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

**В.Н.КОВАЛЕНКО, Е.И.КОВАЛЕНКО,
О.Н.ШОРИН**

РАЗРАБОТКА ДИСПЕТЧЕРА ЗАДАНИЙ ГРИД, ОСНОВАННОГО НА ОПЕРЕЖАЮЩЕМ ПЛАНИРОВАНИИ

**Москва
2005 г.**

УДК 519.68

В.Н.Коваленко, Е.И.Коваленко, О.Н.Шорин. **Разработка диспетчера заданий грид, основанного на опережающем планировании.**

В работе рассматривается реализация программного комплекса GrAS (Grid Advanced Scheduler), который выполняет задачу диспетчеризации заданий в грид. Особенностью GrAS является использование механизмов локального прогнозирования и опережающего планирования. С помощью прогноза загрузки ресурсов можно построить план эффективного распределения ресурсов таким образом, чтобы избежать перегрузки одних ресурсов во время простоя других. Опережающее планирование позволяет осуществить заблаговременную доставку входных файлов задания на освобождающийся вычислительный ресурс, что приводит к минимизации простоя ресурсов. Программная реализация GrAS поддерживает полный цикл обработки заданий и выполнена на основе Globus Toolkit 3.

Ключевые слова: грид, распределенный компьютинг, диспетчеризация, планирование, резервирование ресурсов, кластер, качество обслуживания.

V.N. Kovalenko, E.I. Kovalenko, O.N. Shorin. **Development of grid job dispatcher based on lookahead scheduling.**

In this paper an implementation of the program complex GrAS (Grid Advanced Scheduler) for dispatching grid jobs is described. The main feature of GrAS is the usage of a Lookahead scheduling method. This method has the ability to build up a plan of resource allocations for some period of the time in the future. Because the plan relies on the accurate prediction of resource states the distribution of jobs may be done efficiently to avoid resource overloading. Also lookahead scheduling further to stage in job files in advance thereby minimizing idle time of resources. Program implementation of GrAS supports full cycle of job processing and is built on the basis of Globus Toolkit 3.

Key words: grid, distributed computing, dispatching, scheduling, resource reservation, cluster, quality of service

---

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект 04-07-90299-в).

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ

В последнее десятилетие активно развивается новая форма организации вычислительных средств, получившая название грид [1]. Эта организация основывается на том, что современные телекоммуникационная аппаратура и каналы связи дают возможность интегрировать разнесенные на любые расстояния компьютерные установки разных типов и назначений. Конечно, для работы в глобальной сети существуют и активно используются протоколы и построенные на их основе программные средства, обеспечивающие передачу данных и возможность работы на удаленных машинах. Однако, в отличие от Интернет, грид призван обеспечить прозрачный доступ в рамках динамической глобальной среды как к данным, так и к вычислительным ресурсам.

Идея объединения компьютеров имеет давнюю историю. Начало положили стандарты и протоколы, позволяющие создавать локальные сети из нескольких компьютеров. Через некоторое время появилась возможность связать множество локальных сетей в одну глобальную – интернет.

В рамках локальных сетей были разработаны программные средства, позволяющие использовать суммарные вычислительные мощности машин, принадлежащих одному административному домену. Естественным продолжением развития информационных систем является перенос возможности утилизации вычислительных мощностей удаленных компьютеров с локального уровня на глобальный.

Так зародилась идея новой формы организации вычислительных средств, впоследствии получившая название грид, которая позволяет унифицированным образом объединять различные виды ресурсов в рамках динамически организующейся глобальной среды.

Для обеспечения практической применимости грид должна быть решена проблема обеспечения качества обслуживания (QoS – quality of service) пользователей [2]. Качество обслуживания – многоаспектное понятие, включающее: безопасность участвующих в грид ресурсов и безопасность выполняющихся заданий, надежность и постоянную доступность ресурсов. В конечном счете качество обслуживания должно обеспечивать приемлемое и предсказуемое время выполнения заданий.

В грид одновременно и независимо друг от друга работают множество пользователей, в то время как базовый слой грид [3] образуют протоколы удаленного доступа к ресурсам, которые поддерживают лишь индивидуальные операции отдельных пользователей. Результатом независимой работы пользователей будут локальные перегрузки одних ресурсов, при простое других.

В условиях коллективного характера функционирования грид необходимым механизмом обеспечения качества обслуживания является **планирование**, которое, выполняясь в контексте **диспетчеризации**, то есть

автоматического распределения ресурсов при обслуживании запросов, осуществляет координацию разделения ресурсов между заданиями пользователей.

В данной работе рассматривается программный комплекс GrAS (Grid Advanced Scheduler) [4] – разработанный в ИПМ им. М.В. Келдыша и реализующий централизованный подход к диспетчеризации заданий в грид с использованием локального прогнозирования. На каждом вычислительном ресурсе, находящимся под управлением GrAS, установлена специальная компонента, позволяющая получать прогноз занятия и освобождения ресурсов. На основании множества прогнозов планировщик в GrAS распределяет задания по ресурсам. GrAS применяет механизм опережающего планирования, использование которого позволяет осуществить заблаговременную доставку файлов на освобождающийся вычислительный ресурс.

В разделе 2 вводятся основные понятия, используемые в грид, описывается схема взаимодействия различных его участников. В разделе 3 дается краткий обзор существующих форм диспетчеризации, которые могут быть применимы в грид. Рассматриваются принципы функционирования, а также отмечаются преимущества и недостатки каждой из них. В разделе 4 описываются основные подходы, примененные при разработке системы диспетчеризации GrAS. В частности объясняется принцип опережающего планирования и обосновывается необходимость его применения. Пятый раздел посвящен описанию программной реализации GrAS, его отдельных компонент, а также применяемого алгоритма планирования. В 6-ом разделе приводятся математические выкладки, позволяющие оценить степень масштабируемости разработанной системы.

2. ОСНОВНЫЕ ПОНЯТИЯ

Определим **грид** как среду, которая объединяет множество пространственно распределенных **ресурсов** и поддерживает возможность использования любой их совокупности, независимо от места их расположения, для выполнения на них широкого класса приложений.

Грид является средой коллективного компьютинга, в которой каждый ресурс имеет **владельца** или **поставщика ресурсов**, а доступ к ресурсам открыт некоторому множеству **пользователей** или **потребителей ресурсов** в разделяемом по времени и по пространству режиме. Владельцы и пользователи, действующие на основании определенных правил предоставления/потребления ресурсов – **политики использования ресурсов**, образуют **виртуальную организацию** [5] грид. Отметим, что члены виртуальной организации могут совмещать роли владельца и потребителя. Виртуальная организация может образовываться динамически и иметь ограниченное время существования.

В рамках грид происходит выполнение **заданий** пользователей. Задание состоит из **исполняемых** файлов, а также **входных** и **выходных** файлов данных. Для выполнения задания требуются **ресурсы**. В первую очередь – это **вычислительные ресурсы**, на которых, собственно, происходит его исполнение. Примерами вычислительных ресурсов могут служить процессоры, оперативная и дисковая память.

Файлы задания располагаются на **ресурсах хранения** (домашняя директория пользователя, серверы хранения данных и программных кодов), как правило, отличных от вычислительных ресурсов, на которых происходит выполнение задания. До начала выполнения часть файлов задания должна быть доставлена на исполнительные ресурсы, другая часть (результаты) после окончания задания должна быть перемещена с них, а к остальным файлам должен быть гарантирован удаленный доступ посредством соответствующих сервисов или протоколов грид. Во всех этих случаях для перемещения файлов требуются **сетевые ресурсы**.

Существующие операции позволяют выполнять задания грид удаленно, но ПО грид, реализующее их, не имеет прямого доступа непосредственно к вычислительным ресурсам. Грид образуется из компьютерных установок, которые с помощью кластерного программного обеспечения объединяются на основе локальной сети в **вычислительные комплексы** (ВК). Согласно принципу **автономии ресурсов** [6], даже если ВК включаются в грид, они не отчуждаются от владельцев и используются для обработки внутренних (или **локальных**) заданий. Участвуя в грид, помимо локальных ВК обрабатывает **глобальные** задания. Архитектурное требование грид состоит в том, чтобы процесс обработки заданий в ВК контролировался **локальным менеджером управления ресурсами** (ЛМУР). ЛМУР обрабатывает задания в пакетном режиме, поддерживая **локальную очередь** заданий и распределяя их по ресурсам в соответствии с конфигурируемой политикой планирования. Именно поэтому все операции управления заданиями грид реализуются через посредство ЛМУР.

Задача службы **диспетчеризации**, заключающаяся в автоматической обработке множества заданий, включает распределение заданий по доступным ресурсам – **планирование**, доставку необходимых входных файлов на ресурс, запуск, управление и мониторинг выполнения задания, доставку выходных файлов. Система диспетчеризации в нашем подходе имеет очередь заданий грид, в которой задания буферизуются перед запуском на ресурс. В отличие от локальной очереди заданий ЛМУР, очередь системы диспетчеризации называется **глобальной.**

3. ЗАДАЧА ДИСПЕТЧЕРИЗАЦИИ

При запуске задания диспетчер выполняет следующее:

- Выбирает из множества ВК один, подходящий для запуска задания.
- Выполняет доставку входных данных и исполняемых файлов на ВК.
- Передает задание на выполнение, обеспечивая мониторинг.
- Доставляет выходные файлы на ресурсы хранения.

Первый пункт в англоязычной литературе имеет название resource discovery (обнаружение ресурсов) или scheduling (планирование). Он заключается в выборе множества вычислительных ресурсов по некоторому критерию из пула доступных и определению того ресурса из выбранного подмножества, который наиболее подходит для задания.

В некоторых публикациях [7, 8, 9] планирование делится на две независимые части: первая из них определяет ВК, на который задание должно быть доставлено, вторая – момент времени, в который это должно произойти. Тем самым, планирование происходит независимо по двум измерениям: по пространству вычислительных ресурсов (by space) и по времени (by time). В таком случае планирование по пространству вычислительных ресурсов осуществляется во время выполнения первого этапа диспетчеризации, а планирование по времени происходит непосредственно в ЛМУР.

В рамках данной работы деления планирования на составные части проводиться не будет, а будет подразумеваться, что планирование – это единый неделимый процесс сопоставления множества заданий множеству вычислительных ресурсов, в результате которого для каждого задания определяется 1) ВК для запуска и 2) время запуска задания. Подобное соотношение назовем **расписанием**. Итак, планирование – это процесс получения расписания.

Способ получения расписания основывается на политике планирования – наборе правил, ограничений, соглашений, принятых всеми участниками грид: потребителями и поставщиками ресурсов. Планирование осуществляется программной компонентой - планировщиком. Основной целью планировщика является создание расписания, которое бы учитывало интересы всех участников в грид. В то время как поставщики ресурсов заинтересованы в увеличении утилизации своих ресурсов, потребителям важна минимизация времени выполнения задания, что не одно и то же.

3.1. Формы диспетчеризации

Для диспетчеризации заданий в грид используются различные подходы. Далее будет приведен обзор механизмов диспетчеризации и описаны основные принципы, используемые для управления заданиями в грид.

Централизованная диспетчеризация. При использовании централизованного подхода все задания распределяются на ресурсы одной выделенной службой – диспетчером. При этом все задания поступают диспетчеру. Те из поступающих диспетчеру заданий, которые не могут быть

запущены непосредственно в момент поступления, хранятся на диспетчере в пуле заданий. Информация о текущем состоянии всех ресурсов также собирается в информационной базе диспетчера.

Централизованный диспетчер может оказаться узким местом в целом ряде ситуаций: например, в случае сбоя диспетчера данная архитектура перестанет быть работоспособной. С другой стороны, обладая полной информацией, диспетчер может осуществлять эффективное планирование.

Централизованная диспетчеризация применяется в системах [10, 11, 12, 13, 14].

Иерархическая структура диспетчеризации. Возможны конфигурации, когда один централизованный диспетчер оперирует диспетчерами более низкого уровня. Такая структура обладает как свойствами централизованной системы, так и децентрализованной.

Основным преимуществом использования иерархической структуры диспетчеризации является возможность применения различных политик на центральном диспетчере и на подчиненных ему диспетчерах. В таком случае центральный диспетчер выступает в роли метадиспетчера, распределяющего поступающие к нему задания по диспетчерам более низкого уровня в соответствии с некоторым набором правил.

Децентрализованная схема диспетчеризации. При использовании децентрализованной схемы распределенные диспетчеры взаимодействуют друг с другом и отправляют задания непосредственно на вычислительные ресурсы, а не на диспетчеры более низкого уровня. Не существует какого-либо центрального диспетчера, осуществляющего распределение заданий. Также, информация о текущем состоянии ресурсов не группируется в едином месте. Таким образом, проблема масштабирования, присущая централизованной схеме, отсутствует в децентрализованной, а отказ в работе одного из компонентов никоим образом не сказывается на работе системы в целом.

Из-за наличия множества равноправных диспетчеров возможны различные схемы их взаимодействия. В рамках децентрализованной схемы выделяют [15] два основных механизма: непосредственное взаимодействие и с помощью общего пула.

При непосредственном взаимодействии диспетчеры посылают и принимают задания друг от друга. Каждый диспетчер имеет список диспетчеров, с которыми он может обмениваться заданиями, или же существует общедоступная информация обо всех диспетчерах.

Если диспетчер не может в данный момент осуществить запуск задания, он начинает процесс поиска альтернативного диспетчера. В случае успешного нахождения диспетчера, который может запустить задание на выполнение, само задание и все необходимые входные данные пересылаются найденному диспетчеру.

При взаимодействии с использованием общего пула, задания, которые не могут быть запущены на счет, пересылаются в центральный пул заданий, а не другому диспетчеру. В отличие от механизма непосредственного взаимодействия диспетчер выбирает из общего пула задания, для которых имеются свободные вычислительные ресурсы. Однако данный подход требует существования политики, предотвращающей зависания больших заданий в общем пуле.

Механизм взаимодействия с помощью общего пула заданий обычно модифицируют следующим образом: все новые задания поступают сразу же в пул, а не какому-нибудь диспетчеру, откуда впоследствии и выбираются диспетчерами для запуска.

При использовании децентрализованного подхода проблема масштабируемости не возникает. Однако отсутствие полной информации обо всех ресурсах приводит к неэффективному планированию. Также с помощью децентрализованного подхода сложно обеспечить запуск параллельных заданий, части которых выполняются на разных ресурсах, поскольку это требует согласованной работы различных диспетчеров для синхронного запуска частей задания.

Децентрализованный подход применяется, например, в системе PAUA [16].

4. ДИСПЕТЧЕРИЗАЦИЯ РЕСУРСОВ В ГРИД, ОСНОВАННАЯ НА ЛОКАЛЬНОМ ПРОГНОЗИРОВАНИИ

В данной главе будет описана разработанная система диспетчеризации, основанная на локальном прогнозировании состояния ресурсов. Будут приведены объяснения тех решений, которые были приняты при проектировании данной системы. Особое внимание будет уделено алгоритму планирования, используемому в рамках разработанной системы.

4.1. Постановка задачи диспетчеризации

GrAS представляет собой программный комплекс диспетчеризации заданий в грид. Он осуществляет прием заданий от пользователей, распределяет их по ресурсам, производит запуск и обеспечивает мониторинг их выполнения. GrAS базируется на ряде принципов, которые обуславливают условия его применения.

4.1.1. Организация ресурсов

Грид образован из кластеризованных ресурсов. Множество ресурсов, находящихся под управлением системы диспетчеризации, являются не единичными узлами, а представляют собой кластеры, содержащие множество

компьютеров. Для эффективного распределения заданий система должна учитывать внутренние строения кластеров и динамически реагировать на изменение состава и состояния ресурсов внутри кластеров.

Грид состоит из неотчуждаемых ресурсов. Помимо заданий, поступающих от системы диспетчеризации, - глобальных заданий, на ресурсы поступают потоки заданий владельцев ресурсов – локальные задания. Администратор каждого ресурса должен иметь возможность изменять политику предоставления своего ресурса глобальным заданиям в зависимости от множества факторов: параметров претендующего на ресурс глобального задания, времени суток, количества локальных заданий.

Неделимость ресурсов. Количество заданий, одновременно выполняющихся на одном компьютере, не может быть больше количества процессоров данного компьютера.

Однопроцессорные задания. Все задания, локальные и глобальные, являются однопроцессорными заданиями и не зависят друг от друга.

4.1.2. Требования к реализации GrAS

Globus Toolkit и аппарат грид-служб. Система диспетчеризации реализуется с помощью инструментальной среды Globus Toolkit [3], являющейся стандартом де-факто в области создания грид-систем. Globus Toolkit позволит использовать систему диспетчеризации в имеющихся грид-инфраструктурах.

Модульность системы диспетчеризации. GrAS должен поддерживать возможность замены одного алгоритма планирования другим без существенных модификаций.

4.1.3. Способ использования GrAS

Интерфейс пользователя аналогичен Globus Toolkit. В инструментарии Globus Toolkit имеется пользовательский интерфейс, с помощью которого задания посылаются на счет, осуществляется управление заданиями. GrAS должен иметь пользовательский интерфейс идентичный интерфейсу Globus Toolkit. Это поможет пользователям избежать изучения нового протокола работы с грид.

Качество обслуживания. Пользователь должен иметь возможность узнать подробную информацию о состоянии задания. Также пользователю должны быть предоставлены механизмы, с помощью которых он может повлиять на то как быстро будут выполнены его задания. В частности, он может ускорить процесс запуска, т.е. уменьшить время старта, части своих заданий за счет увеличения времени старта другой части.

4.2. Подходы, используемые в системе диспетчеризации

Руководствуясь приведенными выше ограничениями, разработана система диспетчеризации, основанная на следующих принципах:

Централизованная диспетчеризация. Система диспетчеризации является централизованной, т.е. предполагается наличие единственной службы, которая, не взаимодействуя со службами аналогичного назначения, выполняет распределение поступающих к ней заданий по ресурсам. Централизованная схема имеет ограниченные возможности масштабирования, и область ее применения – виртуальная организация. Представляется, что в ограниченных масштабах централизованная схема дает более широкие возможности и может служить основой для построения схемы взаимодействующих служб диспетчеризации.

Доставка задания на вычислительный ресурс осуществляется непосредственно перед запланированным запуском. Большинство существующих централизованных систем диспетчеризации осуществляют запуск задания на ВК сразу же после его распределения [10, 12, 13, 14], не учитывая при этом время запуска задания на вычислительном ресурсе. Задание попадает в локальную очередь ЛМУР и находится в режиме ожидания. Однако, во время ожидания может освободиться другой вычислительный ресурс, способный выполнить данное задание. Или же в локальную очередь могут поступить высокоприоритетные задания, вследствие чего запуск глобального задания будет отложен на неопределенный срок. Использовать механизм перераспределения в таких случаях представляется нецелесообразным, поскольку это ведет к лавинообразному росту пересылок заданий с одних ВК на другие.

Выходом из положения является включение в состав системы диспетчеризации специальной компоненты - очереди заданий. В этой очереди задания хранятся, а пересылка их в ВК осуществляется незадолго до запланированного времени старта.

Упреждающее распределение. Доставка заданий на ВК непосредственно в момент освобождения ресурса невозможен, поскольку доставка необходимых файлов занимает много времени. Освобожденный вычислительный ресурс может быть занят другим заданием в тот момент, когда осуществляется доставка входных файлов для запланированного задания. Таким образом, пересылка должна осуществляться с неким упреждением, достаточным для доставки необходимых входных файлов. Упреждение основывается на механизме прогнозирования освобождения/занятия ресурсов.

Приоритизация заданий. Пользователь должен иметь возможность влиять на скорость запуска задания. Для обеспечения этой возможности каждое задание в очереди имеет приоритет, а очередность распределения заданий по ресурсам производится в зависимости от приоритетов. Пользователю

предоставлен механизм, позволяющий повышать или понижать приоритеты своих заданий, влияя тем самым на скорость распределения. Для избежания монополизации всех ресурсов каждому пользователю выделяется бюджет, в рамках которого он может назначать плату за выполнение заданий. В этом случае, плата выступает в роли приоритетов.

Пересчет приоритетов. Задания в глобальной очереди обрабатываются в соответствие с приоритетами. В локальной очереди также используется аппарат приоритизации заданий. Поскольку каждый ресурс имеет свою метрику приоритетов, в GrAS введен механизм пересчета приоритетов глобальных заданий в локальные приоритеты. Этот механизм учитывает атрибуты глобального задания и параметры конкретного ресурса. Реализации этого механизма основана на том, что администратор каждого ресурса предоставляет функцию пересчета глобального приоритета в локальный. Подобная функция имеет двойственную функциональность: с одной стороны она используется для унификации метрик приоритетов в рамках всей инфраструктуры, с другой, с ее помощью владелец ресурсов может управлять интенсивностью потока глобальных заданий на свои вычислительные ресурсы. Таким образом, механизм пересчета приоритетов выступает в роли средства проведения в жизнь администраторами ресурсов политик предоставления своих ресурсов в грид.

Соглашения, принятые в системе диспетчеризации. Диспетчер должен иметь возможность получить информацию о возможности запуска глобального задания на ресурсе, не обращаясь к ЛМУР, поскольку подход, основанный на опросе ЛМУР, является ресурсоемким в плане использования сети и времени. Вывод о возможности запуска задания можно сделать, пересчитав глобальный приоритет в локальный и сравнив полученный результат с приоритетами заданий, находящихся в локальной очереди. Таким образом диспетчер может определить, как будет выглядеть локальная очередь, если данное глобальное задание будет послано на этот ВК.

5. РЕАЛИЗАЦИЯ GRAS

На основе вышеперечисленных подходов реализован программный комплекс GrAS. GrAS создан с использованием инструментальной среды Globus Toolkit версии 3.2.1. Коммуникации распределенных компонентов друг с другом реализованы с помощью аппарата грид-служб – стандартным средством дистанционного взаимодействия в грид.

5.1. Структурная схема GrAS

GrAS состоит из трех основных компонент:

- интерфейса пользователя, который позволяет отправить задание в GrAS, узнать статус уже отправленного задания, изменить его параметры, отменить его и получить результаты;
- кластерного Агента, который составляет прогноз занятия/освобождения ресурсов в кластере;
- а также серверной части GrAS, которая осуществляет прием сообщений от Агентов и пользователей, распределение заданий по ресурсам и их запуск.

Серверная часть GrAS в свою очередь содержит в себе:

- ядро, которое обрабатывает поступающие события и осуществляет планирование;
- утилиту запуска заданий в кластер и управления запущенными заданиями – Job Control;
- грид-службы, принимающие сообщения от Агентов и пользователей;
- базу данных, в которой хранится очередь заданий и информация о ресурсах;
- очередь сообщений.

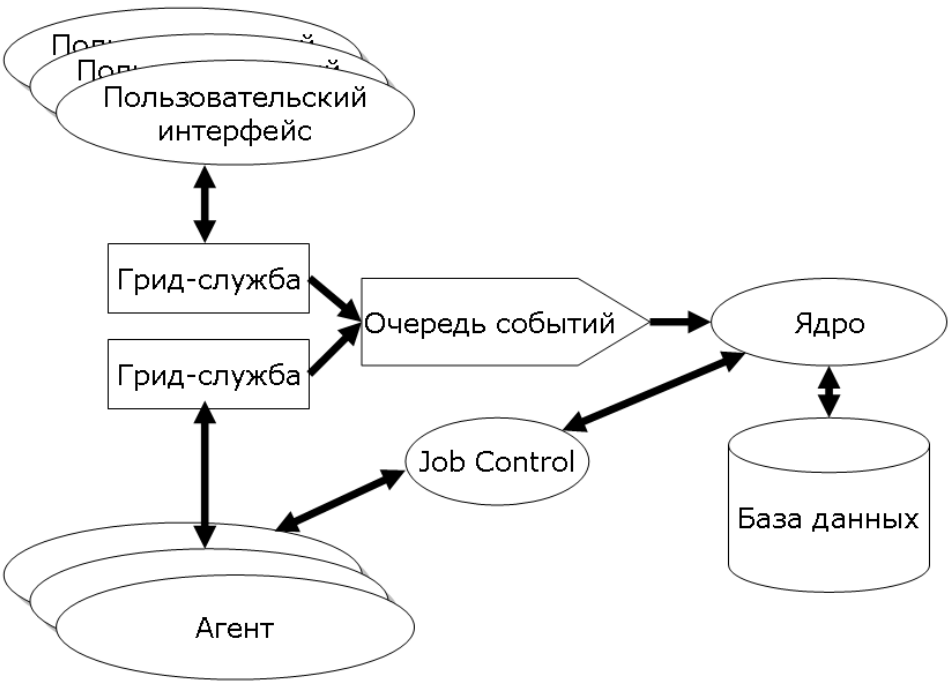

Рисунок 1. Структурная схема GrAS

5.1.1. Пользовательский интерфейс

Для реализации пользовательского интерфейса разработан набор программных интерфейсов (API) на языке Java, позволяющих отсылать запросы грид-службе диспетчера. На основе этих интерфейсов создана клиентская утилита, которая в зависимости от аргументов выполняет те или иные операции.

Простейшей операцией пользовательского интерфейса является ping. Для ее выполнения, равно как и для любой другой операции, необходимо указать адрес (URL – Uniform Resource Locator) грид-службы, принимающей запросы от пользователей. Операция ping посылает пробный запрос и завершается успешно, если грид-служба ответила на этот запрос. Данная операция является диагностической. С ее помощью можно проверить работоспособность сети, грид-службы, а также годность используемого пользователем сертификата безопасности.

Для запуска заданий в состав операций пользовательского интерфейса входит submit. Помимо адреса грид-службы для этой операции необходимо указать файл, содержащий описание задания на языке RSL. В нем содержится информация о расположении входных данных задания, о количестве ресурсов, необходимых для успешного выполнения, а также сведения о том, куда доставить выходные файлы. Сначала проверяется синтаксис файла описания, а потом он пересылается грид-службе. При обработке запроса грид-служба формирует уникальный идентификатор, возвращает его клиентской утилите и помещает описание задания в очередь событий. С целью минимизации количества пересылок, в GrAS введено ограничение: входные файлы задания должны быть доступны посредством механизма удаленного доступа к внешней памяти. Благодаря такому подходу нет необходимости доставлять входные файлы на сервер диспетчера. Когда задание будет отправлено на счет на какой-нибудь ВК, входные файлы будут доставлены на этот вычислительный ресурс непосредственно с места хранения, минуя диспетчер.

Пользователь может узнать информацию о своем задании с помощью операций status и log. Для этих операций необходимо указать уникальный идентификатор, полученный во время операции submit. Status возвращает пользователю текущее состояние задания, а операция log – подробный отчет с указанием всех действий, которые были произведены GrAS с заданием.

Отменить ранее запущенное задание пользователь может с использованием операции cancel. Для этой операции также необходимо указать идентификатор задания. Изменить же параметры запущенного задания, например, приоритет можно, применяя операцию refresh. Для операции refresh задается идентификатор изменяемого задания и файл с новым RSL описанием. Утилита проверяет синтаксис RSL файла и отсылает его грид-службе, которая помещает событие об изменении параметров задания в очередь.

Также предусмотрена команда list, которая выдает идентификаторы всех заданий, запущенных данным пользователем.

5.1.2. Схема работы Агента

Процесс планирования в системе диспетчеризации строит расписание для глобальных заданий – **глобальное расписание** (ГР). ГР содержит время старта и отводимые для каждого задания ресурсы. Для построения ГР диспетчер должен иметь информацию о возможности занятия ресурсов глобальными заданиями. Не обладая непосредственным контролем над ресурсами, диспетчер может получить эту информацию, только взаимодействуя с ЛМУР.

ГР строится на основе множества **локальных расписаний** (ЛР) различных ВК. Локальное расписание представляет собой прогноз занятия/освобождения ресурсов на некотором ВК при заданном состоянии локальных очередей. ЛР содержит информацию как о заданиях, которые выполняются на данный момент, так и информацию о том, когда они закончатся, и какие задания будут запущены после них. В периоды выполнения заданий состояние ресурса не изменяется, поэтому всю информацию о состоянии ресурса можно разделить на части – **слоты**, в рамках которых не происходит изменений.

Программно функция составления ЛР реализована как специальный режим работы локального планировщика, в котором сохраняется основной алгоритм планирования и имитируется:

- запуск/окончание счета заданий, причем время счета берется из ресурсного запроса;
- взаимодействие с ресурсами;
- доступ к заданиям.

Реализация этого режима основывается на функции моделирования планировщика Maui [17]. Появление этой функции в Maui не было связано с грид, а было вызвано задачей оценки качества планирования и настройки оптимальных конфигурационных параметров планировщика. То, что моделирование появилось именно в Maui, вполне естественно из-за свойств применяемого в Maui алгоритма планирования BackFill [18]. Однако, составление ЛР может быть реализовано и для любых других алгоритмов планировщиков систем пакетной обработки.

Агент GrAS создан на базе системы управления пакетной обработкой OpenPBS с внешним планировщиком Maui. Выбор системы OpenPBS связан, главным образом, с ее большей популярностью среди ЛМУР. Агент будет работать с любыми системами, которые совместимы с Maui. Замена штатного планировщика OpenPBS на Maui существенна, так как его расширенные возможности обеспечивают реализацию основных функций Агента.

Среди многих достоинств Maui важно то, что это открытый продукт. Кроме того, наличие специального режима SIMULATION позволяет моделировать

процесс размещения заданий в кластере в будущем времени, обеспечивая механизм генерации локальных расписаний. И, наконец, Maui поддерживает аппарат предварительного резервирования, т.е. позволяет зарезервировать локальные ресурсы для гарантированного запуска глобальных заданий.

Для сбора информации о прогнозируемом состоянии ресурса и построения расписаний Агент взаимодействует с локальным планировщиком ВК. Локальный планировщик принимает решение о запуске всех заданий в кластере, поэтому именно его данные о состоянии ресурсов существенны для построения прогноза. Следует отметить, что такая информация не всегда совпадает с теми данными, которые можно получить прямо от ОС машины или от других объектов ЛМУР.

ЛР передаются серверной части диспетчера в те моменты, когда они обновляются. Поэтому Агент отслеживает события, приводящие к изменению ЛР, и инициирует передачу новой информации.

Для работы Агенту необходимо иметь доступ к данным, размещенным в информационных структурах Maui. К сожалению, в настоящее время нет готового API для доступа к такой информации, поэтому Агент реализован путем модификации Maui версии 307p6. В результате был получен модифицированный планировщик, который кроме своих действий выполняет и ряд функций Агента по информационному обслуживанию.

В каждом цикле работы модифицированного Maui выполняется проверка наличия событий, способных повлиять на ЛР. Если такое событие происходит, новая информация о статических и динамических параметрах кластерных объектов выводится в файлы, которые далее используются для построения локального расписания и пересылки его ядру GrAS транспортными службами Агента. Агент формирует два выходных файла: файл статических ресурсов и файл расписания. Статические ресурсы обновляются при появлении нового узла в кластере, изменении состояния уже существующих узлов (рабочее–нерабочее) или при изменении конфигурационного файла Maui. Файл расписания обновляется при каждом поступлении нового задания в кластер или при старте какого-либо задания из очереди.

Если полная информация об уже работающих заданиях присутствует в информационных структурах Maui, то для заданий, на данный момент только ожидающих запуска, информации о том, где и когда они будут запущены, нет. Она может быть получена в результате моделирования процесса обработки кластерной очереди. Для этого используются возможности Maui работать в режиме моделирования (SIMULATION). При необходимости построения расписания запускается новый процесс - Симулятор Агента (модифицированный Maui в режиме SIMULATION). В результате его работы формируется прогноз о будущих запусках.

Поскольку рассматривается планирование для однопроцессорных заданий и компьютеров, планировщику достаточно ограниченного ЛР. В настоящее

время Агент строит расписание, только для двух последовательных заданий на каждом кластерном узле (уже запущенного и следующего за ним - прогнозируемого).

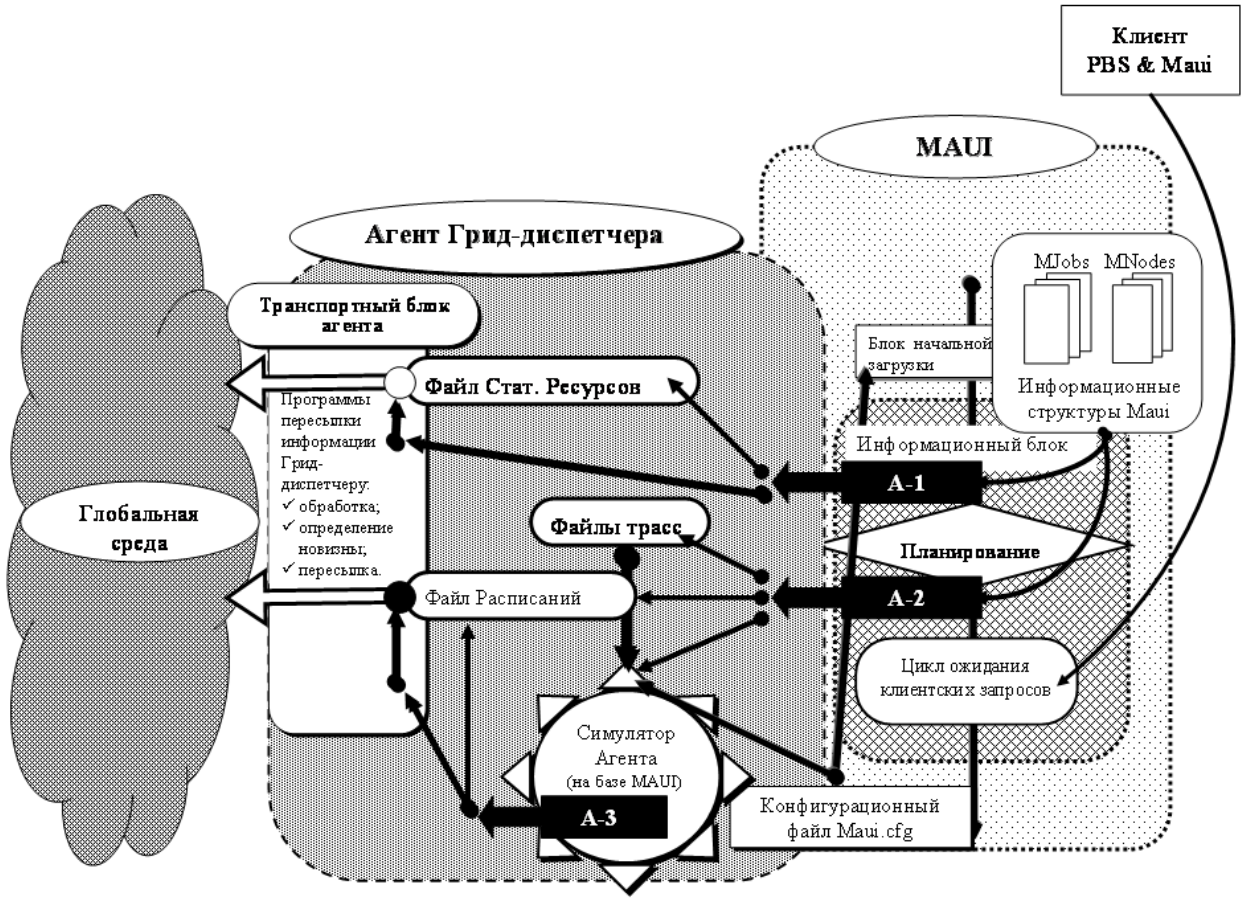

Рисунок 2. Схема взаимодействий между компонентами Maui и Агента GrAS

После создания ЛР происходит запуск транспортных служб Агента. Программы этого блока проверяют содержимое файлов статических ресурсов и расписаний и готовят информацию для транспортировки. При наличии новых данных посредством грид-служб происходит их пересылка ядру GrAS.

Таким образом, Агент, установленный в каждой локальной кластерной системе, управляемой GrAS, решает следующие задачи:

- обеспечивает ядро GrAS информацией о состоянии локальных очередей планировщика;
- фиксирует события, способные изменить ЛР;
- формирует локальное расписание;
- организует доставку нового ЛР ядру диспетчера.

5.1.3. Реализация очереди сообщений

Очередь предназначена для буферизации сообщений, приходящих от Агентов и пользователей, перед тем, как они будут обработаны ядром GrAS. Буферизация сообщений в очереди позволяет работать пользователям и Агентам с диспетчером в асинхронном режиме. Информация в очереди хранится в порядке поступления, однако поддерживается механизм, позволяющий выбрать из любого места очереди сообщения определенного типа. Все содержимое очереди физически хранится в базе данных.

Для работы с очередью сообщений разработаны программные интерфейсы (API) для двух языков программирования: C и Java. Это связано с тем, что грид-службы, принимающие запросы от пользователей и Агентов, написаны на языке Java, а ядро GrAS, выбирающее сообщения из очереди, на C.

5.1.4. Структура базы данных

В базе данных GrAS содержится информация, имеющаяся в распоряжении системы диспетчеризации. В качестве СУБД используется система PostgreSQL. Выбор в пользу данной СУБД был сделан по нескольким причинам. Во-первых, это открытость PostgreSQL, а во-вторых, поддержка механизма хранимых процедур, позволяющих повысить скорость работы с базой данных на несколько порядков. К тому же PostgreSQL является реляционной базой данных, что соотносится с характером хранящейся в ней информации.

База данных GrAS содержит несколько таблиц.

Cluster. В данной таблице находится общая информация обо всех кластерах, участвующих в виртуальной организации GrAS. Основным содержанием данной таблицы являются URL грид-службы для запуска заданий в кластер, функция пересчета глобального приоритета задания в локальный, тип ЛМУР. Таблица выглядит следующим образом:

Название поля	Тип	Содержимое
id_cluster	text	Уникальный идентификатор кластера
URL	text	URL грид-службы для запуска заданий в кластер
LRM	text	Тип ЛМУР
Params	text	Функция пересчета глобального приоритета в локальный

Node. Данная таблица предназначена для хранения описания вычислительных узлов, входящих в кластеры. В ней содержится информация о статических параметрах узлов: количество процессоров и их производительность, объем оперативной памяти, жесткого диска и файла подкачки, архитектура, операционная система.

Название поля	Тип	Содержимое
Id_cluster	text	Уникальный идентификатор кластера
Id_node	text	Уникальный идентификатор узла в кластере
ResourceType	text	Тип ресурса
NodeState	text	Состояние узла
Swap	decimal	Объем файла подкачки
Mem	decimal	Объем оперативной памяти
Disk	decimal	Объем свободного места на жестком диске
Procs	decimal	Количество процессоров
Opsys	text	Операционная система узла
Arch	text	Архитектура узла

Content. В этой таблице хранятся локальные расписания, приходящие от Агентов. Каждая запись таблицы содержит подробную информацию о локальном задании, выполняющемся или планируемым к выполнению на одном из узлов какого-либо кластера. Основной является информация о том, на каком узле выполняется задание, и параметры задания: время старта, длина и приоритет задания.

Название поля	Тип	Содержимое
id_cluster	text	Уникальный идентификатор кластера
id_node	text	Уникальный идентификатор узла в кластере
id_task	text	Идентификатор задания
UserName	text	Пользователь, от имени которого запущено задание
GroupName	text	Группа, к которой принадлежит пользователь, запустивший задание
JobState	text	Состояние задания
QueueTime	decimal	Время поступления задания в локальную очередь
StartTime	decimal	Время старта задания
CompleteT	decimal	Время окончания задания
Procs	decimal	Количество процессоров, занятых данным заданием
Mem	decimal	Объем оперативной памяти, занятой данным заданием
Disk	decimal	Объем использования жесткого диска данным заданием
StartPrior	decimal	Локальный приоритет задания
Dirty	text	Поле, используемое для резервирования ресурсов

Queue. Данная таблица предназначена для хранения глобальной очереди заданий. В ней хранятся описания всех глобальных заданий, находящихся под управлением GrAS. В полях этой таблицы отображена информация о ресурсном запросе задания, текущем статусе и приоритете, соответствия задания к ресурсу и времени запуска на нем в случае, если данное задание уже

распланировано на какой-либо ресурс. Также в данной таблице хранится полная информация о ходе выполнении задания на ресурсе.

Название поля	Тип	Содержимое
id_global	text	Глобальный идентификатор задания
id_local	text	Идентификатор задания, полученный при запуске на ресурс
status	text	Статус задания
RSL	text	Ресурсный запрос задания
dn_user	text	Имя пользователя задания
submit_date	decimal	Время запуска задания на счет
priority	decimal	Глобальный приоритет задания
memory	decimal	Объем оперативной памяти, необходимый заданию
processor	decimal	Количество процессоров, необходимых заданию
Arch	text	Архитектура узла, необходимого заданию
walltime	decimal	Время выполнения задания
Disk	decimal	Объем свободного места на жестком диске, необходимый заданию
id_cluster	text	Идентификатор кластера, на который распределено задание
local_priority	decimal	Локальный приоритет задания в кластере, если оно уже распределено
Log	text	Полный лог выполнения задания

Messages. Эта таблица предназначена для хранения сообщений, находящихся в очереди сообщений GrAS. В ней содержатся тела и типы сообщений, а также информация о длине и дате поступления сообщения в очередь.

Название поля	Тип	Содержимое
Number	decimal	Порядковый номер сообщения
Type	decimal	Тип сообщения
Size	decimal	Размер сообщения
Message	text	Тело сообщения

Users. Данная таблица содержит соответствия имеющихся ресурсов и пользователей системы диспетчеризации и служит для определения возможности использования некоторым пользователем определенного ресурса. Данная таблица заполняется в соответствии с gridmap файлами, получаемыми от Агентов.

Название поля	Тип	Содержимое
id_cluster	text	Уникальный идентификатор кластера
dn_user	text	Имя пользователя

5.1.5. Ядро диспетчера

Ядро - это компонента GrAS, в рамках которой выполняется процесс планирования. Оно реализовано в виде фонового процесса (демона), и использует механизмы, предоставляемые операционной системой, для управления демонами. Как уже было сказано выше, ядро написано на языке C и использует вызовы хранимых процедур базы данных, реализованных на языке PL/pqSQL.

Работа ядра диспетчера является циклической: во время работы цикла все поступающие сообщения буферизуются в очереди сообщений, после окончания цикла ядро выбирает сообщения из очереди и начинает новый цикл, выполняющий те или иные действия в зависимости от пришедших сообщений.

Основная задача ядра – планирование распределения заданий по ресурсам. Основываясь на информации, хранящейся в базе данных, планировщик GrAS осуществляет распределение заданий. В случае успешного распределения какого-либо задания, оно помечается в базе данных как распределенное, а ядро инициирует процесс доставки задания на ресурс.

5.1.6. Утилита запуска и управления заданиями

Утилита запуска и управления запущенными заданиями Job Control предназначена для взаимодействия GrAS с вычислительными ресурсами, на которых выполняются глобальные задания.

Job Control позволяет запустить глобальное задание на счет, отменить и/или узнать статус уже запущенного задания. Взаимодействие ядра GrAS с Job Control является асинхронным: ядро запускает утилиту Job Control, а результат работы помещается в очередь сообщений. На следующем цикле сообщение с результатами работы Job Control попадает ядру.

Job Control представляет собой утилиту, написанную на языке Java, и является клиентом грид-службы Globus Toolkit.

5.2. Алгоритм планирования

Планирование в GrAS выполняется периодически. На каждом цикле планирования обрабатывается фиксированное множество глобальных заданий и локальных расписаний. Во время работы планировщика все поступающие изменения буферизуются в очереди сообщений.

Схематично алгоритм работы планировщика можно представить следующим образом:

Цикл планирования
{
  J = Ø; /* *J – множество нераспределенных глобальных заданий* */

```
S = Ø; /* S – множество слотов локальных расписаний */

/* Выбор заданий */
По всем j Є (множеству глобальных заданий)
{
        Если j не распределено
                J += j;
}
Сортировать J по приоритетам;

/* Отбор слотов */
По всем s Є (множеству слотов)
{
        /* старт(s) – время старта слота */
        /* t – текущее время */
        /* tmin и tmax – переменные, значения которых
        будет объяснено позднее */
        Если (старт(s) >= t+tmin) и (старт(s)<=t+tmax)
                S += s;
}
Сортировать S по времени старта;

/* Попарное сопоставление */
По всем j Є J
{
        По всем s Є S
        {
                /* локальный приоритет – функция пересчета глобального
                приоритета в локальный */
                /* приоритет – значение приоритета задания */
                /* ВК – вычислительные ресурсы, на которых выполняется
                слот */
                /* владелец – владелец задания */
                /* ресурсы – ресурсы, либо необходимые для задания, либо
                занятые  слотом */
                Если (локальный приоритет(приоритет(j), ВК(s)) >
                приоритет(s))
                и (владелец(j) имеет право выполнять задания на ВК(s))
                и (ресурсы(j) ≤ ресурсы(s))
                {
                        J -= j;
                        S -= s;
```

```
                        Зарезервировать s для j;
                        Запустить j на s;
                    }
                }
            }
        }
```

Цикл планирования можно разделить на 3 больших этапа: выбор заданий, отбор слотов и попарное сопоставление.

На этапе выбора заданий из глобальной очереди выбираются те задания, которые еще не были распределены, и сортируются в порядке уменьшения приоритетов.

На втором этапе из локальных расписаний извлекается множество слотов. Критерием отбора слотов является время старта слота. Оно не должно быть слишком близким к моменту планирования. Это необходимо для того, чтобы планировщик успел произвести планирование, осуществить резервирование и доставить входные файлы. Но оно также не должно отстоять слишком далеко от текущего времени, поскольку в противном случае к моменту начала данного слота ситуация на ВК может измениться из-за прихода новых локальных заданий. Тем самым отбираются слоты, времена старта которых попадают во временной интервал [t + tmin, t + tmax], где t – это момент планирования, а tmin и tmax выбираются из тех соображений, что T<tmin<tmax<2*T, где T – время необходимое для осуществления цикла планирования, резервирования и доставки входных файлов на вычислительный ресурс. Строго говоря, в общем случае tmin и tmax зависят от множества параметров: количества отобранных слотов, времени, необходимого на доставку данных с ресурсов хранения на конкретный вычислительный ресурс. В GrAS tmin и tmax имеют предопределенные значения.

Далее начинает работать двойной цикл, в процессе которого глобальные задания сопоставляются со слотами. Если сопоставление задания со слотом завершается успешно, то данные задание и слот удаляются из дальнейшего рассмотрения и инициируется процесс резервирования ресурсов для глобального задания на месте выбранного слота.

Сопоставление задания со слотом заключается в последовательной проверке нескольких условий. Слот – это описание ресурса, на котором по прогнозу в определенный момент времени будет запущено локальное задание. Поэтому в первую очередь сравниваются приоритеты глобального задания и локального. Для этого происходит пересчет глобального приоритета задания в локальный с использованием формул, предоставленных администратором ВК.

Если пересчитанный приоритет глобального задания оказался выше приоритета локального задания, происходит проверка прав доступа владельца глобального задания на вычислительный ресурс, относящийся к рассматриваемому слоту.

Если владелец имеет право выполнять свои задания на данном вычислительном ресурсе, выполняется проверка соответствия ресурсов слота, ресурсному запросу глобального задания.

В случае удовлетворения всем 3 условиям сопоставление считается успешным.

6. МАСШТАБИРУЕМОСТЬ GRAS

GrAS, как и любая другая централизованная система, обладает ограниченными возможностями масштабирования. В связи с этим вопросы оценки максимального количества обслуживаемых узлов, а также разработка способов повышения масштабируемости являются важными.

Слоты, поступающие от Агентов в виде локальных расписаний, соответствуют отдельным событиям, происходящим в ВК. В отличие от локальных планировщиков, где производится реакция на каждое событие, GrAS обрабатывает слоты не по одному, а целыми множествами (пачками). Подобный подход имеет преимущество перед способом, при котором GrAS обрабатывает все слоты поодиночке. Данное преимущество основывается на том факте, что помимо непосредственной обработки необходимо произвести дополнительные действия, направленные на инициализацию диспетчера. Во время групповой операции подобная инициализация выполняется единожды, в отличие от обработки отдельных слотов. Тем самым, если за t_{in} обозначить время, необходимое на инициализацию планировщика, а за t_w – время обработки одного слота, то для обработки N слотов пачкой потребуется $t_{in}+N* t_w$ времени, в то время как для обработки тех же N слотов по одному требуется $N*(t_{in}+ t_w)$.

На рисунке 3 приведен, полученный в результате моделирование работы планировщика, график зависимости продолжительности цикла планирования от количества слотов, обрабатываемых пачкой, и количества глобальных заданий. Из графика видно, что зависимость получилась линейная.

Приведем некоторые цифры, иллюстрирующие эту зависимость. Продолжительность работы планировщика для 10 глобальных заданий и 1 слота составляет 5 мск, для такого же количества заданий и пачки из 10 слотов – 10 мск, для пачки из 20 слотов – 16 мск, а для обработки пачки из 1000 слотов требуется 468 мск. Т.е. для сопоставления глобальных заданий с пачкой 1000 слотов потребовалось 468 мск, для сопоставления с таким же количеством слотов порциями по 20 штук потребовалось бы 16*50=800 мск, порциями по 10 штук – 1000 мск, обработка же тысячи слотов по одному заняла бы 5000 мск. Таким образом, видно, что обработка множества слотов за один раз дает выигрыш.

Поскольку GrAS обрабатывает изменения ЛР пачками, его работу можно поделить на циклы планирования. В начале очередного цикла диспетчер

выбирает из очереди все находящиеся в ней локальные расписания и новые глобальные задания и обновляет базу данных. Приходящие за время одного цикла новые ЛР и глобальные задания буферизуются в очереди сообщений и будут использованы на следующем цикле.

Рисунок 3. Зависимость длительности цикла планирования от количества слотов, обрабатываемых пачкой, и глобальных заданий

Нарушение штатного режима происходит, когда отставание ($T_{ир}$-$T_{пр}$), где $T_{пр}$ - время производства расписания на ВК, $T_{ир}$ – время использования расписания при планировании, становится большим. Как меру допустимого отставания введем понятие **времени жизни расписания** (TTL – Time To Live): планировщик не должен использовать расписание, время жизни которого истекает ко времени конца цикла планирования.

Для каждого ВК значение TTL может устанавливаться индивидуально, исходя из смысла этого параметра: значение TTL определяет, с каким предельным отставанием планировщик должен реагировать на изменения состояния вычислительного ресурса. При большом TTL получение ресурсов для вновь поступающих локальных заданий будет иметь тенденцию к задержке: полагающиеся им ресурсы могут занять глобальные задания. Однако, задержка всегда ограничена, и, если, например, TTL устанавливается равным минимально возможной длительности задания, то локальное задание может пропустить не более одного глобального задания вне очереди, причем задерживаться будут только такие локальные задания, которые имеют самый высокий приоритет в локальной очереди и могут быть сразу запущены.

Необходимое условие правильного функционирования заключается в том, что длительность цикла планирования должна быть меньше TTL всех расписаний. В работе [19] выведена оценка, выражающая зависимость максимального числа компьютеров грид от минимальной длины задания: $N<\sqrt{L/(2*C)}$, где N – число компьютеров грид, L – минимальная длина задания, а C – константа. Также в работе [19] показано, что в случае, если средняя длина локального задания будет составлять 30мин., планировщик способен обслуживать несколько тысяч компьютеров.

На оценку максимальной производительности системы диспетчеризации оказывают влияние сетевые операции по доставке расписаний. Планировщику необходимо иметь локальные расписания к началу цикла планирования. Агент может предоставить планировщику локальное расписание двумя способами:

- Переслать расписание, содержащее информацию обо всех слотах, непосредственно перед началом цикла планирования.
- Пересылать информацию об отдельных слотах каждый раз, когда они меняются.

При выборе одного из этих подходов, можно руководствоваться соображениями, аналогичными тем, что применялись для обоснования целесообразности обработки нескольких слотов пачкой. В случае пересылки информации обо всех слотах расписание будет передаваться один раз за цикл планирования, однако его размер будет больше. Во втором же случае за один цикл планирования будет передано несколько маленьких расписаний, при этом суммарный размер переданных данных может превосходить размер данных в первом случае, поскольку существует вероятность непустого пересечения нескольких расписаний по ресурсам.

Поскольку для коммуникаций используется аппарат грид-служб, передача любых данных влечет за собой дополнительные накладные расходы, связанные с инициализацией грид-службы. Тем самым, передача всех расписаний за один раз выгоднее, т.к., во-первых, в данном случае грид-служба будет инициализироваться только один раз, и, во-вторых, как было сказано выше, суммарный размер передаваемого расписания будет не больше суммы размеров маленьких расписаний.

Переходя к количественным характеристикам сетевых операций, заметим, что максимальный размер информации об одном слоте не превышает V=1Кб. В работе [20] показано, что в Globus Toolkit 3 для инициализации грид-службы в среднем требуется t1=0,77с. Таким образом, для передачи расписания, состоящего из N=2000 слотов требуется T=t1+N*V/U, где U – пропускная способность сети. Приняв U=10Мбит, получим T≈3с. Такая задержка является незначительной по сравнению со временем работы планировщика.

7. ЗАКЛЮЧЕНИЕ

Работа посвящена реализации системы диспетчеризации в грид, имеющем двухуровневую структуру: централизованный глобальный планировщик и множество локальных менеджеров управления ресурсами.

В работе рассмотрены методы построения систем диспетчеризации, которые могут быть применены в грид. Произведен анализ преимуществ и недостатков этих методов.

Предложен ряд принципов, таких как локальное прогнозирование и опережающее планирование. При использовании механизма локального прогнозирования можно избежать перегрузки одних ресурсов во время простоя других. Упреждающее планирование позволяет осуществить заблаговременную доставку входных файлов задания на освобождающийся вычислительный ресурс, что приводит к минимизации простоя ресурсов. На основе предложенных принципов разработана система диспетчеризации, алгоритм планирования распределения заданий по ресурсам, способ получения локального прогноза, модель базы данных.

Программная реализация GrAS поддерживает полный цикл обработки заданий и выполнена на основе Globus Toolkit 3.

8. ЛИТЕРАТУРА

1. I. Foster, C. Kesselman. Computational Grids. Chapter 2 of “The Grid: Blueprint for a New Computing Infrastructure”, Morgan-Kaufman, 1999. http://www.globus.org/alliance/publications/papers/chapter2.pdf
2. Foster, A. Roy, V. Sander, L. Winkler. End-to-End Quality of Service for High-End Applications. Technical Report, 1999. http://www.globus.org/alliance/publications/papers/end_to_end.pdf
3. Globus Toolkit. http://www.globus.org
4. Коваленко В.Н., Коваленко Е.И., Корягин Д.А., Любимский Э.З., Хухлаев Е.В., Шорин О.Н. Грид-диспетчер: реализация службы диспетчеризации заданий в грид, Труды международной конференции "Распределенные вычисления и Грид-технологии в науке и образовании" (Дубна, 29 июня-2 июля 2004 г.). - Дубна: 11-2004-205, ОИЯИ, 2004, сс. 133-139.
5. Foster, C. Kesselman, S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, 15(3), 2001. http://www.globus.org/alliance/publications/papers/anatomy.pdf
6. Коваленко В.Н., Корягин Д.А. Вычислительная инфраструктура будущего // Открытые системы. 1999. №11-12. С. 45-52. http://www.osp.ru/os/1999/11-12/045.htm

7. F. Berman. High-Performance Schedulers. Chapter 12 of “The Grid: Blueprint for a New Computing Infrastructure”, Morgan-Kaufman, 1999, 279-309.
8. M. Roehrig, W. Ziegler. Grid Scheduling Dictionary of Terms and Keywords. http://www-unix.mcs.anl.gov/~schopf/ggf-sched/GGF5/sched-Dict.1.pdf
9. Jennifer M. Schopf. Ten Actions When Grid Scheduling. Chapter 2 of “Grid Resource Management: State of the Art and Future Trends”, Springer, 2003, 15-23.
10. WMS – Workload Management System. http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/wms.shtml
11. Moab Grid Scheduler. http://www.clusterresources.com/products/mgs/specoverview.shtml
12. SQMS - Simple Queueing Management System. http://www.thaigrid.net/projects/drug_design/sqms-g
13. GridWay. http://www.gridway.org/int_arch.php
14. S. Venugopal, R. Buyya, L. Winton. A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids. http://www.gridbus.org/papers/gridbusbroker.pdf
15. V. Hamscher, U. Schwiegelshohn, A. Streit, R. Yahyapour. Evaluation of Job-Scheduling Strategies for Grid Computing. “Grid Computing - GRID 2000: First IEEE/ACM International Workshop”, Bangalore, India, December 2000, 191-202.
16. W. Cirne, F. Brasileiro, L. Costa, D. Paranhos, E. Santos-Neto, N.Andrade. Scheduling in Bag-of-Task Grids: PAUA Case. http://walfredo.lsd.ufcg.edu.br/papers/PAUA.pdf
17. Maui Cluster Scheduler. http://www.clusterresources.com/products/maui
18. В.Н. Коваленко, А.В Орлов. Управление заданиями в распределенной среде и протокол резервирования ресурсов. Препринт №1. Москва: ИПМ РАН, 2002. http://www.gridclub.ru/library/publication.2005-03-17.8842489366
19. В.Н. Коваленко, Е.И. Коваленко, Д.А. Корягин, Э.З. Любимский. Метод опережающего планирования для Грид. Препринт ИПМ им. М.В.Келдыша РАН, 2005, №112.
20. D.Chen, A.Demichev, D.Foster, V.Kalyaev, A.Kryukov, M.Lamanna, V.Pose, R.Rocha, C.Wang. OGSA Globus Toolkit3 evaluation activity at CERN. Nuclear Instruments and Methods in Physics Research A 534 (2004) 80-84.