УДК 681.31:323

РАСПРЕДЕЛЕННЫЙ МЕТАПЛАНИРОВЩИК ГРИД

В. В. Корнеев[1], **Д. В. Семенов**[1]

Рассматривается подход к планированию ресурсов на базе модели с распределенной очередью и реализующие его алгоритмы функционирования распределенного планировщика грид-среды.

Ключевые слова: грид, распределенное планирование заданий, архитектура грид для параллельных вычислений.

1. Введение. Грид-технологии [1] предоставляют программные решения для построения грид как особого класса сетей, объединяющих совокупность вычислительных установок различной архитектуры (вычислительных систем, отдельных компьютеров) и выделяющих по запросу, не содержащему указания адресов ресурсов (например MAC (Media Access Control)- и IP (Internet Protocol)-адресов), требуемые пользователю ресурсы из совокупных ресурсов сети с предотвращением несанкционированного доступа к остальным ресурсам, в том числе выделенным другим пользователям.

Все грид имеют проблемную ориентацию, например коллективная обработка результатов физических экспериментов, предоставление вычислительных услуг и др. Грид-сеть, предназначенная для выполнения вычислительных заданий множества пользователей, должна, во-первых, перераспределять задания пользователей между входящими в нее территориально разнесенными вычислительными системами (ВС) и, во-вторых, согласованно выделять вычислительные модули (ВМ), возможно разных ВС, для исполнения параллельных программ. Будем называть грид с такими свойствами сетевой средой распределенных вычислений [2, 3] или для краткости грид-средой. Критериями для выбора алгоритма планирования грид-среды могут служить максимизация пропускной способности, выражающаяся в числе выполненных заданий в единицу времени, требуемое качество обслуживания, характеризующееся такими параметрами, как время отклика, гарантированная пропускная способность, доступность и отказоустойчивость. Определяющими требованиями к грид-среде, создаваемой в Межведомственном суперкомпьютерном центре (МСЦ) РАН, служат увеличение пропускной способности совокупности ресурсов за счет исключения простоя одних ресурсов при перегруженности других ресурсов, а также предоставление вычислительного ресурса, превышающего ресурсные возможности отдельных ВС, для исполнения параллельных программ масштабных вычислительных задач, допускающих эффективное выполнение на ресурсах нескольких ВС.

Структура грид-среды показана на рис. 1. Каждая ВС имеет управляющую машину (УМ), в качестве которой может выступать сама ВС, как, например, ВС_3 на рис. 1. УМ всех ВС объединены сетью, которая позволяет передавать программы и данные между ними и выполнять удаленный запуск программ. Вычислительные модули отдельных ВС или даже нескольких разных ВС могут быть объединены одной или несколькими высокоскоростными сетями, например Myrinet, Infiniband, Quadrics [2], для выполнения обменов данными при исполнении параллельных программ.

В каждой ВС, входящей в состав грид-среды, на УМ функционирует локальная система пакетной обработки (СПО) заданий, не обязательно одна и та же на всех ВС. Известны более двадцати СПО, из которых наиболее популярны свободно распространяемые PBS (Portable Batch System) и Condor, а также коммерческие LoadLeveler, PBS Professional и LSF (Load Sharing Facility). К значимым отечественным разработкам в этой области следует отнести СУПЗ [4], созданную в ИПМ им. М. В. Келдыша РАН в кооперации с ИММ УрО РАН и НИИ "Квант", и CLEO [5], разработанную в НИВЦ МГУ им. М. В. Ломоносова. Применение СПО позволяет в каждой ВС перейти от работы с индивидуальными ВМ к работе с единым пулом вычислительных модулей в режиме пакетной обработки заданий, в том числе параллельных. Используя интерфейс СПО, можно: помещать задания в общую очередь СПО, модифицировать, удалять задания из очереди и снимать с выполнения, а также получать информацию о состоянии заданий. СПО автоматически распределяет задания по ВМ с учетом их загрузки, освобождает занятые ресурсы после завершения задания, в том числе по истечении времени исполнения и при аварийном завершении, доставляет результаты пользователю, а также обеспечивает разграничение прав пользователей и защиту вычислительных ресурсов от несанкционированного доступа.

[1] Научно-исследовательский институт "Квант", 4-й Лихачевский пер., 15, 125438, Москва; В. В. Корнеев, зам. директора по научной работе, e-mail: korv@rdi-kvant.ru; Д. В. Семенов, ст. науч. сотр.

Для включения ВС разных владельцев в состав грид-среды без вмешательства в существующие программные средства и режим работы этих ВС необходимо использовать дополнительный уровень программного обеспечения (ПО), реализующий функции системы управления (СУ) грид-среды. СУ грид-среды предоставляет возможность постановки заданий пользователей в общую очередь грид-среды, запуска заданий посредством СПО на свободных ресурсах любых одной или нескольких ВС, мониторинга грид-среды, обеспечения отказоустойчивых и защищенных от несанкционированного доступа вычислений. При этом каждая ВС получает задания как непосредственно от пользователей этой ВС, так и посредством ПО СУ грид от пользователей других ВС. СУ переносит, в соответствии с принятыми в ней алгоритмами планирования, задания из очереди грид-среды в очередь локальных СПО этих ВС.

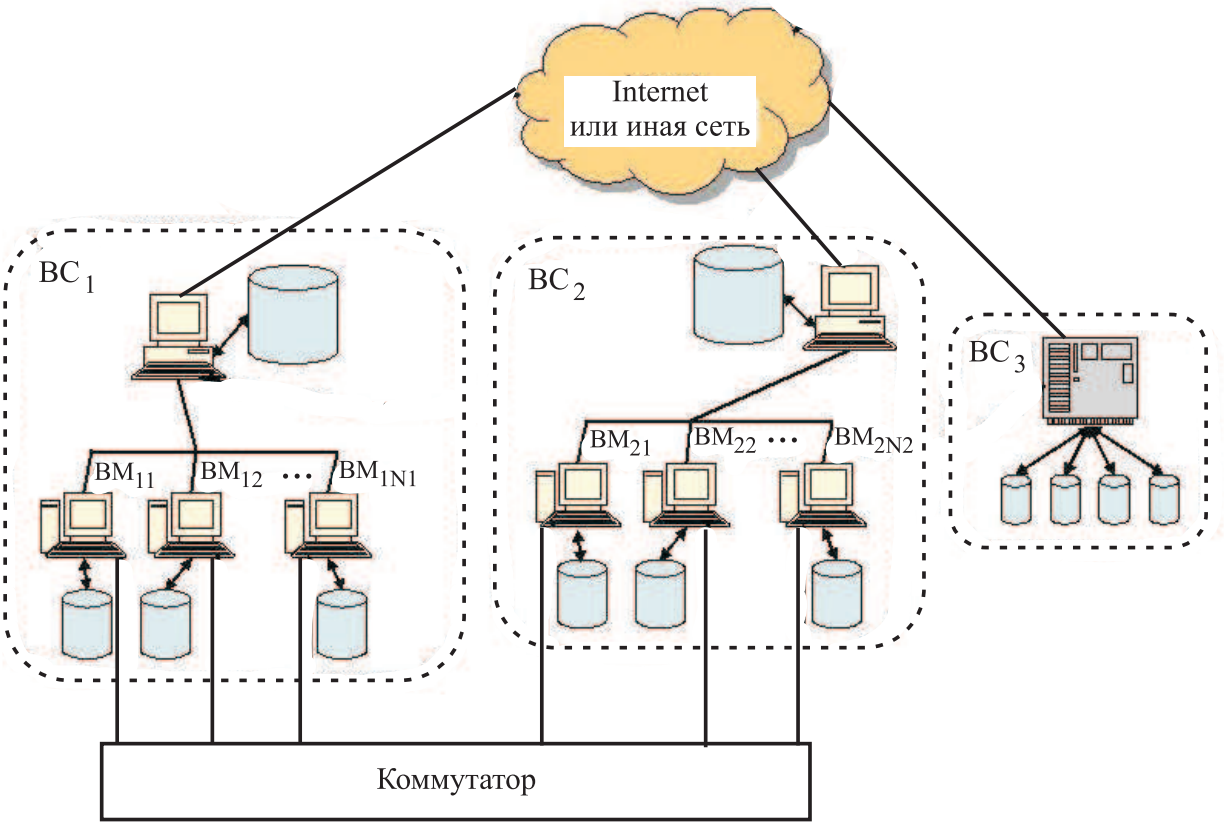

Рис. 1. Структура грид-среды

Наиболее исследованы и практически реализованы системы управления вычислительными ресурсами, базирующиеся на модели с разделением ресурсов [6]. В рамках этой модели задания пользователей поступают в одну очередь, разделяемую всеми процессорами параллельной системы. Когда процессор становится свободным, он сам берет задание для выполнения из общей очереди или это делает некий системный процесс, отслеживающий состояние процессоров. Применительно к грид-системам этот подход использован в ИПМ им. М. В. Келдыша РАН [7] при создании метадиспетчера и в проекте GridWay [8]. В последнем назначение заданий на ресурсы выполняется диспетчером, представление о функционировании которого можно получить из перечня задаваемых параметров: периода между двумя итерациями планирования, периода поиска новых свободных ресурсов, периода обновления сведений по каждому ресурсу, периода опроса состояния заданий, максимального количества заданий, которые будут обработаны планировщиком за один шаг планирования.

Однако в грид-среде, объединяющей много ВС и имеющей существенно разнящиеся пропускные способности каналов между ВС и внутри ВС, невозможно в приемлемое время получить полное и точное описание текущего состояния вычислительных ресурсов и находящихся в системе заданий для оптимального планирования одной общей очереди. Поэтому в больших грид-средах необходимо использовать распределенные метапланировщики на базе модели с распределенной системой очередей. В данной работе рассматривается подход к планированию ресурсов на базе модели с распределенной общей очередью и реализующие его алгоритмы функционирования распределенного планировщика грид-среды, созданной

в МСЦ РАН [9, 10].

Статья имеет следующую структуру. Во втором разделе представлена архитектура системы управления грид-среды. В третьем разделе рассмотрен ряд эвристических алгоритмов децентрализованного планирования заданий. Четвертый раздел содержит результаты экспериментов по исследованию эффективности предложенной децентрализованной системы управления грид-среды.

2. Распределенный метапланировщик. Задания, поступающие в ВС, должны регистрироваться в одной из очередей распределенной системы очередей. Каждая очередь обслуживается собственным локальным планировщиком, принимающим для очередного задания из очереди одно из трех решений: задание может быть запланировано для выполнения на ресурсах одной или нескольких ВС, либо оставлено в очереди для его последующего планирования, либо передано в другую очередь. При создании распределенного метапланировщика необходимо, с одной стороны, обеспечить возможность независимого одновременного планирования заданий, находящихся в разных очередях, локальными планировщиками, а с другой стороны, согласованно использовать ресурсы грид-среды.

Представленный в настоящей статье подход предлагает разрешение этого противоречия выделением домена ресурсов грид-среды каждой очереди для независимого планирования локальным планировщиком в этом домене. Если выделить в отдельные домены ресурсы каждой ВС грид-среды и создать еще один домен, включающий ресурсы всех ВС, то образуется иерархия очередей. В этой иерархической очереди возможно согласованное распределение ресурсов между заданиями при выполнении следующего алгоритма работы с очередями: передача заданий между очередями нижнего уровня иерархии возможна только через очередь верхнего уровня, используемую только для планирования заданий между очередями нижнего уровня.

Каждый выделенный домен управляется компонентом СУ грид-среды — менеджером СУ. Менеджер имеет набор структур данных, необходимых для локального планировщика:

— информационную систему (ИС), содержащую таблицу ресурсов управляемого домена и описание, возможно редуцированное, состояния грид-среды в целом;

— очередь заданий, подлежащих планированию.

Основными функциональными процессами менеджера служат:

— собственно локальный планировщик, вырабатывающий решения о назначении заданий на ресурсы домена или пересылке их в другую очередь на основе данных ИС и очереди заданий;

— процесс, поддерживающий актуальное состояние данных ИС, когерентное с ИС других локальных планировщиков;

— служебные процессы, обеспечивающие отказоустойчивость и информационную безопасность (защиту от несанкционированного доступа к ресурсам).

Менеджер СУ, локальный планировщик которого принимает решение о назначении задания непосредственно на ресурсы ВС, будем называть менеджером М1 или менеджером 1-го уровня. Соответственно, менеджер, планировщик которого распределяет задания между локальными планировщиками, будем называть менеджером М2 или менеджером 2-го уровня. Менеджеры М1 передают задания в очередь системы пакетной обработки ВС или менеджеру М2. На УМ каждой ВС обязательно запущен менеджер М1 этой ВС. Количество менеджеров М2 может быть от одного и более в зависимости от требуемых показателей надежности и пропускной способности грид-среды. Менеджеры СУ могут выполняться на управляющих машинах ВС или на специально выделенных компьютерах ВС. Информационные связи между менеджерами образуют ациклический граф. Взаимодействие между менеджерами выполняется по IP-адресам и номерам портов, используемых менеджерами.

В локальных планировщиках менеджеров СУ могут применяться различные алгоритмы выделения ресурсов: от решения оптимизационных задач до эвристических алгоритмов, что позволяет учитывать специфику неоднородности компонентов грид-среды.

В [11] предложен протокол параллельного распределения ресурсов менеджерами иерархической структуры, для которого доказано отсутствие дедлоков, вызванных конфликтом взаимной блокировки заданий из-за частичного одновременного выделения разными менеджерами ресурсов заданиям и невозможности продолжения выполнения этих заданий по причине нехватки ресурсов каждому из них без освобождения ресурсов другим заданием.

Иерархическая организация менеджеров СУ грид-средой позволяет:

— обеспечить отсутствие дедлоков при распределенном выполнении алгоритмов планирования заданий и выделении им ресурсов;

— учитывать специфику управления разнородными объектами, объединяя под управлением одного менеджера сходные объекты (ВС, домены). Объединение ВС в домен может осуществляться по различным

критериям общности: архитектуры, аппаратно-программной платформы, политике администрирования, территориального расположения, принадлежности одной организации и т.п.;

— контролировать менеджеры СУ, входящие в один домен, одной организацией, обеспечивающей их сопровождение, и использовать общие для данного домена алгоритмы планирования заданий на вычислительные ресурсы;

— сократить количество и разнообразие типов объектов управления, приходящихся на один субъект управления (менеджер), что упрощает процесс выработки и реализации управленческого решения и уменьшает неопределенность сложной мультипрограммной ситуации, определяя и фиксируя ряд параметров для вышестоящего уровня.

3. Алгоритмы распределенного планирования. Будем полагать, что задание может быть выполнено на любой ВС из состава грид-среды и что при выработке управляющего решения в локальном планировщике менеджера СУ (в дальнейшем, при выработке управляющего решения в менеджере) используются два основных параметра задания: необходимое для счета количество вычислительных модулей и запрашиваемое время выполнения задания. Реально используется большее количество параметров [10], но для понимания сути достаточно этих двух.

В предлагаемых алгоритмах выработки управляющего решения в локальном планировщике для описания вычислительных ресурсов используют следующие характеристики заданий и СПО ВС:

— площадь пользовательского задания, под которой понимается произведение запрашиваемого заданием количества ВМ на запрашиваемое время выполнения задания;

— суммарная площадь заданий на конкретной ВС, которая равна сумме площадей заданий, находящихся в очереди СПО этой ВС и выполняющихся на ее вычислительных ресурсах;

— загруженность, под которой понимается отношение суммарной площади заданий ВС к общему числу вычислительных модулей, выделяемых для выполнения пользовательских программ. Эта характеристика показывает, сколько времени в среднем будет еще занят вычислениями каждый из ВМ кластерной ВС;

— граница загруженности ВС, которая ограничивает сверху загруженность ВС.

Разница между границей и значением текущей загруженности ВС характеризует суммарную площадь заданий, которые можно поставить в очередь СПО этой ВС.

Менеджеры М1 ставят пользовательские задания в очередь СПО, не допуская превышения границы загруженности (рис. 2). Необходимо отметить, что в некоторых случаях СПО накладывает ограничение на максимальное время счета пользовательских заданий. Поэтому площадь задания, которое можно поставить в очередь СПО, ограничивается либо разницей между границей и значением текущей загруженности ВС, либо значением, равным произведению запрашиваемого заданием числа ВМ на максимальное допустимое время счета для данной СПО. Локальный планировщик учитывает эту особенность и не может планировать в очередь СПО пользовательское задание, площадь которого будет больше этого ограничения. Изменяя значения границы загруженности, можно изменять перераспределение заданий между очередями СПО и очередями менеджеров СУ.

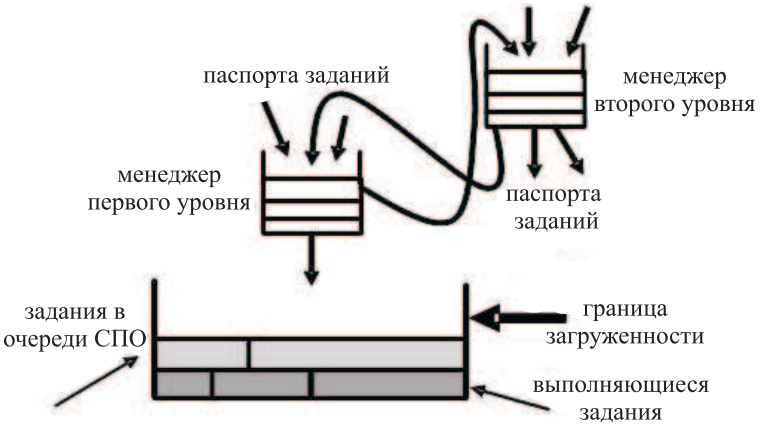

Рис. 2. Распределение заданий по очередям СПО и менеджеров

Как показано в [11], при соответствующем выборе целевой функции, например отклонения значения нагрузки вычислительного модуля от среднего значения нагрузки вычислительных модулей по локальной окрестности, возможна минимизация этой целевой функции посредством выработки локальных управляющих решений по балансировке нагрузки ВМ.

В основу выбранной в грид-среде стратегии планирования положен принцип выработки субоптимального решения в ходе согласованного взаимодействия менеджеров СУ. Распределенные по сетевой распределенной вычислительной среде менеджеры СУ принимают локальные управляющие решения, которые являются частью вырабатываемого СУ глобального решения. Решение о выделении вычисли-

тельных ресурсов для пользовательского задания принимается только менеджером СУ, управляющим этим ресурсом, так как он обладает наиболее полной и точной информацией о выделяемом ресурсе, что позволяет принять решение на основе актуальных данных.

Поясним, используя рис. 3, организацию функционирования системы менеджеров СУ грид-среды. Итак, на рис. 3 представлена грид-среда, состоящая из 7 ВС, каждая из которых имеет запущенный на ее УМ менеджер $M1_i$, $i = 1, \ldots, 7$. Каждый менеджер $M1_i$, $i = 1, \ldots, 7$, хранит в своей таблице ресурсов строку, в которой записано полученное от СПО соответствующей $ВС_i$ значение суммарной площади заданий, которые могут быть приняты СПО до превышения ее границы загруженности. На рис. 3 — это указанные в выносках 3, 5, 7, 4, 6, 6, 1 для менеджеров $M1_1$, $M1_2$, …, $M1_7$, соответственно.

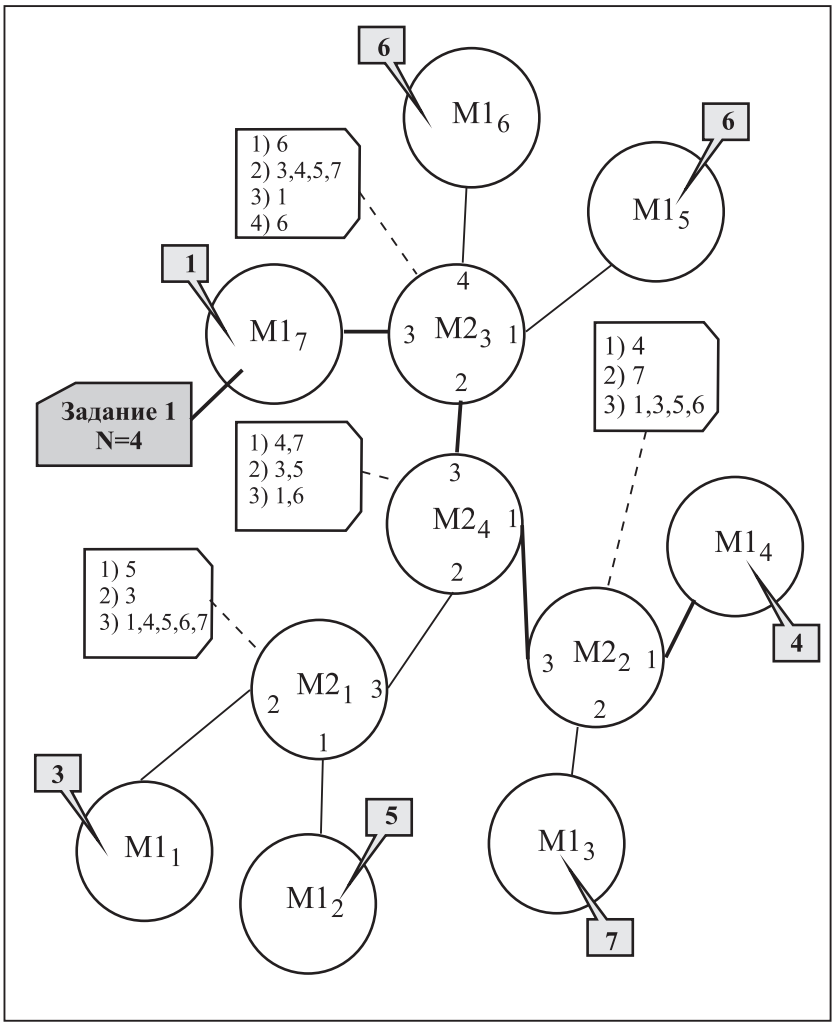

Рис. 3. Пример структуры СУ грид-среды, состоящей из 7 ВС

Менеджеры М2 имеют множество пронумерованных портов логических каналов, связывающих их с менеджерами М1 и М2. Для каждого порта менеджер М2 содержит строку таблицы ресурсов, содержащую значения площадей заданий, которые могут быть приняты СПО ВС, менеджеры М1 которых достижимы по ациклическому графу логических каналов из рассматриваемого менеджера М2. Для сокращения объема таблиц совпадающие значения площадей разных ВС в таблице представляются одним значением. Так, например, менеджер $M2_4$ имеет в строке таблицы ресурсов порта 1 сведения о $ВС_3$ и $ВС_4$, в строке порта 2 — сведения о $ВС_1$ и $ВС_2$, а в строке порта 3 — сведения о $ВС_5$, $ВС_6$, $ВС_7$.

Завершение заданий в ВС, выход из строя отдельных ВС, каналов связи, а также восстановление ВС и каналов и подключение новых ВС к грид-среде вызывают соответствующее изменение таблиц ресурсов менеджеров. В целом, каждый менеджер М2 имеет полную информацию о ресурсах грид-среды, но в разных менеджерах она представлена своей таблицей ресурсов.

Очередь заданий менеджера СУ функционирует по дисциплине FIFO. Каждый раз при изменении таблицы ресурсов или истечении заданного интервала времени от завершения предыдущей менеджер СУ производит попытку планирования заданий из своей очереди. Сначала анализируется список ВС в таблице ресурсов. В соответствии с применяемым алгоритмом планирования выбирается ВС с достаточным для выполнения задания количеством вычислительных узлов. Если подходящих ВС из таблицы ресурсов менеджера СУ найдено не было, поиск осуществляется по записям, соответствующим смежным менеджерам СУ.

В случае наличия требуемых ресурсов в системе задание передается соответствующему смежному менеджеру СУ. При отсутствии ресурсов задание ставится в конец очереди менеджера СУ, и менеджер распределяет следующее задание из очереди. Совокупность очередей всех менеджеров СУ образует глобальную единую глобальную очередь к совокупным ресурсам грид-среды.

Очевидно, что при передаче задания между менеджерами возможно его бесконечное пребывание

в очередях менеджеров. Для предотвращения этого задание имеет метку, значение которой разрешает или запрещает перепланирование задания. Если перепланирование запрещено, то локальный планировщик менеджера пересылает задание менеджеру М1 ВС, ресурсы которой запланированы для исполнения задания. Конечно, эта пересылка возможна при наличии места в очереди этого менеджера М1, иначе задание ждет возможности такой пересылки. Установка метки запрещения перепланирования задания может быть обусловлена либо превышением количества перепланирований, либо истечением временно́го интервала пребывания задания в грид-среде.

Эта же метка используется при планировании выполнения параллельного задания на ВМ разных ВС. Части одного задания, назначенные на разные ВС, представляются как отдельные задания с меткой, запрещающей их перепланирование, что гарантирует поступление этих заданий в назначенные ВС.

В ряде исследований, например [12], применительно к планированию заданий в древовидной структуре очередей применяются алгоритмы, выравнивающие количество заданий в очередях. Однако представляется, что более подходящими будут алгоритмы, учитывающие загруженность ресурсов грид-среды. В настоящей статье исследуются эвристические алгоритмы, базирующиеся на принципах минимальной загруженности и минимальной достаточности [3].

В соответствии с принципом минимальной загруженности задание направляется на ВС, имеющую максимальную разницу между значениями границы загруженности и загруженности ВС, то есть пользовательское задание распределяется на менее "загруженную" ВС, либо, при одинаковой загруженности, на ВС с минимально необходимым числом единиц свободного ресурса. Данный алгоритм обеспечивает динамическое выравнивание вычислительной нагрузки на ВС, поэтому будем называть его балансирующим алгоритмом.

Альтернативным вариантом является использование принципа минимальной достаточности: задание планируется на ВС, имеющую достаточное для немедленного начала выполнения задания число единиц свободного ресурса. Было проведено исследование 2-х модификаций алгоритма планирования, использующего принцип минимальной достаточности. В первой модификации алгоритма задания поступали на ресурсы грид-среды по очереди, во второй — задания из очереди менеджера СУ поступали на счет по мере возможности, без учета очередности: как только на кластерной ВС освобождалось определенное число вычислительных модулей, из очереди менеджера извлекалось первое задание, способное выполняться на освободившихся ресурсах. Это позволяет более полно загружать ВС, но при этом нарушается очередность поступления заданий. Для ряда применений очередность выполнения не является важным параметром, однако важна скорость обработки потока заявок.

Поясним функционирование системы менеджеров СУ с помощью примеров выполнения алгоритма планирования параллельных заданий на вычислительные ресурсы грид-среды, приведенной на рис. 3. В данных примерах реализуется стратегия назначения задания на ВС с минимально достаточным количеством ресурсов.

Пример 1. Задание, запускаемое на $ВС_7$, которому необходимо 4 единицы ресурса, будет передано от $М1_7$, к $М2_3$, а затем, согласно таблице ресурсов этого менеджера, задание будет передано $М2_4$, а затем к $М2_2$ и $М1_4$. Менеджер $М1_4$ спланирует задание на подконтрольную ему $ВС_4$. Назначение задания на $ВС_4$ приведет к изменению таблиц ресурсов менеджеров $М1_7$, $М2_3$, $М2_4$, $М2_2$ и $М1_4$ (будет исключено значение 4 из всех таблиц ресурсов). В связи с тем, что менеджеры уровня 2 имеют агрегированную информацию, решение о выделении ресурсов может приниматься только менеджером уровня 1 или смежным с ним менеджером уровня 2.

Пример 2. Задание, требующее 8 единиц ресурсов, запущенное через любой М1, будет приостановлено в инцидентном ему М2 до появления необходимого количества свободных ВУ.

В таблицах ресурсов менеджеров могут быть сформированы также прогнозные значения свободных ресурсов ВС грид-среды для заданной шкалы моментов времени в будущем. Менеджеры могут принимать решения на основе этих прогнозных значений.

4. Исследование эффективности алгоритмов распределенного планирования. Была проведена серия экспериментов: однократно сформированный тестовый поток заданий поступал на вход грид-среды, состоящей из двух ВС, next и neo, и функционирующей под управлением созданной СУ грид-среды [10]. Характеристики кластерных ВС представлены в табл. 1.

Тестовый поток состоит из двух частей по 25 заданий, представляющих собой типовой набор заданий для МВС-1000, сформированный в ИПМ им. Келдыша РАН. В отдельной серии экспериментов сформированный тестовый поток заданий запускался независимо на каждой из ВС, входящей в состав грид-среды. В дальнейших экспериментах оба эти потока одновременно поступали на вход системы управления грид-средой. В табл. 2 приведены результаты всей серии экспериментов.

Таблица 1

Название ВС	Количество процессоров	Архитектура ЦП	Сеть
next	16	Xeon	Myrinet
neo	9	Itanium	Myrinet

Таблица 2

	ВС	Количество выполненных заданий	Время счета ч:мин:сек
раздельно	next neo	25 25	02:43:54 05:03:58
A1 граница = ∞	next neo	42 8	03:47:35 03:36:18
A2 граница = 80	next neo	41 9	03:21:11 03:18:29
A3 граница = 70	next neo	26 24	03:31:46 03:04:30
A4 граница = 60	next neo	32 18	04:58:01 02:32:39
A5 граница = 50	next neo	29 21	03:14:21 04:23:49
B1 в порядке очереди	next neo	33 17	03:38:32 03:54:27
B2 без очередности, при первой возможности	next neo	28 22	02:56:17 03:06:31

В экспериментах A1–A5 используется балансирующий алгоритм с различными значениями параметра “граница загруженности” ВС.

В экспериментах B1 и B2 используется алгоритм планирования, основанный на принципе минимальной достаточности. В варианте B1 задания поступали на ресурсы грид-среды по очереди, в B2 задания поступали на счет по мере возможности, без учета очередности.

В серии экспериментов A1 с использованием в СУ грид-среды балансирующего алгоритма с границей загруженности, равной бесконечности, поток заданий распределялся по вычислительным ресурсам грид-среды, не задерживаясь в очередях менеджеров СУ. Необходимо отметить, что если на две одинаковых ВС распределить поток заданий таким образом, чтобы обеспечить в начальный момент одинаковое значение загруженности, то из-за различия времени реального выполнения заданий время выполнения частей потока на каждой из ВС будет различаться. На одной ВС-next часть потока заданий посчиталась быстрее, и эта ВС простаивала, пока другая ВС досчитывала свои задания, что и наблюдается в результатах эксперимента с границей, равной бесконечности. Однако нужно отметить, что за счет более рационального распределения заданий по ВС (на ВС с бóльшим числом ресурсов поступило бóльшая часть заданий потока) удалось уменьшить максимальное из времен обработки заданий по сравнению с выполнением тех же потоков заданий на двух независимых ВС.

Из табл. 2 видна тенденция ухудшения качества планирования потока заданий при значениях загруженности меньших и бóльших 80. Это связано с тем, что при больших значениях границы загруженности результаты экспериментов стремятся к результатам эксперимента с границей загруженности равной бесконечности. При меньших значениях границы загруженности ВС бывают ситуации, когда задания с малым количеством запрошенных процессоров и большим запрошенным временем счета вносят значительный вклад в загруженность ВС, тем самым превышая границу загруженности, в то время как в ВС остаются свободные процессоры. В таких ситуациях время простоя процессоров ВС оказывается значительным, а значит, и суммарное время счета тоже возрастает.

Результаты эксперимента, в котором использовалась модификация алгоритма, учитывающая очередность поступления задания, приведены в таблице в строке "B1 в порядке очереди". Видно, что использование такого алгоритма не дает выигрыша по сравнению с лучшими результатами экспериментов с использованием балансирующего алгоритма. В результате экспериментов B2 максимальное из времен обработки частей потока оказалось чуть меньше наименьшего из времен, полученных в экспериментах с балансирующими алгоритмами. В данном эксперименте был получен наилучший показатель эффективности обработки потока заданий на ресурсах грид-среды, однако применимость такого алгоритма ограничена.

Необходимо отметить также, что на протяжении всех экспериментов СУ [10] демонстрировала устойчивую работу и в штатных режимах, и при большой нагрузке: все поступившие в систему задания были спланированы на вычислительные ресурсы грид-среды и успешно выполнены.

В результате выполнения серии экспериментов были сделаны следующие выводы:

— использование балансирующих алгоритмов A1–A5 позволяет равномерно загружать вычислительные ресурсы грид-среды, что подтверждается практически одинаковым временем счета каждой из частей потока заданий на ВС;

— за счет изменения параметров алгоритмов, а также изменения самой стратегии выделения ресурсов пользовательским заданиям можно оказывать значительное влияние на качество планирования пользовательских заданий на ресурсы грид-среды, что подтверждается существенной разницей суммарного времени выполнения потока заданий в различных экспериментах;

— объединение разнородных ВС в грид-среду под управлением СУ позволяет за меньшее время обрабатывать поток пользовательских заданий по сравнению с использованием ресурсов независимо функционирующих ВС.

СПИСОК ЛИТЕРАТУРЫ

1. *Foster I., Kesselman C., Tsudik G., Tuecke S.* A security architecture for computational grids // Proc. 5th ACM Conference on Computer and Communications Security. San Francisco: ACM Press, 1998. 83–92.
2. *Корнеев В.В.* Вычислительные системы. М.: Гелиос АРВ, 2004.
3. *Корнеев В.В, Киселев А.В., Семенов Д.В., Сахаров И.Е.* Управление метакомпьютерными системами // Открытые системы. 2005. № 2. 11–16.
4. http://www.jscc.ru/informat/MVS15kPrgGuide.zip.
5. http://www.parallel.ru/cluster/batch-system.html.
6. *Столингс В.* Операционные системы. 4-е изд. М.: Издат. дом "Вильямс", 2002.
7. *Богданов С.А., Коваленко В.Н., Хухлаев Е.В., Шорин О.Н.* Метадиспетчер: реализация средствами метакомпьютерной системы Globus. Препринт ИПМ № 30. Москва, 2001.
8. GridWay Metascheduler: Metascheduling Technologies for the Grid. URL: http://gridway.org.
9. *Савин Г.И., Корнеев В.В., Шабанов Б.М., Телегин П.Н., Семенов Д.В., Киселев А.В., Кузнецов А.В., Вдовикин О.И., Аладышев О.С., Овсянников А.П.* Создание распределенной инфраструктуры для суперкомпьютерных приложений // Программные продукты и системы. 2008. № 2. 2–7.
10. Руководство программиста грид (http://www.jscc.ru/informat/grid1.zip).
11. *Корнеев В.В.* Архитектура вычислительных систем с программируемой структурой. Новосибирск: Наука, 1985 (http://andrei.klimov.net/reading/1985.Korneev.- .Arkhitektura.vychislitel'nykh.sistem.s.programmiruemoi.strukturoi.zip).
12. *Houle M., Symvonis A., Wood D.* Dimension-exchange algorithms for token distribution on tree-connected architectures // J. of Parallel and Distributed Computing. 2004. № 64. 591–605.

Поступила в редакцию
24.05.2010