. Административный клиент этой службы позволяет добавлять пользователей, задавать значения атрибутов, создавать группы и т.д. Пользовательский клиент контактирует со службой VOMS, предъявляя сертификат некоторого пользователя, и получает список (если пользователь принадлежит нескольким ВО) его групп, ролей и возможностей.

Использование VOMS позволяет создать схему, в которой авторизация проводится исключительно исходя из атрибутов FQANs, а доставка этих атрибутов осуществляется путем их включения в состав прокси-сертификата расширенного формата. Такой прокси-сертификат генерируется в результате обращения к службе VOMS (команда voms-proxy-init) и содержит информацию для авторизации в виде расширения – атрибутного сертификата [65]. В результате провайдеры ресурсов освобождаются от необходимости получать и обновлять списки членов ВО: информация для авторизации получается самим пользователем в начале сеанса работы в гриде и передается в составе прокси-сертификата при всех дистанционных контактах.

**Выполнение авторизации**. В среде ресурсов на основе представляемой в прокси-сертификате информации в соответствии с локальной политикой выносится решение по авторизации. Служба безопасности ресурсного центра (Gatekeeper) начинает с того, что проверяет валидность сертификата, и обращается к службе Local Credential Authorization Service (LCAS) [66].

LCAS представляет собой настраиваемую программную оболочку с разъемами для подключения вставных модулей. Поставляемые в ее составе модули поддерживают проверку специфических для ресурса ограничений: "черный список" пользователей, ограничения по запрашиваемому для обработки заданий времени и прочее. Модули, специально разработанные для VOMS, выделяют авторизационные атрибуты, вынося на их основе решение о возможности авторизации. Заметим, что LCAS может работать как с сертификатами VOMS, так и с сертификатами предыдущих версий, используя в этом варианте gridmap-файл.

Вторая служба – Local Credential Mapping Service (LCMAPS) - отображает авторизационную информацию на локальные права доступа, например в идентификаторы системы UNIX (uid,gid) таким образом, чтобы получаемые права ограничивали круг возможных действий только допустимыми. LCMAPS, как и LCAS, является модульной средой и способен поддерживать как фиксированное отображение авторизационных атрибутов в предопределенные значения пар (uid,gid), так и динамическое отображение в пул лизинга экаунтов, выделяемых, например, посредством программы Gridmapdir.

**Управление экаунтами**. Программа Gridmapdir [67] решает проблему создания и управления экаунтами в Unix-системах путем их динамического выделения из общего пула. Пул экаунтов, создаваемый администратором, не может быть очень большим: требуемое количество экаунтов ограничивается предполагаемым числом одновременно работающих в ресурсном центре пользователей. Экаунты выдаются в режиме лизинга: они выделяются при поступлении запроса на обработку задания или передачу файлов и закрепляются за пользователем только на время выполнения задания, затем экаунт возвращается обратно в пул. Для обеспечения взаимно однозначного соответствия между пользователями и экаунтами применяются блокирующие файлы. Поскольку в Unix-системах права доступа к файлам являются производными от экаунтов, способ выделения экаунтов предотвращает возможность влияния друг на друга заданий разных пользователей.

## Заключение

Современные программные средства грида имеют развитые функциональные возможности по обеспечению работы с глобально распределенными компьютерными ресурсами, ресурсами хранения, файловыми и информационными ресурсами.

Для работы с компьютерными ресурсами в GT4 реализован основанный на стандартах набор операций удаленного доступа. Комплекс gLite поддерживает виртуализацию – автоматический выбор компьютерных ресурсов на основе планирования, а также более широкий класс заданий (сериализуемых, параллельных и др).

С целью управления памятью хранения и файлами комплексы GT4 и gLite развивают схожие наборы служб. В оба комплекса введены средства надежной передачи данных. На основе глобальной файловой системы проведена в жизнь виртуализация доступа к файлам и репликация, что дает возможность работать с файлами по присваиваемым пользователями логическим именам.

Имеются средства дистанционного управления информационными ресурсами: распределенными базами данных каталогами, и потоками данных. На основе этих средств решена задача мониторинга произвольных (в смысле стандарта WSRF) ресурсов.

Помимо вопросов интеграции ресурсов, в программных средствах решаются вопросы поддержки функционирования грида. Реализация функций безопасности – аутентификации и авторизации – выполнена на уровне контейнеров, что обеспечивает защиту всех устанавливаемых служб. Поддерживается обслуживание больших пользовательских коллективов: для этого предназначены системы хранения сертификатов и получения их по запросу. В составе gLite имеются также службы мониторинга заданий, протоколирования процесса их обработки, учета потребления ресурсов.

При всех достижениях современного этапа развития существуют проблемы, затрудняющие применение грида. Как видно из представленного обзора, комплексы GT4 и gLite имеют различающиеся по функциональным возможностям множества служб, однако совместное использование служб разных комплексов невозможно. Основная причина такого положения – несовпадение программных архитектур. Помимо этих двух крупных комплексов в исследовательских проектах создано большое число программных систем, реализующих отдельные функции грида, но их использование в контексте других средств также сопряжено с трудностями адаптации.

Поэтому наиболее актуальными представляются задачи разработки стандартов, их программной реализации и формирования на их основе развитой платформы грида. Первый существенный шаг в этом направлении – реализация стандартов Web-служб, WSRF и WS-Notification в GT4. Решая проблему интероперабельности распределенных систем, GT4 дает стабильный инструментальный аппарат для ускорения работ по созданию платформы. Однако лимитирующим фактором остается отсутствие ряда стандартов по функциональным возможностям грида, определяющим методы интеграции и обеспечение технологических свойств.

## Литература


1. В.Н.Коваленко. Комплексное программное обеспечение грида вычислительного типа. Препринт ИПМ им. М.В.Келдыша РАН, 2007, № 10, с. 1-39.
2. Condor Project Homepage. http://www.cs.wisc.edu/condor/
3. The DataGrid Project. http://eu-datagrid.web.cern.ch
4. Research & technological development for a Data TransAtlantic Grid (DataTag) http://datatag.web.cern.ch
5. Grid Physics Network (GriPhyN). http://www.griphyn.org
6. International Virtual Data Grid Laboratory (iVDGL). http://www.ivdgl.org/
7. Workload Management System User and Reference Guide, https://edms.cern.ch/file/572489/1/WMS-guide.pdf
8. GLUE Schema. http://glueschema.forge.cnaf.infn.it/
9. Job description language attributes specification for the gLite middleware. EGEE, 2006. https://edms.cern.ch/file/590869/1/EGEE-JRA1-TEC-590869-JDL-Attributes-v0-8.pdf
10. A.Roy, M.Livny, Condor and Preemptive Resume Scheduling, Published in Grid Resource Management: State of the Art and Future Trends, 2003, p. 135-144. http://www.cs.wisc.edu/condor/doc/condor_preemptive_scheduling_2003.pdf
11. Directed Acyclic Graph Manager (DAGMan). http://www.cs.wisc.edu/condor/dagman/
12. K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Tuecke. A Resource Management Architecture for Metacomputing Systems. Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, pg. 62-82, 1998.
13. Disk Pool Manager (DPM). POOL - Persistency Framework. Pool Of persistent Objects for LHC. http://lcgapp.cern.ch/project/persist
14. P.Fuhrmann. dCache, the Overview. http://www.dcache.org/manuals/dcache-whitepaper-light.pdf
15. CASTOR. http://cern.ch/castor/
16. MDS 2.2 features in the Globus Toolkit 2.2 Release, http://www.globus.org/toolkit/mds/#mds gt2
17. Cooke A, Gray AJG, Ma LS, Nutt W, Magowan J, Oevers M, Taylor P, Byrom R, Field L, Hicks S, Leake J, Soni M, Wilson A, Cordenonsi R, Cornwall L, Djaoui A, Fisher S, Podhorszki N, Coghlan B, Kenny S, O'Callaghan D, R-GMA: An information integration system for grid monitoring. Lecture Notes in Computer Science 2888: 462-481 2003.
18. B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor and R. Wolski, A Grid Monitoring Architecture, Global Grid Forum, Lemont, Illinois, U.S.A., GFD.I.7, January 2002. http://www.ggf.org/documents/final.htm
19. LB Service User's Guide. https://edms.cern.ch/document/571273/
20. C. Anglano, S. Barale, L. Gaido, A. Guarise, S. Lusso, and A. Werbrouck. An Accounting System for the DataGrid Project: Prelimiary Proposal v3.1, 2003. http://www.to.infn.it/grid/accounting/Current prop.pdf
21. Инфраструктура публичных ключей. http://www.ietf.org/html.charters/pkixcharter.html
22. S. Farrel and R. Housley, An Internet Attribute Certificate Profile for Authorization, RFC3281 (2002)
23. A. Frohner, V. Ciaschini, VOMS Credential Format, EDG Draft, 2004
24. R.Alfieri, R.Cecchini, V.Ciaschini, L.dell'Angelo, A.Gianoli, F.Spataro, F.Bonnassieux, P.Broadfoot, G.Lowe, L.Cornwall, J.Jensen, D.Kelsey, A.Frohner, D.L.Groep, W.Som de Cerff, M.Steenbakkers, G.Venekamp, D.Kouril, A.McNab, O.Mulmo, M.Silander, J.Hahkala, K.Lorentey. Managing Dynamic User Communities in a Grid of Autonomous Resources, Computing in High Energy and Nuclear Phisics, La Jolla, California, 24-28 March 2003.
25. The gridmapdir patch for Globus: http://www.gridpp.ac.uk/gridmapdir/


**Коваленко Виктор Николаевич**. Заведующий сектором Института прикладной математики им. М.В. Келдыша РАН. Окончил Московский физико-технический институт в 1973 г. Кандидат физико-математических наук. Автор более 60 научных работ. Область научных интересов: распределенные системы, технологии грида. E-mail: kvn@keldysh.ru.