УДК 519.68

ОСНОВНЫЕ ПОЛОЖЕНИЯ МЕТОДА ОПЕРЕЖАЮЩЕГО ПЛАНИРОВАНИЯ ДЛЯ ГРИД ВЫЧИСЛИТЕЛЬНОГО ТИПА[1]

© 2005 В.Н. Коваленко, Е.И. Коваленко, Д.А. Корягин, Э.З. Любимский[2]

Одной из наиболее важных программных технологий, образующих фундамент глобально распределенной компьютерной инфраструктуры грид, является виртуализация ресурсов. В работе предложен новый метод виртуализации, названный опережающим планированием. Особенностью метода является то, что планировщик грид заблаговременно строит план использования загрузки ресурсов. Благодаря этому, планирование даже в условиях разделения ресурсов с их владельцами обладает свойством детерминированности — выполнение плана обеспечивается путем предварительного резервирования ресурсов. Свойство детерминированности делает метод опережающего планирования пригодным для обслуживания многопроцессорных заданий.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проекты 05-01-00626-а, 04-07-90299-в).

1. Краткие предварительные сведения

Хотя единой точки зрения на то, что такое грид, и, тем более, общепринятого определения этого понятия до сих пор не существует, дискуссии [1] ведутся главным образом по поводу принципов организации, стандартов архитектуры и конкретных технологий. В то же время расхождений по поводу базовой идеи — интеграции распределенных в сети ресурсов и обеспечения эффективного удаленного доступа к ним — существенно меньше, что позволяет сформулировать следующее общее определение грид.

Грид — это программно-аппаратная среда, которая построена из вычислительных установок, находящихся в различных административных доменах телекоммуникационной сети, и которая позволяет дистанционно использовать любое количество ресурсов этих установок — процессорных, оперативной и постоянной памяти, программ и данных.

[1] Представлена доктором физико-математических наук профессором В.А. Салеевым.

[2] Коваленко Виктор Николаевич (kvn@keldysh.ru), Коваленко Евгения Ивановна (kei@keldysh.ru), Корягин Дмитрий Александрович (koryagin@keldysh.ru), Любимский Эдуард Зиновьевич (ljubimsk@keldysh.ru), Институт прикладной математики им. М.В.Келдыша РАН, 125047, Россия, г. Москва, Миусская пл., 4.

Усилия специалистов, разрабатывающих технологии грид, направлены на то, чтобы обеспечить качественно новый уровень интеграции распределенных ресурсов, такой, чтобы грид можно было рассматривать как единую операционную среду обработки запросов (заданий), работа в которой была не сложнее, чем с более привычными компьютерными архитектурами: современными компьютерами, многопроцессорными и кластерными системами.

С помощью разработанного к настоящему времени программного обеспечения грид [2] созданы функционирующие в производственном режиме глобально распределенные компьютерные комплексы сверхвысокой производительности и пропускной способности с агрегированной мощностью порядка нескольких терафлопс и петабайтными объемами хранимых данных: EGEE (`www.eu-egee.org`), NorduGrid (`www.nordugrid.org`), Grid2003 (`www.ivdgl.org/grid2003`), TeraGrid (`www.teragrid.org`). Построение подобных компьютерных инфраструктур дает возможность решать прикладные задачи беспрецедентного уровня сложности как в научно-технической, так и в производственной сфере [3].

Грид принадлежит к архитектурам распределенного компьютинга с коллективной формой обслуживания пользователей, и у него, конечно, есть аналоги — в качестве ближайшего можно указать кластерные системы. Поэтому многие из задач, решаемых программным обеспечением грид, известны, однако в условиях грид постановка этих задач значительно усложняется, и для их решения требуются новых подходы.

Среди наиболее значимых отличительных свойств грид отметим:

- Глобальную распределенность. Составляющие грид вычислительные установки распределены в открытой глобальной сетевой среде, в которой велики коммуникационные издержки и предъявляются повышенные требования к безопасности.
- Автономность ресурсов. Ресурсная база грид формируется из независимых друг от друга вычислительных центров или отдельных компьютеров. Ресурсы имеют разных владельцев, которые, обслуживая и администрируя свою часть ресурсов, имеют право проводить собственную политику доступа к ним.
- Разделение ресурсов. Ресурсы грид используются коллективно, поэтому должно обеспечиваться гибкое, скоординированное их распределение между пользователями, решающими разные задачи.
- Динамичность среды. Как состав ресурсов, так и состав пользователей может меняться, причем это является правилом, а не исключением.

Набор функций, которые требуются в этих условиях от программного обеспечения, в главных чертах определился — это средства обеспечения безопасности, надежности, мониторинга устройств, учета и протоколирования заданий. Ключевое место в этом ряду занимает функция управления заданиями — диспетчеризация: она обеспечивает распределение ресурсов из общего ресурсного пула грид между заданиями, доставку программ и данных, реализуя одну из важнейших концепций грид — виртуализацию ресурсов.

Уже сейчас существуют программные разработки, в которых достигнут практически приемлемый уровень реализации перечисленных функций. С наибольшей полнотой круг вопросов, связанных с поддержкой функционирования грид, решен в программных платформах LCG [4] и gLite [5], в которых интегрированы и взаимоувязаны компоненты программного обеспечения грид, разработанные разными

исследовательскими коллективами. Тем не менее, едва ли можно утверждать, что получены окончательные решения.

Тема данной работы — планирование распределения заданий по ресурсам грид. Планирование происходит в контексте системы управлениями заданиями, обеспечивающей их дистанционное выполнение. Объекты планирования — задания и ресурсы, но задания запускаются пользователями, а ресурсы имеют владельцев, и мы рассматриваем такое планирование, которое учитывает интересы тех и других (разд. 2). В разд. 3 вводится понятие детерминированного планирования: такого, что построенный планировщиком план действительно реализуется. Оказывается, в условиях грид (разд. 3.1–3.3) сделать это непросто. В первую очередь это обусловлено тем, что архитектура грид имеет два уровня управления: на верхнем действует планировщик грид, а на нижнем — локальный менеджер ресурсов. В большинстве применяемых на практике систем планирования в качестве аппарата взаимодействия этих двух уровней используются традиционные для локальных менеджеров ресурсов интерфейсы, изначально предназначенные для обслуживания пользователей систем пакетной обработки (СПО). Использование этого аппарата в грид приводит к существенным дефектам планирования (разд. 3.4), например к невозможности обеспечить точное время старта заданий, а также накладывает жесткие ограничения на способы организации грид, что не позволяет в полной мере удовлетворить такие необходимые требования, как возможность включения в грид разнотипных компьютеров или возможность использования ресурсов грид совместно с их владельцами.

В разд. 4 излагаются исходные положения метода опережающего планирования, определяющие стратегию, основанную на соглашениях о разделении ресурсов между пользователями и владельцами ресурсов. В разд. 5 показывается, каким образом можно добиться детерминированности планирования путем расширения интерфейса СПО двумя новыми функциями: прогнозирования загрузки ресурсов и предварительного резервирования. В разд. 6 приводится программная схема опережающего планирования, использующего эти функции. Дана также оценка масштабируемости предлагаемого метода (разд. 6.1). Разд. 7 посвящен обзору современных систем планирования в грид и их сравнению с предлагаемым подходом.

2. Место планирования в архитектуре грид

Получивший общее признание способ организации программного обеспечения грид основан на открытой архитектуре служб OGSA [6], в соответствии с которой грид квалифицируется как программная система, состоящая из распределенных компонентов — служб, взаимодействующих между собой посредством стандартных, открытых и универсальных протоколов и интерфейсов [7]. Имея в виду главным образом грид вычислительного типа, можно рассматривать его функционирование как процесс обслуживания стандартизированных запросов на выполнение вычислений, оформленных в виде заданий для общераспространенных операционных систем, причем выполнение этих заданий производится на ресурсах, которые выбираются из общего пула. Перечислим основные этапы обработки задания (рис. 1.):

- Планирование ресурсов. Специальная компонента программного обеспечения — планировщик выделяет из общего пула исполнительные ресурсы — те, на которых задание будет выполняться.

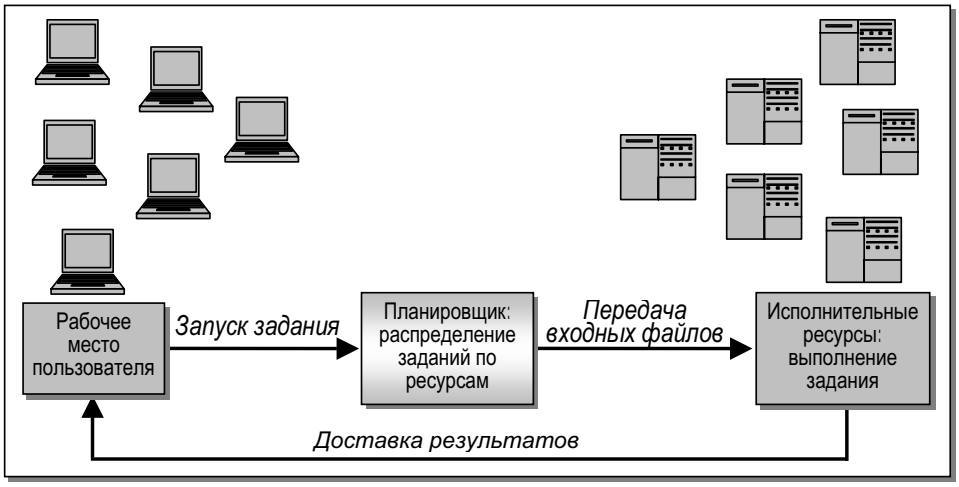

Рис. 1. Схема обработки запросов и виртуализация ресурсов

- Доставка исполняемых файлов и входных файлов на исполнительные ресурсы.
- Выполнение задания.
- По окончании задания доставка результирующих файлов на серверы хранения (в частности, на рабочее место пользователя).

Все перечисленные этапы обработки задания выполняются автоматически, без участия субъекта, выдавшего запрос (в частности, пользователя, хотя это может быть и программа), так что грид действительно представляет собой единую операционную среду.

Поскольку грид — распределенная среда, то, как показала практика, необходимы специальные службы, выполняющие:

- мониторинг и протоколирование процесса обработки запроса, что позволяет в каждый момент получать информацию о местонахождении и состоянии задания, а также управлять его обработкой (удаление, модификация);
- обеспечение надежности — перезапуск задания при сбоях программных или аппаратных средств на ресурсах и телекоммуникациях;
- мониторинг ресурсной среды и учет потребления ресурсов, на основе чего производятся планирование и диагностика сбоев оборудования.

Таким образом, планирование ресурсов — основная тема данной работы — происходит в контексте сложной многокомпонентной системы обработки заданий, примером которой является WMS (Workload Management System) [8], входящая в состав упоминавшейся выше платформы LCG. В этом контексте назначение планирования — распределение ресурсов грид между заданиями. Планирование имеет два аспекта: во-первых, оно определяет исполнительные ресурсы, на которых будет выполняться каждое задание, и, во-вторых, время, когда исполнительные ресурсы отводятся тому или иному заданию. Цель планирования — обеспечить скоординированное разделение ресурсов, учитывающее как интересы пользователей,

так и владельцев ресурсов. Эти интересы выражаются правилами и соглашениями, которые устанавливаются объединениями пользователей и владельцев ресурсов — виртуальными организациями. Задача планирования в грид представляет интерес как самостоятельное исследовательское направление, которому посвящено большое количество публикаций (см., например, [9]).

3. Специфика планирования в грид

Планирование в грид хотя и имеет специфику, относится к широкому классу задач планирования, общая постановка которых характеризуется наличием множества ресурсов, множеством их потребителей и определенными условиями выделения ресурсов потребителям [10]. Известно, что способы планирования в зависимости от этих условий, целей и других факторов могут быть совершенно разными.

Что касается постановки задачи в данной работе, то в качестве ресурсов будут рассматриваться вычислительные ресурсы компьютеров $\mathbf{RS}=\{\mathbf{RS_i},\ \mathrm{i}=1,\ldots,\mathrm{N}\}$ — процессоры вместе со связанными с ними оперативной и дисковой памятью. Компьютеры глобально распределены, то есть могут находиться в произвольных точках глобальной сети. В систему планирования (планировщик) от работающих независимо друг от друга пользователей поступает неограниченный во времени поток заданий $\mathbf{Z}(\mathrm{T})=\{z_i(t_i),\ \mathrm{i}=1,2,\ldots\}$, где t_i — время поступления задания. Мы предполагаем, что все компьютеры однопроцессорные и выделяются заданиям в эксклюзивном режиме: в любой момент времени на компьютере выполняется не более одного задания.

Результатом планирования является последовательность *аллокаций* ресурсов $\mathbf{A}=\{a_i,\ \mathrm{i}=1,2,\ldots\}$. Нас интересуют способы планирования, способные строить *точные аллокации* $a \in \mathbf{A}$, $a=[\mathrm{z},\ \mathbf{IRS},\ t_b,\ t_e]$, в которых определяется множество исполнительных ресурсов $\mathbf{IRS} \subseteq \mathbf{RS}$ для выполнения задания z и временной интервал $[t_b,t_e]$, на который они отводятся заданию z. На этом интервале ресурсы считаются занятыми и недоступными другим заданиям.

Назовем планирование *детерминированным*, если оно строит точные аллокации, и запуск заданий на исполнительных ресурсах осуществляется в соответствии с этими аллокациями. Для этого необходимо, чтобы:

- характеристики среды исполнительных ресурсов отвечали требованиям задания;
- задание могло быть доставлено на исполнительные ресурсы к моменту начала аллокации;
- исполнительные ресурсы были к этому моменту свободны, и политика управления ресурсами не препятствовала их выделению для задания.

Как будет показано в следующих разделах, особенности грид делают возможность построения точных аллокаций проблематичной, а вместо этого во многих алгоритмах и практических реализациях планировщиков результатом являются аллокации, в которых некоторые элементы (исполнительные ресурсы, время начала аллокации) определены примерно, что делает планирование недетерминированным в той или иной степени. Проявляется это в серьезных дефектах обработки заданий: например, если не известно точно время начала аллокации, то в некоторых ситуациях запуск задания вообще может не состояться — есть шанс, что задание "зависнет".

Рассмотрим далее обстоятельства, в которых происходит планирование в грид и которыми в большой степени определяется возможность построения метода планирования, обладающего свойством детерминированности: гетерогенность ресурсов; их организация; а также режимы использования.

3.1. Гетерогенность ресурсов и их отбор для заданий

Задание грид представляет собой обычный исполняемый файл (скрипт, программный код). Службами грид оно доставляется на исполнительные ресурсы, а собственно выполнение происходит в среде операционной системы (ОС) этих ресурсов. Как правило, программа, подготовленная на определенном компьютере, не требует каких-либо модификаций для использования в грид. Однако любая программа рассчитана на определенную среду выполнения — ОС, архитектуру компьютера, объемы и характеристики его ресурсов.

Строя аллокации, планировщик должен учитывать, что грид является гетерогенной инфраструктурой, в состав которой могут включаться компьютеры с разной архитектурой и комплектацией. Гетерогенность проявляется в том, что разные классы ресурсов (процессор, основная память, кэш-память, дисковая память) различаются типом (процессоры — архитектурой) и характеристиками (процессоры — производительностью, память — объемом). В связи с этим выбираемые исполнительные ресурсы не могут быть произвольными, а должны соответствовать требованиям задания. Эти требования не обязательно ограничиваются приведенными примерами: любая особенность компьютера, существенная для выполнения задания, например операционная система, может рассматриваться как класс ресурсов.

При запуске через систему планирования требования задания оформляются в виде *ресурсного запроса*. Используемые на практике формализмы записи ресурсного запроса, хотя и разнообразны, но ориентированы на то, чтобы рассматривать ресурсы унифицированным способом, предполагая, что набор классов является стандартизированным (с возможностью расширения). Простейший вид ресурсного запроса можно представить таким образом:

<ресурсный запрос>:={<класс>={<тип>|<характеристики>}...} [<время использования ресурсов>]

Например, по ресурсному запросу: CPU=1 ГГц, OS=Linux должны выделяться машины с процессорной частотой не меньше 1 ГГц и операционной системой Linux.

Обратим внимание на параметр ресурсного запроса, определяющий время использования ресурсов, то есть время выполнения задания. Известно, что пользовательская оценка этого времени редко бывает точной, однако наличие этого параметра представляется весьма важным по двум причинам. Во-первых, даже приблизительная оценка времени выполнения позволяет использовать более эффективные алгоритмы планирования [11]. Во-вторых, существует общепринятая практика работы в системах с разделяемыми ресурсами, согласно которой заказ времени использования ресурсов (как и ресурсный запрос в целом) служит защитой от программных ошибок, представляя собой ограничения (по объемам и по времени) на потребляемые ресурсы: при превышении указанных в запросе лимитов задание принудительно завершается. Наличие этого параметра не препятствует тому, чтобы в грид могли обрабатываться задания с неограниченным временем выполнения, однако они, по-видимому, должны рассматриваться как особый случай.

Отметим, что параметр времени исполнения заказывается в расчете на определенную производительность ресурсов. Как в процессе планирования, так и при запуске задания должен производиться пересчет времени в соответствии с конкретными исполнительными ресурсами. Этот вопрос обсуждается в [12].

На практике получили распространение несколько разных языков ресурсных запросов. Язык RSL [13], применяемый в системе Globus Toolkit [14], ориентирован на запуск как однопроцессорных, так и многопроцессорных MPI-заданий. Языки ClassAd [15] системы Condor [16] и JDL [17] (WMS) позволяют определять альтернативные варианты ресурсного запроса и специфицировать пользовательские предпочтения при выборе ресурсов. В ряде работ предлагаются существенные расширения формализма языка запросов, направленные на спецификацию связанных заданий и их цепочек [18].

При планировании отбор ресурсов производится по информационной базе, содержащей сведения о составе и характеристиках ресурсов грид. Поставка этих данных в информационную базу осуществляется в оперативном режиме специализированными распределенными системами мониторинга ресурсов, из которых наибольшую распространенность получили MDS [19] и R-GMA [20]. Ресурсный запрос выступает в качестве формального критерия отбора, однако он лишь сужает множество возможных исполнительных ресурсов. Для их однозначного определения должны приниматься во внимание дополнительные соображения.

3.2. Способы организации грид и современные интерфейсы доступа к ресурсам

Важной особенностью грид является то, что процесс обработки заданий в нем происходит на двух уровнях: глобальном и локальном. На глобальном уровне управление заданиями осуществляется программным обеспечением грид, выполняющим планирование, дистанционную передачу заданий и другие перечисленные в п. 2 функции. После того как задание доставляется на исполнительные ресурсы, оно передается программному обеспечению локального уровня — *менеджеру ресурсов*. Менеджер ресурсов является неотъемлемой составляющей архитектуры грид, так как именно эта компонента фактически реализует поступающие дистанционные запросы, такие как запуск и управление заданиями, в локальной среде их обработки. Отметим также, что менеджер ресурсов регулирует с помощью очередей загрузку ресурсов в соответствии с политикой, которая определяется их владельцем.

Такая двухуровневая организация управления приводит к тому, что принадлежащий глобальному уровню планировщик не имеет непосредственного контроля над ресурсами. Все операции над заданиями, а также операции по получению информации о состоянии ресурсов могут производиться только через менеджера ресурсов. Из этого следует важный вывод: возможности планирования в грид определяются теми интерфейсами, которыми располагают менеджеры ресурсов. В связи с этим рассмотрим, как устроен локальный уровень грид.

На современном этапе развития практически полностью доминируют менеджеры, управляющие кластеризованными ресурсами. При организации ресурсов в виде кластеров множество компьютеров, принадлежащих одному административному домену, включаются в грид как один узел, а в качестве менеджера выступает система пакетной обработки (СПО), обслуживающая всю совокупность компьютеров узла. Кластер в совокупности с СПО во многом аналогичен грид, однако он

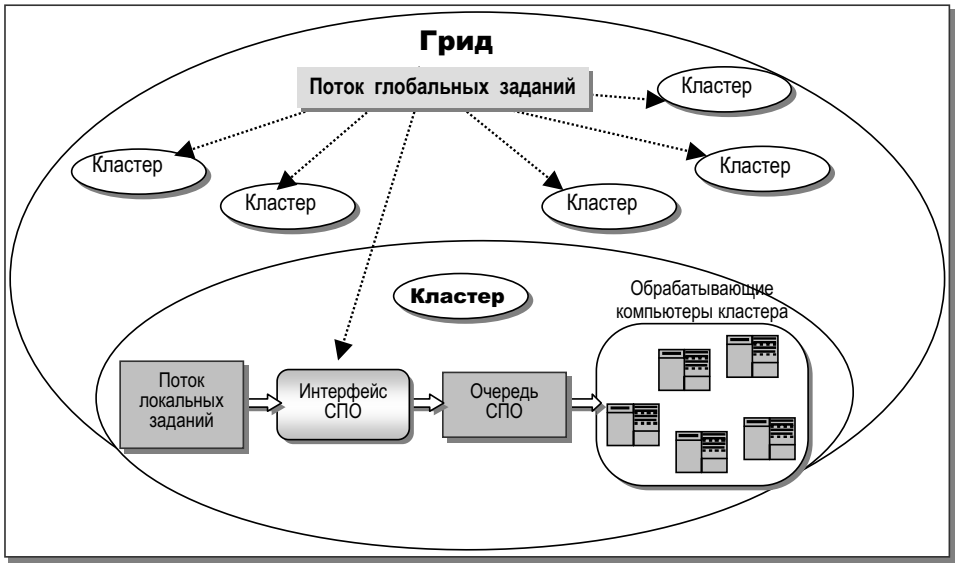

Рис. 2. Схема обработки заданий в системе пакетной обработки (СПО)

функционирует в замкнутой корпоративной среде и администрируется централизованно. СПО имеет интерфейсы управления заданиями и выполняет их обработку, реализуя распределение заданий по входящим в состав кластера компьютерам в соответствии с ресурсными запросами, аналогичными описанным выше. СПО ведет собственные очереди, буферизуя поступающие задания, если требуемые для их выполнения ресурсы временно заняты.

Большинство используемых на практике СПО (PBS [21], SGE [22], LSF [23], Condor [16]) были созданы задолго до появления грид и проектировались как самодостаточные системы. Поэтому возможности взаимодействия программного обеспечения грид с такими менеджерами ограничены пользовательскими (и эквивалентными им программными) интерфейсами двух типов:

- запуска и управления заданиями;
- информирования о состоянии отдельных заданий и очередей.

Схема взаимодействия системы управления заданиями грид с СПО приведена на рис. 2. Поступающие дистанционно запросы на обработку заданий грид передаются в СПО через пользовательский интерфейс, где попадают в одну из очередей (которых в СПО как правило бывает несколько). После этого порядок выделения заданиям ресурсов определяет планировщик СПО в соответствии с устанавливаемой владельцем политикой управления.

3.3. Режимы использования ресурсов

Сейчас, особенно в крупных проектах для поддержки научных приложений, наиболее распространен способ создания грид, когда владельцы выделяют в него свои ресурсы и не используют их сами, то есть выступают в роли профессиональных провайдеров. Этот способ имеет свою область существования, но представляет интерес также грид, в котором организации или отдельные лиц объединяют

имеющиеся ресурсы, но не передают их в грид полностью — ресурсы должны совместно использоваться и в грид, и владельцами. Ресурсы, использующиеся в режиме разделения между владельцами и пользователями грид, мы будем называть *неотчуждаемыми*. Достоинство грид с неотчуждаемыми ресурсами в том, что, в отличие от первого способа, здесь не требуется затрат на формирование специальной ресурсной базы, и грид может создаваться динамически на ограниченный период времени с целью решения какой-либо крупной проблемы, для которой ресурсов отдельных членов кооперации не хватает.

С точки зрения планирования неотчуждаемость ресурсов усложняет ситуацию, поскольку в этом случае на них поступает два потока заданий: 1) поток из грид (глобальный), управляемый планировщиком, и 2) локальный поток заданий, которые запускаются средствами, отличными от пользовательских интерфейсов грид. Например, в условиях кластеризованных ресурсов локальные задания вводятся непосредственно через интерфейсы СПО. Таким образом, локальные задания неподконтрольны планировщику, хотя создаваемая ими загрузка ресурсов должна учитываться при распределении глобальных заданий.

3.4. Проблемы планирования при использовании стандартных интерфейсов менеджеров ресурсов

Стандартный набор интерфейсов менеджеров ресурсов является слишком ограниченным, для того чтобы на их базе можно было построить детерминированное планирование. Покажем, какие проблемы порождает их использование.

Определение исполнительных ресурсов. В рамках интерфейсов СПО планировщик может получить только интегральную информацию о состоянии кластеров: количество заданий в очередях, общее число процессоров, число загруженных процессоров и т.п. Однако характеристики обрабатывающих компьютеров, существенные для подбора исполнительных ресурсов, такие как платформа или быстродействие, остаются недоступными. Информация общего плана, которой располагает планировщик, позволяет построить аллокацию, в которой в качестве исполнительных ресурсов определен кластер, однако не гарантируется, что в нем есть необходимые заданию компьютеры с нужной архитектурой, достаточным объемом памяти, и оно действительно может быть в нем запущено.

Планирование, основанное на такой информационной базе, может применяться только в грид с однородными по ресурсам кластерами. Если же это не так, то, по-видимому, единственно возможный способ планирования — спекулятивный запуск заданий во все кластеры грид.

Оценка времени начала аллокации. При построении аллокации планировщик может точно оценить время ее начала только при условии, что в кластере есть достаточное для планируемого задания количество свободных компьютеров — тогда оно может начать выполняться сразу. В противном случае в результате использования аллокации задание будет поставлено в очередь, но время его запуска на счет будет неопределенным. Говоря точнее, оно зависит от всех заданий, находящихся в кластере — в состоянии выполнения или ожидания в очереди, а также от того, какие задания будут поступать в кластер в дальнейшем.

Возможность получения аллоцированных ресурсов. Даже в ситуации, когда в кластер не поступают локальные задания, неопределенность времени начала аллокации приводит к задержкам в обработке глобальных заданий (и простою ресурсов), однако, при разделении ресурсов с владельцами последствия будут более

тяжелыми. Если в кластер поступает неконтролируемый планировщиком локальный поток, задание грид может оставаться в очереди кластера неограниченно долго — оно может "зависнуть".

Для полноты картины рассмотрим положение дел с планированием при ином способе организации ресурсов, который не предполагает их кластеризацию. Хотя грид с кластеризованными ресурсами получил наибольшее распространение, развивается и альтернативный, весьма перспективный, по нашему мнению, подход, в котором грид строится из отдельных компьютеров [24, 25]. Сравнению этих двух способов посвящена работа [26].

При включении ресурсов в грид без объединения их в кластеры менеджер ресурсов устанавливается на каждый компьютер и обслуживает только его, осуществляя локальное управление заданиями путем непосредственного взаимодействия с ОС. При таком "индивидуальном" подходе информационные интерфейсы менеджера позволяют предоставлять планировщику полный набор данных (объемы и характеристики ресурсов), необходимых для того, чтобы отбирать исполнительные компьютеры в точном соответствии с ресурсным запросом. Решается также проблема гарантированного запуска заданий: если в варианте с кластеризованными ресурсами время начала аллокации определяется путем опроса состояния кластера через интерфейсы СПО, что приводит к дефектам планирования, то менеджер компьютера сам определяет возможность выполнения задания грид и выдает планировщику запрос на его получение.

Основная проблема грид с некластеризованными ресурсами — непредсказуемость времени окончания задания. Такой грид наиболее выгоден, если ресурсы используются в режиме разделения с владельцами. Разделение ресурсов происходит здесь непосредственно во время выполнения: ресурсы, в том числе и процессорные, делятся между заданием грид и приложениями, выполняемыми владельцем компьютера. Владелец компьютера устанавливает политику разделения ресуров таким образом, чтобы обработка задания грид не оказывала существенного влияния на его деятельность, то есть фактически разрешает занимать компьютер в периоды простоя. Помимо того, компьютеры могут выключаться в произвольные моменты времени. Эти обстоятельства приводят к тому, что реальное время обработки задания на таких ресурсах будет существенно отличаться от процессорного времени, заказанного в ресурсном запросе. Проблема планирования в этом случае — сделать реальное время выполнения приемлемым для пользователя.

4. Основные положения предлагаемого метода планирования

Как показано в предыдущем разделе, возможности планирования в условиях кластеризованных неотчуждаемых ресурсов на основе интерфейсов современных СПО весьма ограничены. Используемые на практике брокеры ресурсов, например GRB [8] системы WMS и брокер платформы ARC/NorduGrid [27], имеют довольно жесткие ограничения по применению и не способны исключить такие нежелательные эффекты, как непредсказуемость времени обработки заданий, задержка обработки в ситуациях, когда имеются простаивающие ресурсы. Существенный недостаток этих систем — невозможность планирования многопроцессорных параллельных заданий с синхронным выделением ресурсов в нескольких кластерах.

Предлагая новый подход к планированию, мы исходим из того, что грид, по-

видимому, допускает многообразные формы воплощения в зависимости от мотивов его создания, организации ресурсов, предметной ориентации, способа использования и иных обстоятельств. Многообразие форм грид предполагает и различие в методах планирования, поэтому, ориентируясь главным образом на кооперативный грид некоммерческих организаций, определим условия применения и основные принципы, которые характеризуют предлагаемый подход.

Ресурсы и задания. Ресурсы грид организованы в кластеры, управляемые СПО, и используются совместно с владельцами, то есть рассматривается грид с кластеризованными неотчуждаемыми ресурсами. Объектами планирования являются одно- и многопроцессорные задания, причем выполнение последних может происходить как в пределах одного кластера, так и на ресурсах нескольких кластеров.

Стратегия планирования. Отбор ресурсов для заданий на основе ресурсных запросов не в полной мере определяет способ планирования, полностью оставляя в стороне вопрос о согласовании распределения ресурсов между разными заданиями. Необходимые дополнительные условия, которые можно назвать стратегией планирования, часто удается формализовать в виде целевой функции, оценивающей качество планирования для различных вариантов распределения заданий на некотором временном интервале. Планирование в этом случае сводится к оптимизационной задаче. Такой способ применим и к грид, где в качестве оптимизационного критерия могут выступать максимизация загрузки ресурсов, дохода владельцев или минимизация расходов на оплату ресурсов со стороны пользователей [28].

В любом случае выбор стратегии должен исходить из определенных соглашений по разделению ресурсов. В грид это соглашения двух типов: 1) между пользователями и 2) между пользователями и владельцами ресурсов. В качестве основы для выработки этих соглашений наиболее перспективным представляется экономический подход [29], в котором делается попытка формализовать процесс распределения ресурсов с помощью модели рынка: на рынке ресурсов грид в качестве продавцов выступают владельцы ресурсов, а в качестве покупателей — пользователи. В рамках этой модели рассматриваются проблемы ценообразования, обеспечения надежности в расчетах, представления политик переговоров, а также применения соответствующих программных разработок [30] в различных операционных компонентах программного обеспечения грид, в том числе в брокерах ресурсов [31]. Основываясь на основных идеях экономического подхода, рассмотрим далее конкретный вид предлагаемых соглашений по разделению ресурсов и способ их поддержки при планировании.

4.1. Разделение ресурсов между пользователями

Наш выбор стратегии разделения ресурсов между пользователями направлен на то, чтобы каждый пользователь мог контролировать полное время обработки своих заданий и, в том числе, управлять скоростью получения ресурсов на этапе планирования. Конечно, прямой заказ времени получения ресурсов невозможен, поскольку за ресурсы конкурирует множество заданий, однако если задействовать механизм приоритетов и позволить пользователю устанавливать приоритеты своих заданий, то тем самым у него появится средство для изменения их положения в очереди на получение ресурсов.

Существенен вопрос, как совместить присвоение приоритетов заданиям самим пользователем с принципом "справедливого" разделения ресурсов между пользо-

вателями. В локальных системах расстановка приоритетов осуществляется административно, однако для грид такое простое решение не годится. В рамках экономического подхода эквивалентом приоритета может выступать плата за задание, вносимая пользователем. Изменяя плату, пользователь регулирует время старта задания. Отметим, что платность не исключает возможности обслуживания и "бесплатных" заданий, но они будут получать ресурсы в последнюю очередь.

Важно, ведутся ли расчеты реальными (в коммерческих виртуальных организациях) или условными деньгами (в кооперативных виртуальных организациях). В последнем варианте возможность оперативного управления отдельными заданиями должна быть дополнена интегральными ограничениями на общее количество получаемых пользователем ресурсов. Проблема "справедливого" (fair-share) разделения ресурсов возникает и в корпоративных системах коллективного пользования, а некоторые решения [32] применимы в условиях грид. При использовании экономического подхода виртуальная организация выделяет каждому пользователю ограниченный бюджет на определенный период времени, из которого он вносит плату при запуске задания. Определение величин бюджетов для множества пользователей грид производится иерархическим способом аналогично, например, принятому в грид способу определения прав доступа к ресурсам. Совокупность двух механизмов — оплаты заданий и ограничения бюджета, — позволяя управлять скоростью обработки каждого отдельного задания, не дает монополизировать ресурсы отдельному пользователю.

Подводя итог, скажем, что в качестве стратегии планирования в аспекте разделения ресурсов между пользователями мы выбираем приоритетное обслуживание заданий. Этот выбор определяет *архитектуру* системы планирования: централизованная архитектура (в противоположность распределенной [33]) с общей очередью глобальных заданий.

4.2. Разделение ресурсов с владельцами

Соглашение о разделении ресурсов с владельцами является ключевым принципом предлагаемого метода планирования. В рассматриваемой ситуации с неотчуждаемыми ресурсами имеется два потока заданий — глобальный и локальный, и соглашение о разделении ресурсов определяет, какие из распределяемых ресурсов и в какое время могут быть отданы тому или иному глобальному заданию. Соответствующие правила должны быть специфицированы таким образом, чтобы планировщик грид был в состоянии установить возможность занятия ресурсов заблаговременно, в момент построения аллокации.

На практике применяются два варианта разделения ресурсов между локальным и глобальным потоками.

1) Ресурсы кластера делятся на две непересекающиеся части: для глобальных и локальных заданий. Система очередей СПО конфигурируется таким образом, чтобы задания каждого сорта распределялись только на свою часть ресурсов.
2) Ресурсы кластера образуют одно множество. Задания грид запускаются в СПО с некоторым фиксированным приоритетом.

И первый, и второй варианты имеют недостатки и не позволяют эффективно управлять разделением ресурсов. В первом варианте одна часть ресурсов будет простаивать, если нет распределяемых на эту часть заданий, в то время как ресурсов для обслуживания заданий другого сорта может не хватать. Имеющаяся

статистика показывает [34], что при таком статистическом разделении средний уровень использования ресурсов не превышает 25%. Во втором варианте время получения ресурсов глобальными заданиями будет неопределенным, так как оно зависит от приоритетов локальных заданий, находящихся в СПО. Если же глобальным заданиям присваивается максимально возможный приоритет, то они всегда будут иметь преимущественное право получения ресурсов, и при интенсивном их потоке обработка локальных заданий может быть блокирована полностью.

Мы предлагаем соглашение о разделении ресурсов, которое осуществляет динамическую балансировку потоков заданий. Соглашение состоит в том, что ресурсы, которые получает глобальное задание, должны оплачиваться, и глобальное задание может занять ресурсы при условии, что назначенная пользователем при запуске задания плата не меньше их стоимости. Принципиальной особенностью этого соглашения является то, что стоимость ресурсов меняется во времени и определяется не только их объемами, а вычисляется с учетом приоритетов локальных заданий, которые могут занять те же ресурсы. Цена использования ресурсов в единицу времени задается формулой PR(t)=C*P(t)*F($\mathbf{R}$), где F($\mathbf{R}$) — неубывающая функция, зависящая от объема ресурсов $\mathbf{R}$ каждого типа, которыми располагает компьютер, а P(t) — приоритет локального задания, которое должно выполняться на компьютере во время t. Это дает гибкий механизм балансировки потоков, позволяющий, с одной стороны, администратору кластера путем изменения цены ресурсов (коэффициент C) ограничивать интенсивность потока внешних заданий (при достаточно больших значениях коэффициента C глобальные задания вообще не будут поступать в кластер), а с другой — дающий возможность пользователю грид влиять на скорость получения ресурсов для своих заданий, повышая плату.

Поскольку стоимость ресурсов определяется тем, какие локальные задания претендуют на их получение, планировщик грид должен заблаговременно снабжаться информацией, описывающей будущее состояние каждого компьютера грид.

5. Обеспечение детерминированности планирования

Предлагаемый метод планирования, базирующийся на рассмотренных в предыдущем разделе принципах, мы называем *опережающим* планированием, так как ключевой механизм в нем — опережающее события в кластерах построение точных аллокаций на основании прогноза состояния ресурсов. Цель разработки нового метода — сделать планирование детерминированным. Для этого требуется, чтобы построенные планировщиком точные аллокации могли быть реализованы, то есть чтобы по ним мог быть осуществлен запуск заданий на исполнительных ресурсах. Обеспечить это возможно, но, как показано в п. 3.4, необходимо расширение механизмов взаимодействия системы планирования с менеджерами ресурсов. Вопрос о развитии функциональности СПО и соответствующем расширении их интерфейсов назрел, и есть достаточно большое количество предложений на этот счет. Их систематизации посвящена работа [35]. Два новых механизма: прогноз загрузки и предварительное резервирование ресурсов, на основе которых строится метод опережающего планирования, лежат в русле этих предложений.

5.1. Прогноз загрузки кластерных ресурсов

Опережающее планирование опирается на прогноз распределения локальных заданий по ресурсам кластеров — расписание загрузки компьютеров. Это расписание описывает, какие задания будут занимать компьютеры в любой момент времени из некоторого временного диапазона. Расписание состоит из множества слотов вида:

Слот := [Компьютер], [Локальное задание], [Начало, Конец]

Каждый слот определяет интервал времени [Начало, Конец], в течение которого компьютер выделяется локальному заданию. Совокупность интервалов, относящихся к одному компьютеру, может покрывать временной диапазон расписания лишь частично: промежутки между слотами соответствуют периодам, когда компьютер не занят.

Прогноз, содержащийся в расписании, зависит от алгоритма планирования СПО, параметров политики управления, а для его получения требуется учет характеристик (ресурсных запросов, включая заказанное время исполнения, а также приоритетов) всех заданий, находящихся в кластере. По-видимому, единственный способ построения прогноза такого рода — моделирование процесса обработки заданий в кластере.

Хотя ни одна из известных СПО не имеет готовых средств для построения расписаний, возможность реализации такой функциональности была нами показана [36] на примере кластерного планировщика Maui [37]. Реализация выполнена на базе входящей в состав Maui подсистемы моделирования. Исходное назначение этой подсистемы — оценка эффективности загрузки ресурсов: получая на вход модельный поток заданий и используя текущие настройки кластерного планировщика, она выдает повременное распределение заданий по ресурсам. Реализация построения расписаний свелась в основном к внедрению в цикл планирования кода, фиксирующего текущее состояние кластера, запускающего процесс моделирования и передающего ему исходные данные. Опыт этой разработки позволяет утверждать, что создание компоненты построения расписаний в СПО любого типа, в том числе и с отличными от Maui алгоритмами планирования, возможно: задача сводится к эмуляции управления очередями и выполнения заданий на ресурсах.

Для того чтобы планировщик грид мог вычислять стоимость ресурсов, требуется форма расписаний, несколько отличная от получаемой в результате моделирования. Поэтому перед передачей планировщику грид слоты расписания преобразуются так, что в них вместо заданий указывается вычисляемая с учетом приоритетов тех же заданий цена ресурсов в единицу времени:

Слот := [Компьютер], [Цена ресурсов], [Начало, Конец]

Построенные в разных кластерах расписания, которые мы будем называть *локальными расписаниями*, поставляются в информационную базу планирования, причем поставка ведется в оперативном режиме, отражающем происходящие изменения. Напомним, что в ту же базу заносятся данные о характеристиках компьютеров, но эта информация более статична. По информационной базе планировщик грид производит размещение глобальных заданий, строя для них аллокации ресурсов. В планировщике может использоваться любой алгоритм (важно только, чтобы он был модифицирован с учетом соглашения о разделении ресурсов), а само планирование может вестись на любую глубину по времени, согласованную с временным диапазоном расписаний. Так, например, если планировщик обслуживает только однопроцессорные задания, в качестве алгоритма может выступать,

например, FCFS [38], и распределять достаточно только те компьютеры, на которых освобождение ресурсов, то есть смена заданий, прогнозируется на ближайшее время. Однако в полной мере достоинства опережающего планирования проявляются в том, что его можно применить для многопроцессорных заданий.

Одна из наиболее трудных проблем планирования в грид — *коаллокация*: синхронное выделение ресурсов в нескольких кластерах для многопроцессорных заданий. Известно, что простые алгоритмы приоритетного планирования не способны обеспечить запуск заданий, требующих большого количества ресурсов, независимо от того, насколько высок их приоритет. Причина заключается в фрагментации ресурсного пула из-за того, что ресурсы постоянно выделяются более мелким заданиям. Для предотвращения зависания многопроцессорных заданий необходимо каким-то образом накапливать ресурсы для них. Лучший на сегодня алгоритм для планирования многопроцессорных заданий — алгоритм "обратного заполнения" BackFill [39], разработанный для систем типа MPP и достаточно широко применяющийся на практике. Идея этого алгоритма состоит в том, что ресурсы выделяются заданиям не непосредственно в момент освобождения, а заблаговременно: планировщик строит план распределения ресурсов — расписание запусков заданий. При построении расписания ресурсы распределяются заданиям в порядке их приоритетов, причем задание могут получить некоторые ресурсы только при условии, что они уже не отведены более приоритетным заданиям. Алгоритм гарантирует получение ресурсов высокоприоритетными заданиями в минимально возможное время и в то же время допускает нарушение порядка очереди, позволяя низкоприоритетным заданиям занимать оставшиеся неиспользованными ресурсы, что способствует повышению коэффициента общей загрузки ресурсов.

Адаптация алгоритмов планирования, разработанных для локальных систем, в том числе и алгоритма BackFill, к условиям грид опирается на прогноз загрузки кластерных ресурсов. Планировщик грид должен размещать глобальные задания с учетом распределения ресурсов для локальных заданий кластеров, используя локальные расписания. Главный пункт адаптации — применение критерия, соответствующего соглашению о разделении ресурсов: ресурсы могут отводиться глобальному заданию лишь при условии, что плата за задание, назначенная пользователем при его запуске, не меньше определяемой по расписанию стоимости этих ресурсов.

Несмотря на перспективность использования расписаний, реализация основанного на них планирования имеет свои трудности. Содержащийся в локальном расписании прогноз получается в результате работы реального планировщика СПО, соответствует параметрам политики управления и в этом отношении является точным. Тем не менее, расписание представляет собой лишь ориентировочный (tentative) прогноз, поскольку ход реальной обработки будет совпадать с расписанием лишь при условии, что в кластер не будут поступать новые задания, и не будет меняться состав ресурсов. Есть и еще одна причина возможных отклонений: расписание строится, исходя из оценки времени выполнения, заданной в ресурсных запросах заданий, которая редко бывает точной.

В реальности непредсказуемые события, такие как преждевременное окончание заданий, будут влиять на расписание, изменяя его. Для того чтобы планирование в грид соответствовало состоянию ресурсов, локальные расписания должны регулярно обновляться и, соответственно, должен перестраиваться план распределения глобальных заданий. В связи с этим возникает вопрос о способе поставки расписаний, хотя в современных информационных службах грид в основном ис-

пользуется метод *опроса* (pull) источников данных, применительно к интерфейсу поставки расписаний более предпочтительным может быть метод *нотификации* (push) об изменениях в расписаниях кластеров или даже комбинация этих методов.

Укажем еще одну форму прогноза о состоянии ресурсов, которая в какой-то мере альтернативна расписаниям, — это список возможных аллокаций для отдельных глобальных заданий. В этом варианте планировщик грид опрашивает расширенные соответствующей функцией менеджеры кластеров, посылая им ресурсный запрос глобального задания и получая в ответ временные слоты, в которые оно может получить ресурсы. Списки возможных аллокаций могут получаться таким же образом, как и расписания, — путем моделирования работы менеджера кластера. Отметим также, что если возможные аллокации основаны на бинарном прогнозе (слот занят/свободен локальными заданиями), как в [40], то их едва ли можно использовать в грид с неотчуждаемыми ресурсами: по нашему мнению разделение ресурсов между локальными и глобальными заданиями должно быть более гибким.

5.2. Предварительное резервирование ресурсов

Предварительное резервирование ресурсов — один из немногих новых механизмов, реализованных в практических системах, хотя примеры реализаций немногочисленны: кластерный планировщик Maui и коммерческий вариант СПО PBS — PBSPro [41]. Соответствующее этому механизму расширение интерфейса СПО позволяет заказывать какое-то количество ресурсов на определенный в запросе интервал времени, отводя их исключительно для выполнения заданий некоторого пользователя или группы пользователей [42]. Заметим, что выполнение резервирований обеспечивается планировщиком кластера, который модифицирует распределение ресурсов таким образом, чтобы к моменту наступления времени резервирования заказанные ресурсы были свободны.

В связи с применением в грид предприняты усилия по разработке и стандартизации дистанционных интерфейсов предварительного резервирования, которые надстраиваются над локальными интерфейсами СПО. В этой связи важными представляются предложения, выработанные в специализированном комитете Global Grid Forum [43]. Стандартизация направлена на обеспечение независимости от интерфейсов конкретных СПО и, помимо того, унифицирует работу с различными видами ресурсов: помимо вычислительных, рассматриваются сетевые ресурсы и ресурсы хранения. Предложенный аппарат позволяет описывать разнообразные сценарии резервирования и включает такие, например, команды, как отказ и подтверждение резервирования, уточнение состава резервируемых ресурсов.

Очевидное применение предварительного резервирования для планирования в грид — гарантирование выделения ресурсов под построенные планировщиком аллокации. Однако сам по себе механизм резервирования не решает вопросы разделения ресурсов и не определяет, какие из запросов на резервирование должны удовлетворяться. Основная цель использования прогноза загрузки ресурсов как раз состоит в том, чтобы создать условия, при которых планировщик грид строит аллокации, удовлетворяющие соглашениям о разделении ресурсов. Запросы на ресурсы по таким аллокациям должны, естественно, удовлетворяться, и предварительное резервирование позволяет это сделать.

Сценарий использования предварительного резервирования состоит в следующем. Когда подходит время доставки какого-то задания, планировщик грид посылает в кластер запрос на резервирование, который в штатной ситуации должен, безусловно, выполняться. Затем происходит передача задания в кластер, и СПО обеспечивает его запуск в запланированное время и на запланированных ресурсах. При определенных условиях возможны отказы при резервировании. Это, однако, обнаруживается планировщиком, и задание никуда не перемещается, а поступает на перепланирование. В грид как распределенной среде важно обеспечить надежность доверительных отношений между планировщиком грид и кластером. Для этого при обработке в кластере запроса на резервирование может производиться проверка его корректности в смысле соответствия соглашению о разделении ресурсов.

6. Схема опережающего планирования

Представим схему опережающего планирования следующим псевдокодом.

```
1. Цикл планирования:{
2. Получение стоящих в очереди заданий грид;
3. Получение расписаний всех кластеров;
4. Цикл размещения заданий:{
5. Построение аллокации для задания;
6. Если время начала аллокации близко:
7. {резервирование; доставка задания в кластер;}}}
```

Планирование представляет собой циклический процесс, в каждом цикле которого строится ориентировочное распределение заданий грид по ресурсам и определяется подмножество заданий, для которых начало выполнения достаточно близко. Для этих заданий выполняется резервирование ресурсов, гарантирующее их выделение в соответствии с построенной планировщиком аллокацией, и инициируется доставка заданий в кластеры.

Первые два оператора цикла планирования получают из информационной базы стоящие в глобальной очереди задания (оператор 2) и множество локальных расписаний (3). Заполнение информационной базы осуществляется отдельным процессом, который может работать параллельно с процессом планирования. Каждый цикл планирования, однако, происходит при фиксированном состоянии информационной базы, а происходящие за время одного цикла изменения состояния учитываются на следующем. Локальные расписания объединяются в общий массив слотов, по которому ведется планирование глобальных заданий.

Собственно планирование заключается в построении аллокаций для заданий в порядке их приоритетов. Обозначим через N требуемое заданию количество процессорных узлов и T — длительность выполнения задания. Задача построения аллокации (5) состоит в том, чтобы найти такие N процессорных узлов и такой интервал времени $[t_0, t_0+T]$, на котором ресурсы этих процессорных узлов могут быть выделены данному заданию. Возможность выделения определяется следующими условиями:

- характеристики ресурсов каждого процессорного узла соответствуют ресурсному запросу задания,

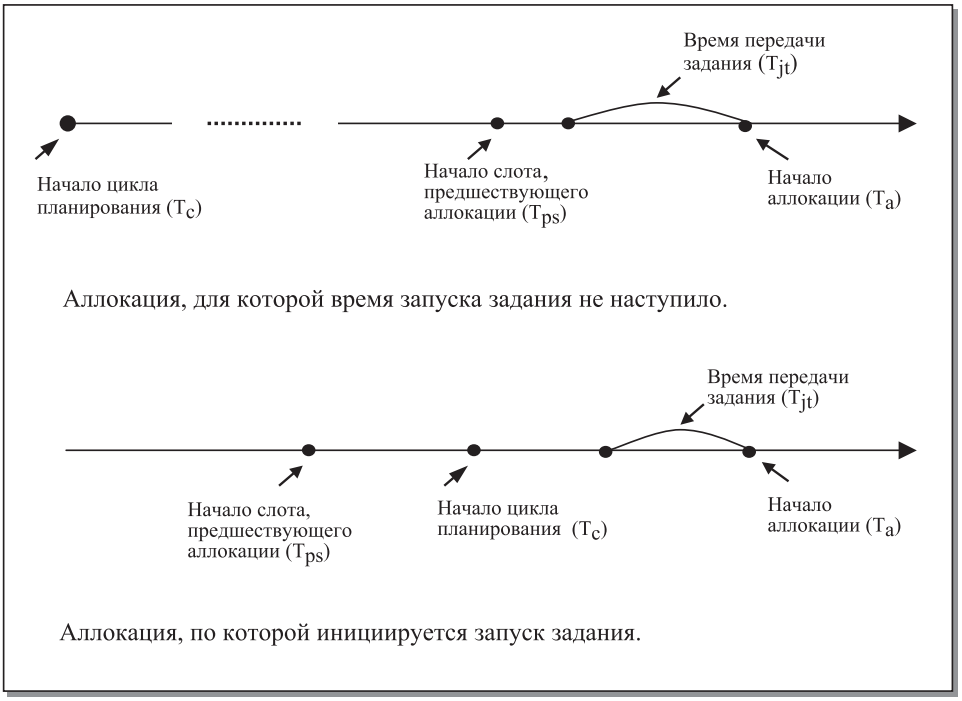

Рис. 3. Выбор времени запуска задания

- суммарная стоимость ресурсов, вычисляемая по слотам, попадающим в интервал действия аллокации, не превышает платы за ресурсы, назначенной пользователем.

После построения аллокации могут быть инициированы действия (7) по обеспечению запуска задания: предварительное резервирование ресурсов и передача задания в кластеры. Эти действия выполняются только для заданий, время начала аллокаций которых достаточно близко к моменту планирования. Для остальных заданий эффект построения аллокации сводится к тому, что слоты, по которым она построена, становятся полностью или частично недоступными на протяжении того же шага планирования для других глобальных заданий. Заметим, что на последующих шагах такие задания могут получить совсем другие ресурсы.

Способ отбора аллокаций (6), для которых необходимо инициировать запуск заданий, иллюстрирует рис. 3.

Выбор момента резервирования отражает компромисс между стремлением основывать реальное выделение ресурсов на оперативной информации о состоянии грид и необходимостью гарантировать запуск заданий. Если резервирования делаются рано, например, непосредственно в момент поступления заданий в глобальную очередь, то изменения состояния (появление новых заданий, отклонение хода обработки заданий от первоначального прогноза) за все время их нахождения в очереди никак не будут отражаться на порядке получения ресурсов, то есть не будет работать механизм приоритетов.

Что же касается минимального упреждения, необходимого для обеспечения запуска задания, то оно определяется двумя условиями. Первое условие: упреждение должно быть таким, чтобы ресурсы аллокации не оказались занятыми ло-

кальными заданиями. При этом надо учесть, что эти ресурсы могут освободиться раньше прогнозируемого в расписании времени. Если происходит такое "преждевременное" освобождение ресурсов, на них может начать выполняться локальное задание, окончание которого происходит после начала аллокации. В связи с этим первое условие начала доставки задания задается неравенством: $T_{ps}<T_c$, где T_{ps} — начало слота, непосредственно предшествующего аллокации на тех же ресурсах, T_c — время начала цикла планирования.

Второе условие заключается в том, что должно хватать времени для передачи задания на исполнительные ресурсы. Процесс передачи включает пересылку всех его входных файлов, так что время передачи может быть значительным. Доставка задания инициируется, как только (T_a-T_c) становится меньше T_{jt}, где T_a — время начала аллокации, T_{jt} — время передачи задания.

Первое из сформулированных условий позволяет с высокой вероятностью рассчитывать, что ресурсы для построенных планировщиком аллокаций будут выделены в кластере. Тем не менее, отказы в резервировании возможны: ресурсы все же могут оказаться занятыми локальным заданием. Такое, однако, может произойти лишь в том случае, когда в промежуток между построением аллокации и резервированием аварийно (в момент старта) завершается несколько заданий, предшествующих началу аллокации. Эта ситуация рассматривается как исключительная, а задание поступает на перепланирование.

6.1. Масштабируемость системы планирования

Способность рассматриваемой схемы планирования обслуживать большое количество компьютеров — масштабируемость — является ограниченной, как и у всякой системы с централизованной архитектурой. В связи с этим важными представляются вопросы оценки пределов масштабируемости, диагностики адекватности функционирования планировщика, а также разработки способов повышения масштабируемости.

В штатном режиме работы планировщик, обслуживая поток поступающих в глобальную очередь заданий, успевает своевременно реагировать на изменения, происходящие в кластерах. Заметим, что в отличие от локальных СПО, планировщик не обрабатывает каждое отдельное событие. Вместо этого он периодически получает от кластеров обновления расписаний и на каждом цикле планирования перестраивает заново аллокации для глобальных заданий. Нарушение штатного режима происходит, когда отставание (T_{su}-T_{sp}), где T_{sp} — время производства расписания в кластере, T_{su} — 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 по отношении к концу цикла планирования.

Для каждого кластера значение TTL может устанавливаться индивидуально, исходя из смысла этого параметра: значение TTL определяет, с каким предельным отставанием планировщик должен реагировать на изменения состояния кластера. При большом TTL получение ресурсов для вновь поступающих кластерных заданий будет иметь тенденцию к задержке, так как полагающиеся им ресурсы могут занять глобальные задания. Однако задержка всегда ограничена, и, если, например, TTL устанавливается равным минимально возможной длительности задания, то локальное задание пропустит не более одного глобального задания вне очере-

ди, 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. Исходя из этого условия, дадим оценку максимальной производительности планировщика, отмечая, что анализ всей системы планирования требует учета ряда дополнительных факторов: скорости генерации локальных расписаний, производительности сетевых операций по доставке расписаний и управлению заданиями.

В цикле планирования при построении аллокаций выполняется сопоставление глобальных заданий, находящихся в очереди, со слотами агрегированного по всем кластерам расписания. Обозначая количество глобальных заданий Nz и число слотов Ns, представим длительность цикла планирования в виде CP=A1·Nz·Ns. Здесь A1 — время одной операции сопоставления, которое мы оцениваем A1=10^{-2} мск. Чтобы получить оценку зависимости CP от числа компьютеров грид N, будем исходить из того, что Nz=C2·N и Ns=C3·N, где константы C2, C3<5 (см. ниже). Тогда CP= A1·C2·C3·N^2. При N=1000 CP<10^{-2}·25·10^6 мск = 25·10^4 мск = 4 мин.

С учетом вышеприведенного определения TTL должно выполняться неравенство TTL>2·CP, и полагая TTL = L, где L минимальная длина задания, получим ограничение масштабируемости: N< $\sqrt{L/(2 \cdot A1 \cdot C2 \cdot C3)}$. При L=30 мин. планировщик способен обслуживать несколько тысяч компьютеров.

В этих выкладках нуждаются в разъяснениях предположения о количестве глобальных заданий и количестве слотов. Смысл первого предположения (Nz=C2·N) в том, что количество запускаемых в грид заданий должно быть сбалансировано с имеющимися ресурсами: как предполагается в оценке, в пересчете на один компьютер длина очереди не превышает 5 заданий.

Второе предположение (Ns= C3·N) исходит из того, что поставляются не полные, а ограниченные расписания, в которых прогнозируется распределение только тех локальных заданий, которые находятся в голове очереди своего кластера. Если планирование выполняется только для однопроцессорных заданий, длина расписаний может быть ограничена двумя слотами на каждом компьютере — такого прогноза хватит, чтобы построить аллокацию и заранее произвести резервирование ресурсов. В случае многопроцессорных заданий теоретически нужно иметь расписания полного объема. Тем не менее, при выборе C3, равном небольшому целому числу (4–5), будет, во-первых, гарантировано выделение ресурсов самым приоритетным глобальным заданиям, и, во-вторых, снижение эффективности алгоритма Backfill, выражающееся в недополучении ресурсов менее приоритетными заданиями, как мы рассчитываем, будет незначительным.

7. Сравнение с современными планировщиками

Не претендуя на полный обзор довольно многочисленных реализаций систем управления заданиями в грид, появившихся в последнее время, рассмотрим их возможности планирования.

Большинство разработок предназначены для обслуживания кластеризованных ресурсов, в том числе система WMS [8], наиболее крупного на сегодняшний день проекта EGEE [44], грид которого имеет мировой масштаб. Брокер системы WMS реализует централизованную схему планирования, работающую в двух режимах распределения заданий: "жадного и ленивого" ("ленивый" режим был впервые реализован в проекте Alien [45]). В "жадном" режиме распределение заданий про-

исходит без глобальной очереди: поступившее в брокер задание сразу же отправляется в некоторый кластер. Для выбора кластера используется информационная база, содержащая данные о текущем состоянии всех ресурсов грид. Задания, запускаемые пользователями в "ленивом" режиме, поступают в глобальную очередь. Кластеры, которые имеют свободные ресурсы, информируют брокер об их характеристиках, а тот, выбирая из очереди подходящее задание, передает его в кластер.

В обоих режимах происходит сопоставление характеристик ресурсов и заданий на предмет выяснения возможности выполнения. Информационная база "жадного" режима, построенная на информационной службе MDS [19] системы Globus Toolkit [14], недостаточно полна: она не содержит данных, описывающих процессорные узлы, так что этот режим может использоваться только в грид с однородными ресурсами. Сопоставление характеристик ресурсов и заданий в "ленивом" режиме производится на основе метода matchmaking [15] и не имеет такого недостатка.

Основное решение планирования — выбор исполнительных ресурсов — в "жадном" режиме делается по интегральной характеристике состояния кластера: выбирается тот, у которого в локальной очереди меньше всего заданий. Как было показано, это потенциально ведет к зависанию заданий, тем более, что глобальная очередь в этом режиме не поддерживается и распределение происходит в момент поступления задания в брокер. Подход "ленивого" режима более аккуратный — задания грид находятся в очереди, а кластер запрашивает у брокера новое задание в моменты времени, определяемые политикой, заданной администратором. Однако вопрос о том, какой должна быть эта политика остается открытым. Простейший способ — запрашивать задание при пустой локальной очереди — не разрешает проблем неоднородного кластера и неотчуждаемых ресурсов.

Система управления заданиями ARC [27] проекта NorduGrid представляет интерес как пример децентрализованной архитектуры планирования. В целом схема планирования в ARC аналогична "жадному" режиму брокера WMS, однако планировщик здесь входит в состав программного обеспечения каждого рабочего места пользователя, а планирование и передача задания в кластер выполняются непосредственно с места выдачи запроса. Преимущества децентрализованной архитектуры известны — это отсутствие "критической точки", живучесть и масштабируемость. Отметим также, что в данной архитектуре планирование производится отдельно для каждого задания и принимаемые решения могут оказаться не согласованными друг с другом.

В некоторых системах находят применение два новых механизма — предварительное резервирование ресурсов и миграция заданий. Сценарий Moab Grid Scheduler [40] выглядит следующим образом:

- задания грид поступают в глобальную очередь,
- планировщик посылает запросы в кластеры и получает от них время возможных аллокаций,
- исходя из полученной информации, выбирается кластер, производится резервирование ресурсов, задание отправляется в кластер.

Moab, так же как и CSF (Community Scheduler Framework) [46], который в свою очередь поддерживает резервирование в службе планирования, не дает все-таки полного решения, предполагая, что кластерное обеспечение имеет средства

для принятия решений о том, когда могут быть отведены ресурсы для резервирования. В стандартных СПО таких средств нет, и их создание составляет отдельную задачу.

Системы GridWay [47] и GridLab Resource Management System (GRMS) [48] используют механизм перераспределения заданий — миграцию. В GridWay после того как глобальное задание отправлено в кластер, специальная служба осуществляет его мониторинг. В случае, если задание не запущено на счет в локальном менеджере ресурсов за некий интервал времени, происходит перепланирование данного глобального задания, и оно перемещается на другие ресурсы.

Миграция представляет собой мощный механизм, который способен расширить возможности планирования. Однако его применение сопряжено с большими временными затратами на перемещение файлов заданий, так что миграция, по-видимому, может рассматриваться как дополнение к качественному способу планирования.

Приведем также ряд исследовательских работ, в которых рассматриваются вопросы, близкие к теме данной работы. Идея использования прогноза для выбора ресурсов грид не является новой. Одна из первых работ такого рода — система Gallop [49], планирование в которой осуществляется на основе предсказания свободной вычислительной мощности по статистическим данным, для чего на каждом кластере Gallop устанавливается специальная компонента (Prophet), вырабатывающая локальный прогноз. Значительное количество публикаций посвящено прогнозированию загрузки ресурсов на основе статистической информации (см., например, [50]).

Детерминированное планирование рассматривается в работе [12], где исследуется эффективность использования предварительного резервирования для многопроцессорных заданий. В предложенной модели планирования прогноз ограничен одним шагом — временем освобождения ресурсов, которое оценивается по заказанному времени выполнения заданий.

В [51] обосновывается необходимость расширения функциональности современных систем планирования функциями предварительного резервирования ресурсов и составления расписаний запуска заданий. Авторы проводят различие между системами планирования двух типов: традиционными — Queuing systems и системами, основанными на расписаниях — Planning systems. В то время как первые распределяют задания, исходя из текущего состояния ресурсов, вторые основываются на полноценном плане на будущее. По мнению авторов, последний подход, реализованный в системе CCS [52], способствует применению аппарата предварительного резервирования и открывает перспективу для решения задачи управления параллельными заданиями. Разделяя эти общие положения, отметим, что в архитектуре CCS только один уровень управления: все задания распределяются через CCS.

Наиболее близко к нашей работе стоит описанный в [34] планировщик Ursala. Для него предложен способ размещения заданий, в том числе параллельных, основанный на механизмах предварительного резервирования и списках возможных аллокаций. Получая такие списки от кластеров, Ursala подбирает такое временное окно, в котором свободных ресурсов досточно для запуска задания, и резервирует их. В работе отмечается способность локального планировщика Maui строить расписание загрузки ресурсов кластера. Сравнивая нашу работу с Ursala, отметим, что мы не ограничиваемся размещением отдельных заданий: предложены динамическая схема приоритетного планирования очереди заданий и конкретный

способ балансировки потоков локальных и глобальных заданий, основанный на плате за ресурсы.

В работе [33] представлена распределенная архитектура планирования с множественными экземплярами планировщиков, управляющих непересекающимися множествами ресурсов. Подбор ресурсов для поступающих в планировщик заданий осуществляется в два этапа: сначала ресурсы ищутся внутри множества, непосредственно связанного с данным планировщиком, а если там они не обнаружены, то поиск продолжается путем передачи задания соседним планировщикам. При выборе ресурсов используется экономическая модель: ресурсы подбираются не только по ресурсному запросу, но и с учетом целевых функций пользователя и владельца ресурсов. Для определения возможного времени старта задания используются расписания, полученные от планировщиков. Как и в описанной выше системе CCS, в этой схеме подразумевается наличие только одного потока заданий, поступающего на каждый планировщик.

Заключение

В работе рассмотрена совокупность принципов: предварительное резервирование ресурсов, предсказание загрузки ресурсов, приоритетное обслуживание заданий, разделение ресурсов между грид и владельцами, на базе которых разработан метод опережающего планирования. Основным достоинством этого метода является способность гарантировать точное время запуска одно- и многопроцессорных заданий в условиях кластеризованных ресурсов, используемых совместно с их владельцами.

Настоящая публикация является развитием работ [53, 54]. Программная реализация системы планирования находится в следующем состоянии. Завершен действующий прототип для управления однопроцессорными заданиями, что позволяет приступить к исследованию характеристик системы: производительности, качества планирования и масштабируемости. В стадии разработки находятся расширения базовых алгоритмов механизмами, адаптирующими параметры планирования к меняющейся нагрузке и масштабам грид. Второе направление — обслуживание многопроцессорных заданий — получило развитие в модификации алгоритма Backfill применительно к двухуровневому планированию [55].

Литература

[1] Ian Foster. What Is The GRID? A Three Point Checklist. GRID Today, July 22, 2002: Vol. 1 No. 6.
`http://www.gridtoday.com/02/0722/100136.html`.
Русский перевод:
`http://www.gridclub.ru/library/publication.2004-11-29.5830756248`

[2] Коваленко, В.Н. Оценка возможностей программных платформ грид: труды международной конференции "Распределенные вычисления и Грид-технологии в науке и образовании" (Дубна, 29 июня-2 июля 2004 г.) / В.Н. Коваленко, Д.А. Корягин. Дубна: 11–2004–205, ОИЯИ, 2004, С. 128–133.
`http://www.gridclub.ru/library/publication.2005-03-17.0023427070`

[3] Приложения, работающие в грид EGEE.
`http://public.eu-egee.org/files/Application%20running%20on%20EGEErusnew.pdf`

[4] LCG middleware documentation.
http://grid-deployment.web.cern.ch/grid-deployment/cgi-bin/index.cgi?var=documentation
[5] gLite — Lightweight Middleware for Grid Computing. http://www.glite.org
[6] The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration / I. Foster [et al.].
http://www.globus.org/research/papers/ogsa.pdf.
Русский перевод:
http://www.gridclub.ru/library/publication.2004-11-29.8307957187
[7] Modeling Stateful Resources with Web Services / I. Foster [et al.]. http://www-106.ibm.com/developerworks/library/ws-resource/ws-modelingresources.pdf. Русский перевод:
http://www.gridclub.ru/library/publication.2005-04-25.6831747572/view
[8] Workload Management System User and Reference Guide,
http://edms.cern.ch/file/572489/1/WMS-guide.pdf
[9] Grid Resource Management: State of the Art and Future Trends. Edited by J.Nabrzyski, J.M.Schopf, J.Weglarz. Kluwer Academic Publishers, 2003.
[10] Coffman, E.G. Operating Systems Theory / E.G. Coffman, P.J. Denning. 1973, Prentice-Hall, Englewood Cliffs.
[11] D.A.Lifka. The ANL/IBM SP Scheduling System.
http://www.tc.cornell.edu/~lifka/
[12] Weissman, J.B. Ensemble Scheduling: Resource Co-Allocation on the ComputationalGrid / J.B. Weissman, P. Srinivasan.
http://www-users.cs.umn.edu/~jon/papers/grid2001_ens.ps
[13] The Globus Resource Specification Language RSL v1.0.
http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html
[14] Globus Toolkit. htpp://www.globus.org
[15] Raman, R. Matchmaking: An extensible framework for distributed resource management / R. Raman, M. Livny M. Solomon. Cluster Computing, 2(2), 1999.
[16] Condor. http://www.cs.wisc.edu/condor/
[17] Job Description Language HowTo,
http://server11.infn.it/workload-grid/docs/DataGrid-01-TEN-0102-0_2-Document.doc
[18] Condor-G: A computation management agent for multiinstitutional Grids / J. Frey [et al.] Cluster Computing, 5(3):237–246, 2002.
[19] Grid information services for distributed resource sharing / K. Czajkowski [et al.] // Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), August 2001.
[20] Relational Grid Monitoring Architecture (R-GMA). A.Cooke et al, UK e-Science All Hands Conference, Nottingham, September 2003.
http://www.r-gma.org/pub/ah03_148.pdf
[21] Portable Batch System. http://www.openpbs.org
[22] Sun Grid Engine. http://www.sun.com/software/gridware/sge.html
[23] Platform LSF.
http://www.platform.com/Products/Platform.LSF.Family/Platform.LSF/
[24] The Entropia Virtual Machine for Desktop Grids / B. Calder [et al.]. 2003.
[25] Scheduling in Bag-of-Task Grids: The PAUA Case. W.C. Francisco Brasileiro Lauro Costa Daniel Paranhos Elizeu Santos-Neto Nazareno Andrade. http://walfredo.lsd.ufcg.edu.br/papers/PAUA.pdf.

[26] Коваленко, В.Н. Организация ресурсов грид / В.Н. Коваленко, Д.А. Корягин. Препринт ИПМ РАН, №63. М., 2004, 25 с.

[27] The NorduGrid architecture and tools / P. Eerola [et al.] Computing in High Energy and Nuclear Physics, 24–28 March 2003, La Jolla, California. http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOAT003.pdf

[28] Abraham, A. Nature's Heuristics for Scheduling Jobs on Computational Grids, The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), December 14–16, 2000 / A. Abraham, R. Buyya, B. Nath. Cochin, India. http://www.buyya.com/papers/nhsjcg.pdf

[29] Buyya, R. A case for economy Grid architecture for service-oriented Grid computing / R. Buyya, J. Giddy, D. Abramson // Proceedings of the Heterogeneous Computing Workshop, April 2001.

[30] Buyya, R. An Economy Grid Architecture for Service-Oriented Grid Computing / R. Buyya, J. Giddy, D. Abramson. 10th IEEE International Heterogeneous Computing Workshop (HCW 2001), San Francisco, USA, April 2001.

[31] Ernemann, C. Economic scheduling in Grid computing / C. Ernemann, V. Hamscher,R. Yahyapour. In D. Feitelson and L. Rudolph, editors, Job Scheduling Strategies for Parallel Processing (Proceedings of the Eighth International JSSPP Workshop; LNCS #2537). Springer-Verlag, 2002. P. 129–152.

[32] How Sun Grid Engine, Enterprise Edition 5.3 Works. http://wwws.sun.com/software/gridware/sgeee53/wp-sgeee/wp-sgeee.pdf

[33] Evaluation of job-scheduling strategies for grid computing / V. Hamscher [et al.]. Lecture Notes in Computer Science, 1971:191–202, 2000.

[34] The Performance Impact of Advance Reservation Meta-scheduling / Q. Snell [et al.] Computer Science Department Brigham Young University Provo, Utah 84602-6576, 2000, http://supercluster.org/research/papers/ipdps2000.pdf

[35] Schwiegelshohn, U. Attributes for Communication Between Grid Scheduling Instances / U. Schwiegelshohn,R. Yahyapour. // "Grid Resource Management. State of the Art and Future Trends". Edited by J.Nabrzyski, J.M.Schopf, J.Weglarz. Kluwer Academic Publishers, pp. 41–52.

[36] Коваленко, Е.И. Расширение возможностей кластерных систем управления для информационного обслуживания Грид-диспетчера: тезисы докладов международной конференции "Распределенные вычисления и Грид-технологии в науке и образовании" / Е.И. Коваленко, О.Н. Шорин. Дубна, 29 июня – 2 июля 2004 г., С. 56.

[37] Maui scheduler. http://www.supercluster.org/maui

[38] Krallmann, J. On the design and evaluation of job scheduling systems / J. Krallmann, U. Schwiegelshohn, R. Yahyapour. In D. Feitelson and L. Rudolph, editors, Job Scheduling Strategies for Parallel Processing (Proceedings of the Fifth International JSSPP Workshop; LNCS # 1659), pages 17–42. Springer-Verlag, 1999.

[39] Feitelson, D.G. Utilization and Predictability in Scheduling the IBM SP2 with Backfilling / D.G. Feitelson A.M. Weil. // Procedings of IPPS/SPDP 1998, pages 542—546. IEEE Computer Society, 1998.

[40] Moab Grid Scheduler. http://www.clusterresources.com/products/mgs/specoverview.shtml

[41] PBS Professional. http://www.pbspro.com/

[42] Коваленко, В.Н. Управление заданиями в распределенной среде и протокол резервирования ресурсов / В.Н. Коваленко, А.В. Орлов. Препринт ИПМ РАН, №1, М., 2002. 25 с.

[43] Roy, A. Advance Reservation API / A. Roy V. Sander. Technical Report GFD-E.5, Global Grid Forum (GGF), 2002.

[44] Проект Enabling Grids for E-SciencE (EGEE), http://www.eu-egee.org

[45] Buncic, P. The AliEn system, status and perspectives / P. Buncic, A.J. Peters, P. Saiz. // Computing in High Energy and Nuclear Physics, 24–28 March 2003, La Jolla, California.
http://www.slac.stanford.edu/econf/C0303241/proc/papers/MOAT004.pdf

[46] Open source metascheduling for Virtual Organizations with the Community Scheduler Framework (CSF).
http://www.cs.virginia.edu/~grimshaw/CS851-2004/Platform/CSF_architecture.pdf

[47] GridWay. http://www.gridway.org/int_arch.php

[48] GridLab Resource Management System.
http://www.gridlab.org/WorkPackages/wp-9

[49] Weissman, J.B. Gallop: The Benefits of Wide-Area Computing for Parallel Processing / J.B. Weissman // Journal of Parallel and Distributed Computing, V. 54(2), November 1998.

[50] Yang, L. Homeostatic and tendency-based CPU load predictions / L. Yang, I. Foster, and J.M. Schopf. In Proceedings of International Parallel and Distributed Processing Symposium (IPDPS), 2003.

[51] Scheduling in HPC Resource Management Systems: Queuing vs. Planning / M. Hovestadt.
http://www.uni-paderborn.de/pc2/papers/files/409.pdf

[52] Keller, A. Anatomy of a Resource Management System for HPC Clusters / A. Keller and A. Reinefeld. In Annual Review of Scalable Computing, vol. 3, Singapore University Press, 2001. P. 1–31.

[53] Метадиспетчер: реализация средствами метакомпьютерной системы Globus / С.А. Богданов [и др.]. Препринт ИПМ РАН, №30. М., 2001. 23 с.

[54] Resource manager for GRID with global job queue and with planning based on local schedules / V.N. Kovalenko [et al.] Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. Volume 502, Issues 2–3 , 21 April 2003. P. 411–414.

[55] Коваленко, В.Н. Использование алгоритма Backfill в грид: труды международной конференции "Распределенные вычисления и Грид-технологии в науке и образовании" (Дубна, 29 июня – 2 июля 2004 г.) / В.Н. Коваленко, Д.А. Семячкин. Дубна: 11–2004–205, ОИЯИ, 2004, сс. 139–144.
http://www.gridclub.ru/library/publication.2004-12-27.6965428621

Поступила в редакцию 16/*I*/2006;
в окончательном варианте — 28/*II*/2006.

BASIC PRINCIPLES OF LOOKAHEAD SCHEDULING METHOD FOR COMPUTATIONAL GRID[3]

© 2005 V.N. Kovalenko, E.I. Kovalenko, E.Z. Ljubimsky, D.A. Koryagin[4]

One of the most important program technologies that forms the base of globally distributed computer infrastructure — grid, is resource virtualization. In this paper we propose a new method of virtualization called Lookahead scheduling. The characteristic feature of the method is its ability to build up in advance a plan of resource allocations. Such "predictive" scheduling even in the conditions of shared resources has a property of the determinacy — realization of the plan is provided by means of advance reservation. The determinacy makes Lookahead method suitable for a multiple-processor job management.

Paper received 16/*I*/2005.
Paper accepted 28/*II*/2006.

[3]Communicated by Dr. Sci. (Phys. & Math.) Prof. V.A. Saleev.

[4]Kovalenko Victor Nikolaevich (kvn@keldysh.ru), Kovalenko Eugenija Ivanovna (kei@keldysh.ru), Koryagin Dmitriy Aleksandrovich (koryagin@keldysh.ru), Ljubimsky Eduard Zinovievich (ljubimsk@keldysh.ru), Keldysh Institute for Applied Mathematics RAS, Moscow, 125047, Russia.