РОССИЙСКАЯ АКАДЕМИЯ НАУК
ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР

СООБЩЕНИЯ ПО ПРИКЛАДНОЙ МАТЕМАТИКЕ

М.Г. КОНОВАЛОВ, Ю.Е. МАЛАШЕНКО,
И.А. НАЗАРОВА

МОДЕЛИ И МЕТОДЫ УПРАВЛЕНИЯ ЗАДАНИЯМИ В СИСТЕМАХ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ РЕСУРСОВ

ВЫЧИСЛИТЕЛЬНЫЙ ЦЕНТР РАН
МОСКВА 2009

УДК 519.87

Ответственный редактор
член-корр. РАН, доктор физ.-матем. наук
Ю.Н.Павловский

Рассмотрена проблема управления процессом коллективного использования ресурсов для систем распределенных вычислений. Для решения проблемы предложена динамическая имитационная модель. Модель опирается на подробное алгоритмическое описание процесса коллективного взаимодействия удаленных ресурсов и пользователей через телекоммуникационную сеть. Проблема сводится к задаче оптимального управления марковскими последовательностями. Для ее решения используются оригинальные адаптивные алгоритмы, свойства которых исследованы с помощью имитационной компьютерной программы. В работе подробно описаны результаты численных экспериментов, проиллюстрированные графиками и таблицами. Дается обширный обзор литературы на эту тему.

Работа поддержана грантом РФФИ N 09-07-12032-офи_м.

Ключевые слова: системы грид, распределенные вычисления, математическое моделирование, адаптивное управление.

Публикуется в авторской редакции.

Рецензенты: Н.М. Новикова,
Я.М. Агаларов

Научное издание

Введение

В последние годы существенно возрос интерес к системам распределенных вычислений, таким как грид [1]. Термин "грид" будем далее использовать преимущественно для удобства и понимать в самом широком смысле.

Системы грид предполагают совместное использование ресурсов, включая процессоры, программное обеспечение, базы данных и пр.; конкурирующими или сотрудничающими потребителями, а так же группами потребителей. При этом ресурсы варьируются от настольных или переносных персональных компьютеров до мощных кластеров, содержащих большое число процессоров. С помощью грида идея кластерных вычислений распространилась на глобальные сети.

Для всех без исключения гридов существует общая научно-техническая проблема, которая заключается в отыскании способов эффективного распределения ресурсов между потребителями. Собственно сама идея грида возникла под влиянием того факта, что в мире уже существует и продолжает наращиваться огромный парк слабо используемых, недостаточно загруженных компьютерных вычислительных мощностей, и естественно возникает мысль повысить их отдачу путем объединения в систему коллективного доступа.

В то время как проблема управления ресурсами содержит многие аспекты, включая необходимость программно-аппаратных решений, данная работа в основном касается общих теоретических вопросов, связанных с логической схемой организации менеджмента в гриде и возможностью применения математических методов для разработки алгоритмов распределения потоков заданий.

Несколько лет назад уже высказывалось соображение о том, что практическая проблема управления процессом коллективного использования ресурсов является одновременно одной из

ключевых для систем распределенных вычислений и привлекательной в смысле использования арсенала средств теоретической информатики и математики [2-4]. Указанная проблема начала довольно интенсивно разрабатываться одновременно с началом создания первых гридов и продолжает занимать одно из центральных мест в исследованиях по теории распределенных вычислений.

В разделе 1 настоящей работы приводится не претендующий на полноту обзор литературы по вопросу управления процессом коллективного использования вычислительных ресурсов, в который помещены результаты, наиболее интересные с точки зрения авторов.

В разделе 2 предлагается новый подход к оперативному управлению размещением заданий в системе распределенных вычислительных ресурсов. Предложена оригинальная динамическая имитационная модель, которая опирается на подробное алгоритмическое описание процесса коллективного взаимодействия удаленных ресурсов и пользователей через телекоммуникационную сеть.

Управление потоками заданий включает в себя назначение ресурсов для их выполнения и выбор объема текущих пакетов заданий. Цели управления формализуются в терминах производительности системы, количества отказов в ней, а также стоимостных показателей. Проблема сводится к задаче оптимального управления марковскими последовательностями. Для ее решения используются адаптивные алгоритмы, оптимизирующие предельные средние характеристики. Предложенные адаптивные стратегии слабо зависят от структуры и особенностей конкретной модели. Поэтому в случае изменения деталей, например, в сторону большего отражения особенностей конкретной системы, работоспособность алгоритмов сохранится.

Имитационная модель реализована в виде программной си-

стемы, что дает возможность экспериментально анализировать свойства исследуемой системы, и, в том числе, свойства предлагаемых алгоритмов адаптивного управления процессом размещения заданий на распределенных вычислительных ресурсах. Результаты численных экспериментов с подробным описанием, иллюстрациями и таблицами можно найти в конце работы.

1. Проблема размещения заданий на распределенных вычислительных ресурсах. Основные подходы и методы

§1. Общее описание проблемы

Грид представляет собой коллекцию идей, методов и технологий, которые способствуют виртуальным организациям многих пользователей с их разнообразными приложениями использовать вычислительные ресурсы разных типов, распределенные на большом географическом пространстве [5].

Характерные признаки грида состоят, в основных чертах, в следующем [6].

Грид порожден составными частями, относящимися к различным административным областям. Это характерное свойство отличает его от кластера, и оно же выдвигает специфическую задачу согласованного управления гридом.

Грид содержит разнообразные типы вычислительных ресурсов от персональных компьютеров до суперкомпьютеров, а также специфические ресурсы, например, телескопы.

Грид обладает высокой степенью масштабируемости и может разрастаться от нескольких штук до нескольких миллионов ресурсных единиц.

Грид должен обладать высокой степенью приспособляемости к отказам отдельных узлов, вероятность которых возрастает с ростом масштаба системы. В этих условиях поведение приложений и брокеров ресурсов должно адаптивно приспосабливаться для обеспечения качества сервиса.

Наиболее важной разновидностью гридов является вычислительный грид. Вычислительный грид представляет собой архитектуру, которая, используя большое число гетерогенных компьютеров, позволяет решать объемные вычислительные задачи. При этом планирование распределения заданий (job scheduling) является важной составной частью высокопроизводи-

тельной вычислительной среды грид.

Вычислительный грид является слабо связной распределенной системой, которая использует объединенные через сеть вычислительные ресурсы для решения крупномасштабных вычислительных систем. В отличие от традиционных кластерных систем грид может осуществлять разделение и совместное (коллективное) использование как вычислительных ресурсов, так и ресурсов памяти. Это является преимуществом грида по сравнению с традиционными кластерными системами [5, 7-9].

Главные составные части грида — это потребители, брокеры ресурсов, информационный сервис (Information Service), провайдеры ресурсов и собственно ресурсы. Каждый потребитель может пользоваться услугами по крайней мере одного брокера ресурсов. Задача брокера — найти и предоставить клиенту ресурс, необходимый для выполнения его приложения. Задача ресурса заключается в предоставлении необходимой информации о его структуре, состоянии и мощности. Брокер ресурсов взаимодействует с информационным сервисом, в котором накапливается полная информация обо всех ресурсах, доступных в гриде. Найденный брокером ресурс может оказаться недостаточным для выполнения конкретного задания, поэтому процесс взаимодействия между потребителем, брокером ресурсов и информационным сервисом продолжается, вообще говоря, в течение всего времени работы приложения и сопровождается привлечением новых ресурсов. Для обнаружения и локализации данных брокер ресурсов пользуется так называемым каталогом копий (Replica Catalog). Фактически эта служба устанавливает соответствие между логическими именами файлов данных и адресами физического расположения их копий.

Различаются три категории гридов [10].

Простейший тип — это кластерные гриды (Cluster Grids), состоящие из одной или более подсистем и обеспечивающие

единую точку доступа для пользователей, работающих над одним проектом или в одной организации.

Гриды университетского типа (Campus Grids) позволяют разделять ресурсы нескольким проектам или нескольким организациям внутри одного проекта или одной организации.

Глобальные гриды (Global Grids) представляют собой объединение гридов университетского типа, которое пересекает границы организаций и образует большую виртуальную систему.

Для того чтобы в такой динамичной и гетерогенной среде как грид добиваться необходимого качества обслуживания, нужны хорошие алгоритмы распределения ресурсов.

Задача распределения ресурсов и теория расписаний являются достаточно популярными среди исследователей вычислительных гридов [11-21]. Достаточно полный обзор результатов по теории расписаний для систем в реальном времени (Real-Time System), можно найти, например, в [22]. Стратегии размещения задач в гриде разрабатывались в [23, 24]. В [25] проблема формулировалась и рассматривалась как задача о рюкзаке. С практической точки зрения можно отметить работу [26].

Задача размещения заданий для единственного компьютера, позволяющего параллельные вычисления, существенно отличается от планирования заданий в гриде. Для последнего составление расписаний становится более сложным, поскольку в рассмотрение оказываются вовлеченными компьютеры различных размеров и типов, к тому же обладающее своими собственными стратегиями распределения заданий [27]. В работах [28-31] эту проблему пытаются разрешить с помощью многокритериального подхода к составлению расписаний.

Многие алгоритмы, составляющие расписание, используют при определении нагрузки в системе пороговый метод [32, 33]. Однако этот подход не вполне адекватен в гетерогенной среде грид, поскольку здесь нагрузка существенно неоднородна,

а вычислительная мощность каждого узла различна. Следовательно, удачное распределение заданий в среде грид важно, прежде всего, потому, что это позволяет более эффективно использовать ресурсы и повысить в целом производительность системы [9, 32-39]. В работе [40] подробно изучено, как влияет на производительность жесткой системы (Hard Real-Time Multiprocessor System) перемещение задач от одного ресурса к другому (с целью их распараллеливания) или прерывание выполнения заданий. Исследовались системы, в которых происходил рост числа процессоров, увеличивалось число заданий, или у поступающих заданий уменьшалось время до дедлайна. Численный эксперимент показал, что в двух последних случаях стратегия прерывания заданий и распараллеливания вычислений оказалась более эффективной, чем в первом случае.

Подходы к распределению ресурсов в гриде также существенно отличаются от распределения ресурсов единичного компьютера. Планировщики заданий рассматривают грид как большой пул ресурсов, к которым они имеют исключительный доступ. Поэтому они осуществляют планирование независимо, не принимая во внимание нагрузку, приоритеты, сценарии расписания, порождаемые другими. Отсутствие координирующего механизма приводит к возникновению узких мест, равно как и к недоиспользованию части ресурсов.

Ресурсы грида (кластеры, суперкомпьютеры) управляются локальными системами (Local Resource Management Systems, LRMS). Эти ресурсы могут быть слабо связаны в грид университетского типа путем использования мультикластерных систем [10], позволяющих совместное использование кластеров, принадлежащих одной и той же организации. Однако такое устройство пула ресурсов ограничивает получение к нему доступа извне, включение его во внешнюю организацию, или участие в кооперативной федерации автономных кластеров.

Конечные потребители (или их локальные планировщики) посылают задания в LRMS, не имея сведений о времени отклика или сервисных возможностях. Иногда эти задания находятся в очереди часами, что ведет к потере качества обслуживания. Для того чтобы минимизировать время обслуживания, необходимо использовать стратегии планирования, которые учитывают приоритеты заданий, их важность в совокупности с информацией о расположении и состоянии ресурсов. Для эффективного планирования необходимо знать, как потребители оценивают трудоемкость своих приложений. Нужна, таким образом, обратная связь, которая позволяла бы потребителям оценивать свои возможности.

В то же время существующие "системно-центрические"(system-centric) подходы к планированию [10] не учитывают требования потребителей по качеству обслуживания. Подобные методы планирования размещают задания в соответствии с системными параметрами, которые касаются загрузки ресурсов или пропускной способности системы. Планировщики уделяют внимание либо минимизации времени отклика (response time — сумма времени ожидания и времени фактического выполнения задания), либо максимизации общей загрузки ресурсов. В этой ситуации целью планирования оказывается улучшение положения системы в целом, но не требования, предъявляемые отдельными потребителями.

Кластеры, мультикластеры и гриды предназначены для выполнения высокопроизводительных приложений, включающих большое число задач, часто исчисляемых сотнями и даже тысячами. Во многих случаях имеется совокупность независимых задач, которые могут выполняться в произвольном порядке [41], а именно: обработка результатов экспериментов с различными исходными параметрами, обработка массивов данных (например, видео), имитационное моделирование методом

Монте-Карло и другие приложения, возникающие в астрономии, физике высоких энергий. Приложение с делимой нагрузкой (divisible load application) обладает тем свойством, что может быть разделено на произвольное количество независимых задач, которые могут выполняться параллельно.

Распределение независимых задач по гетерогенным процессорам изучалось во многих работах [41-44]. В частности, предложены эвристические стратегии календарного планирования. Эти алгоритмы были распространены на ситуацию, когда независимые задачи совместно используют общие входные файлы. В [43, 44] изучалась проблема планирования независимых задач с общими входными файлами в ситуации, когда количество задач по крайней мере на порядок превышает количество доступных процессоров.

В следующем параграфе рассмотрим характерные примеры постановок задач и подходов, которые используются в литературе, посвященной управлению потоками заданий в гриде.

§2. Основные методы планирования размещения заданий

Работы, посвященные размещению заданий в грид с помощью аналитических моделей, опирающихся на богатейший материал, а также развитый математический аппарат теории (сетей) массового обслуживания или, более широко, теории (управляемых) случайных процессов, отличаются двумя характерными чертами:

1) математической корректностью и законченностью постановки и решения задачи;

2) наличием существенных ограничений, которые не выполняются на практике.

Второе обстоятельство делает данный подход уязвимым со

стороны разработчиков систем, однако, не умаляет ценность первого фактора. Здесь важна не столько сама возможность получить математическое решение, сколько достоверность качественных выводов, пусть и сделанных в относительно узких рамках предположений. Ниже рассмотрим характерный пример указанного подхода [45].

Рассматривается система, состоящая из N серверов, каждый из которых может выполнять любой из M типов заданий. Каждый тип заданий имеет свою независимую неограниченную очередь. Поток заданий i-го типа представляет собой прерывающийся пуассоновский процесс с интенсивностями фазы "on"и "off" соответственно $1/\xi_i$ и $1/\eta_i$ и интенсивностью входного потока в фазе "on", равной $\lambda_i,\ i = 1, \ldots, M$. Время выполнения заданий i-го типа в среднем равно $1/\mu_i$.

Любой из серверов, обслуживающих очередь заданий i-го типа, может быть в любой момент переключен на обслуживание заданий j-го типа. При этом сервер оказывается выключенным из работы на время, имеющее экспоненциальное распределение с параметром $1/\zeta_{ij}$. Для заданий i-го типа обозначим:

j_i — размер очереди,

l_i — индикатор состояния прерывающегося пуассоновского процесса,

k_i — количество серверов, занятых обслуживанием заданий этого типа,

m_{ij} — количество серверов, находящихся в стадии переключения с обслуживания потока i на поток j.

Используя эти обозначения, состояние системы можно описать как $S = (j, l, k, m)$, где j, l, k — векторы размерности M, а m — матрица размерности $M \times M$. Если никаких действий не предпринимается, то ситуация описывается марковским процессом с интенсивностью перехода из состояния S в состояние

S', заданной соотношениями

$$r(S,S') = \begin{cases} l_i\lambda_i, & \text{если } j' = j + e_i, \\ min(k_i, j_i)\mu_i, & \text{если } j' = j - e_i, \\ m_{ij}\zeta_{ij}, & \text{если } k' = k + e_i, \text{ и } m'_{ij} = m_{ij} - 1, \\ l_i\xi_i, & \text{если } l'_i = 0, \\ |1 - l_i|\eta_i, & \text{если } l'_i = 1. \end{cases}$$

где e_i — вектор, у которого i-я компонента равна 1, а остальные компоненты равны 0. Предположим, что в каждом состоянии можно осуществить некоторое действие a. Пусть допустимое действие заключается в выборе пары (i, j) и переключении нескольких, скажем k, серверов от обслуживания потока i к обслуживанию потока j. Тогда под влиянием выбранного действия параметры системы изменятся, причем

$$k_i^a - k_i - k, m_{ij}^a = m_{ij} + k, k = 1, ..., k_i.$$

Дальнейшее развитие модели заключается в дискретизации времени и переходу к классическому варианту марковского процесса принятия решений. Отметим, что раздел 2 данной работы целиком посвящен изложению оригинальной модели, разработанной и исследованной в русле адаптивного варианта теории марковского процесса принятия решений.

Обратимся к следующему классу моделей — статическим оптимизационным моделям.

Классическая теория оптимизации является надежным инструментом во многих областях и проблема распределения вычислительных ресурсов не является исключением. Недостатки этого подхода не связаны исключительно с гридом, но проявляются в данной проблематике особенно отчетливо. Первый недостаток в том, что статическое описание не в состоянии адекватно отразить постоянно и принципиально изменяющийся статус большой распределенной вычислительной системы. Второе затруднение связано с тем, что грид — по определению децентрализованная система, поэтому неясно, кто и где

будет решать глобальную оптимизационную задачу, даже если она удачно составлена. Тем не менее, данный подход, безусловно, заслуживает внимания и потому активно эксплуатируется. Далее приводятся два примера (А и Б) подобной постановки задачи применительно к делимой нагрузке (§1).

Пример А. Модель вычислительной среды состоит из набора географически разделенных кластеров, которые соединены через Интернет [46]. Каждый кластер имеет фронтальный (front-end) процессор, который соединен с локальным маршрутизатором через локальный канал связи с глобальной сетью. Маршрутизаторы связаны друг с другом через каналы Интернета. Таким образом, вся система моделируется графом, узлы которого — фронтальные маршрутизаторы, а ребра — линии связи магистральной сети. Предполагается, что каждое ребро характеризуется максимальным количеством соединений, которое может быть выделено на новое приложение, а также шириной полосы пропускания, доступной для одного соединения.

Предполагается, что вся вычислительная мощность кластера C_k, который на самом деле состоит из многих процессоров, сосредоточена в одном фронтальном процессоре C^k_{master}, соединенном с фронтальным маршрутизатором C^k_{router} посредством линии связи, имеющей пропускную способность g_k. Производительность фронтального процессора обозначается через s_k. Известно, что при изучении делимой нагрузки такая "замена" является корректной [47, 48]. Маршруты соединений между кластерами фиксированы. Для соединения кластера C_k и C_l используется один и тот же фиксированный список линий магистральной сети, обозначаемый через L_{kl}.

Предположим, что каждому кластеру соответствует некоторое (единственное) приложение A_k, причем все входные данные, необходимые для его выполнения, находятся внутри этого кластера. По предположению приложение порождает делимую

нагрузку, тогда его выполнение можно осуществлять частями, параллельно, на разных кластерах. Каждому приложению A_k приписан коэффициент π_k, выражающий приоритет этого приложения. Введем следующие обозначения. Пусть

w_k — количество вычислений (в миллионах инструкций в секунду), необходимых для выполнения задания A_k;

δ_k — объем нагрузки (в байтах), связанной с выполнением задания A_k;

α_{kl} — доля приложения A_k, которая направляется из кластера C_k для выполнения на кластере C_l;

β_{kl} — количество сетевых соединений, которые открываются при пересылке части задания из кластера C_k в C_l.

В этих обозначениях выпишем оптимизационную задачу: найти

$$\max \min_k \left(\frac{\alpha_k}{\pi_k} \right)$$

при ограничениях

$$\begin{cases} \forall\, C_k: & \sum\limits_l \alpha_{kl} = \alpha_k, \\ \forall\, C_k: & \sum\limits_l \alpha_{kl} w_l \le s_k, \\ \forall\, C_k: & \sum\limits_{l \ne k} \alpha_{kl}\delta_k + \sum\limits_{j \ne k} \alpha_{jl}\delta_j \le g_k, \\ \forall\, i: & \sum\limits_{l_i \in L_{kl}} \le maxconnect(l_i), \\ \forall\, k,l: & \alpha_{kl}\delta_k \le \beta_{kl} g_{kl}, \\ \forall\, k,l: & \alpha_{kl} \ge 0, \\ \forall\, k,l: & \beta_{kl} \in N. \end{cases}$$

Пример Б. В общем случае предположим, что имеется n заданий, причем задание i состоит из m_i задач. Каждая задача в задании имеет свой набор требований по ресурсам и свое количественное значение полезности. Полезностьj-ой задачи из i-го задания обозначается как u_{ij} и вычисляется с помощью некоторой функции полезности, выбранной с учетом собственных

свойств задачи (размер, объем данных и пр.), либо на основе внешних по отношению к задаче факторов, определяемых пользователем (например, бюджет задачи). Требования по ресурсам выражаются коэффициентом a_{ijkl}, где k - номер ресурсного центра, а l - тип ресурса. Каждый ресурсный центр k характеризуется ограничениями по ресурсу типа l, заданными коэффициентами R_{kl}. Если обозначить через x_{ij} индикатор выбора j-ой задачи в i-ом задании, то получаются следующая обобщенная задача о рюкзаке: максимизировать функцию

$$f(x) = \sum_{i=1}^{n} \sum_{j=i}^{m_i} u_{ij} x_{ij}$$

при ограничениях

$$\begin{cases} \sum\limits_{i=1}^{n} \sum\limits_{j=i}^{m_i} a_{ijkl} x_{ij} \leq R_{kl}, \\ \sum\limits_{j=i}^{m_i} x_{ij} \leq 1, \\ x_{ij} \in \{0,1\}, i \in \{1,...,n\}, \\ k \in \{1,...,r\}, \\ l \in \{1,...,\rho\}. \end{cases}$$

Особое значение во всей методологии имеет выбор функции полезности. Могут быть предложены следующие функции:

кредит CV (credit-value) — позволяет пользователям оперировать с виртуальными деньгами, исходя из приоритетов отдельных задач в задании;

время выполнения ERT (estimated-response-time) — оценивает оставшееся время до выполнения задания на конкретном кластере;

близость к завершению $NTCT$ (nearness-to-completion-time) — увеличивается по мере приближения к завершению задания;

сетевые ресурсы IDT (input data transfer) — учитывает затраты на передачу данных по сети между потребителем и вычислительным комплексом.

Для j-ой задачи из i-го задания перечисленные функции определяются следующим образом:

$$CV = v_{ij},$$
$$ERT = \frac{t - s_i + r_{ij}}{N_i},$$
$$NTCT = \frac{N_i}{r_{ij}},$$
$$ERT = \frac{B_i}{d_{ij}},$$

где v_{ij} означает интенсивность, с которой потребитель готов тратить деньги, t — текущий момент времени, s_i — заявленное время выполнения задания i, r_{ij} — оставшееся время выполнения задачи j в задании i, N_i — общая "длина" задания, B_k — ширина пропускной полосы к серверу k, d_{ij} — размер данных, которые необходимо передать.

Используя указанные метрики, можно вычислять функцию полезности, основываясь на всех перечисленных критериях:

$$u_{ij} = w_{CV}S(CV) + w_{ERT}S(ERT) + \\ + w_{NTCT}S(NTCT) + w_{IDT}S(IDT),$$

где S — неубывающая положительная нормирующая функция, а $w_{(.)}$ — весовые коэффициенты, отражающие относительную важность отдельных компонент.

Отметим, что экономико-стоимостные соображения, которые появились в конце последнего примера, составляют важную составляющую современных исследований в области распределения вычислительных ресурсов. Более подробно этот аспект рассматривается в §3.

В последние годы появились мультикластерные системы, содержащие от двух до нескольких десятков кластеров, в свою очередь состоящих из нескольких сотен или даже тысяч процессоров. Такие системы объединены в единое целое с помощью глобальной телекоммуникационной сети. Примерами подобных систем являются французская Grid5000 [50] или датская Distributed ASCI Supercomputer (DAS) [51]. Одной из центральных проблем в управлении такими системами является проблема организации одновременного доступа (co-allocation) к распределенным ресурсам (процессорам, памяти и т.д.) при выполнении одного задания, состоящего из нескольких задач. Принципиальная особенность co-allocation заключается в том, что задачи, составляющие задание, должны начинать выполняться одновременно. Проблема изучалась в работах [52-55], а механизм co-allocation был внедрен в упомянутой системе DAS.

Вычислительная среда состоит из C кластеров, каждый из которых содержит N процессоров одинаковой мощности [56]. Каждый кластер имеет планировщик для локальных заданий, действующий по принципу FCFS. Эти задания могут быть отвергнуты в случае необходимости. Поступающие глобальные заявки становятся в общую очередь, где ожидают распределения по кластерам. Заявки не приписаны изначально ни к какому кластеру. Потоки заявок пуассоновские с интенсивностями λ_g и λ_l соответственно для глобальных и локальных заявок.

Задание состоит из нескольких задач, которые должны начать выполняться одновременно. Количество задач в задании равномерно распределено на множестве $\{2, \dots C\}$. Локальное задание может состоять из одной задачи. Максимально допустимое время от момента поступления до момента начала выполнения (дедлайн задания) равномерно распределено на отрезке $[D_{min}, D_{max}]$. Количество процессоров, необходимых для выполнения задачи находится в промежутке I_S, где S — размер

наименьшего кластера, причем каждая задача в задании требует одно и то же количество процессоров. В одном варианте модели используется равномерное распределение на $I_S = [4; S]$. Другой вариант предполагает использование более реалистичного предположения (Realistic Synthetic Distribution, [52]): задача имеет размер i с вероятностью $Q = q_i/Q$, если i не является степенью двойки, и с вероятностью $3q_i/Q$, если i является степенью двойки, $0 < q < 1$, а Q — нормировочный множитель. Коэффициент 3 увеличивает вероятность выбора размера задачи, равного степени 2, а q^i повышает вероятность небольшого размера задачи. Время вычисления задания имеет экспоненциальное распределение с параметрами μ_g и μ_l соответственно для глобальных и локальных заявок.

Стратегия размещения глобальных заявок (Repeated Placement Policy, RPP) определяется следующим образом. Последовательные моменты T_k, в которые осуществляются попытки размещения задания, задаются соотношениями

$$T_0 = S + LD, \quad T_{k+1} = T_k + LD,$$

где S — момент поступления задания, D — дедлайн задания, L — параметр алгоритма. Число попыток ограничено: $k \leq K$. Если задание не размещено при попытке в момент T_K, то оно считается потерянным. Задания размещаются на наименее загруженном кластере.

Если величина дедлайна велика, то может быть использовано следующее дополнение к стратегии RPR (Wait-X policy): вновь поступившее глобальное задание игнорируется в течение времени $D - X$, где X — параметр алгоритма.

Приоритеты между глобальными и локальными заявками регулируются следующими правилами. Если глобальная заявка не получила свободных процессоров к моменту своего дедлайна, то локальные задания принудительно удаляются. В аналогичных случаях глобальное задание считается потерянным.

Показатели качества:

— интенсивность успешного обслуживания глобальных заданий (global job success rate) — процент глобальных заявок, выполнение которых началось до наступления дедлайна;

— интенсивность отказов для локальных заданий (local job kill rate) — процент локальных заданий, получивших принудительный отказ;

— общая загрузка (total load) — средний процент занятых процессоров;

— загрузка глобальными заданиями (global load) — процент общей вычислительной мощности, использованной для глобальных заявок;

— потерянное в результате простоя процессорное время (processor wasted time) — процент общей вычислительной мощности, потерянной в результате резервирования процессоров (claiming processors) для глобальных заданий до момента дедлайна.

§3. Экономический подход к управлению ресурсами

Традиционный подход к управлению ресурсами обусловлен критериями, относящимися ко всей системе, такими, как, например, использование инфраструктуры или пропускная способность. При этом стратегия распределения заданий представляет собой отображение множества заданий в множество ресурсов грид. В то же время потребности грид требуют развития моделей, которые в большей степени отражали бы интересы пользователей. Иными словами, требуется создание таких алгоритмов распределения заданий, которые обеспечивали бы максимум полезности для индивидуального потребителя. Однако в силу принципиально эгоистического поведения потребителей, невозможно заставить их объективно учитывать

общесистемные критерии иначе, как посредством организации побудительных, стимулирующих механизмов. Одним из наиболее перспективных механизмов организации грида является рыночный экономический механизм, трактующий наличие свободных вычислительных ресурсов как предложение товара, а потребителей — как источник спроса на вычислительные услуги [57-62]. Хотя первоначально мотивация создателей грида заключалась в поддержке научных исследований, коммерческая состоятельность проекта возможна только после того, как участники, то есть собственники ресурсов и их потребители, начнут платить адекватно. При этом главная цель потребителей состоит в том, чтобы максимальное количество работ было выполнено с необходимым качеством, в то время как задача собственников ресурсов — получить прибыль.

Динамика процессов, происходящих в гриде, очень сложна для моделирования, поэтому использование наработок экономической теории может оказаться полезным для понимания проблемы распределения вычислительных ресурсов.

Одним из наиболее важных для исследования вопросов в проблеме внедрения экономических принципов в организацию использования ресурсов в системах грид является вопрос организации рынка. В настоящее время в экономической литературе представлено несколько форм такой организации, однако, не вполне ясно, какая из них является наиболее подходящей и экономически целесообразной. Например, с точки зрения модели полезности наиболее привлекательным способом организации распределения ресурсов представляются комбинированные аукционы [63], в которых участник может предложить единую цену за комбинацию товаров или услуг. Применительно к системе грид это дает возможность потребителю более точно определить свою оценку набора специфических ресурсов, необходимых для его приложений. Однако этот подход явля-

ется сложным с вычислительной точки зрения, так как для определения множества победителей в таких аукционах, вообще говоря, необходимо решать NP-полную задачу.

Модель рынка ресурсов должна начинаться с описания ресурсов. Полная и точная модель грида должна включать большое количество разнообразных типов ресурсов (процессорное время, различные типы памяти, пропускная полоса сети и многое другое). В то же время для экономической модели существенным моментом является способ, которым данный тип ресурса появляется в качестве товара на рынке. Точную картину системы дал бы полный учет всех возможных ресурсов в гриде, однако, такой подход неизбежно усложнил бы рыночный механизм ценообразования, взаимодействие между рынком и его участниками, а также логику поведения участников.

В широком смысле различают три основных типа рыночных механизмов.

Товарный рынок предполагает наличие различных ресурсов (товаров), таких как компьютеры, дисковая память, пропускная полоса телекоммуникационной сети, которые покупают потребители. Существуют разные схемы определения цены на товары и услуги. Установление "твердой цены" (flat fee) означает, что потребитель платит определенную сумму за определенный период независимо от качества обслуживания. Другая схема основана на "продолжительности использования" (usage duration), то есть потребитель платит только за то время, в течение которого он использовал ресурс. Еще одна схема использует так называемую "подписную цену" (subscription), которая обобщает подход, основанный на твердой цене. В этом случае потребитель платит за определенный, заранее оговоренный промежуток времени, в течение которого он может использовать ресурс. Наконец, существует подход, в котором цена изменяется динамически в зависимости от спроса и предложения

(demand and supply).

Следующий способ организации рынка ресурсов заключается в представлении заявки на подряд, или контракте (tendering, contract net). В этой модели брокер ресурсов запрашивает продавцов о предлагаемой ими цене. Покупатель рассылает спецификацию запроса (ожидаемое время выполнения, требования на ресурсы и т.д.) потенциальным провайдерам, которые также участвуют в обсуждении цены. В результате заключается наиболее подходящий для сторон контракт.

Процесс переговоров (negotiation) о купле-продаже ресурсов для заключения договора между участвующими сторонами необходимо организовывать так, чтобы он был максимально автоматизирован и прозрачен для участников, особенно если масштаб системы велик [64]. Решению указанной задачи посвящены работы [65, 66], в которых рассматривался вариант переговоров типа "один со многими" (one to many). Там же рассматривается конкурентная модель с двусторонними переговорами, которая изучалась в [67].

Модель двусторонних переговоров содержит [68]:

1) протокол переговоров (negotiation protocol);

2) функции полезности, или отношения предпочтения;

3) стратегию переговоров.

Цель, которую преследуют потребители, используя вычислительные ресурсы, как правило, состоит либо в скорейшем выполнении задания, либо в доступе к наиболее дешевому ресурсу. В то же время цель провайдеров ресурсов — в получении наибольшей прибыли и наиболее полной загрузке ресурсов. В данном случае критериями потребителей являются ожидаемое время выполнения задания и приемлемая цена за единицу процессорного времени. Для провайдеров ресурсов это соответственно ожидаемое время до начала выполнения задания и определенное значение платы за ресурс.

Предполагается, что партнеры следуют принципу доверительного ведения переговоров (good-faith bargaining principles), по которому они всячески препятствуют срыву переговорного процесса. Поэтому считается, что в ходе переговоров продавец последовательно уменьшает цену предложения ресурса, а покупатель, наоборот, увеличивает. Этот подход, использующий функцию полезности ресурсов и содержащий в своей основе принцип аренды ресурсов, изучался, помимо собственно грид-приложений [57, 69-76], также в связи с экономическими аспектами так называемых "систем с агентами" [77-79].

Наконец, существует еще так называемая аукционная модель рынка ресурсов. В этой модели провайдер ресурсов получает предложение цены (bid) от брокера ресурсов. Причем существуют два типа таких цен: открытые цены и закрытые цены. В первом случае участники знают предложения цены друг друга, во втором — нет. В течение некоторого периода времени провайдеры ресурсов собирают информацию о предложениях цен, а затем определяют окончательную цену для потребителя и передают ресурс соответствующему брокеру ресурсов. Кроме того, аукционы могут быть повышающегося типа (ascending), когда от невысокой стартовой цены переходят постепенно ко все более высокой цене, пока не остается конкурентов, или понижающегося типа (descending), когда стартуют с высокой цены и постепенно понижают ее до того уровня, когда она станет приемлемой для какого-нибудь участника.

Известно несколько типов аукционов, различающихся по перечисленным признакам.

Аукцион, заявки на который подаются в запечатанных конвертах (sealed bid auction), характерен тем, что участники не осведомлены о ценах, предложенных другими игроками. В конце аукциона конверты вскрываются, и ресурс получает участник, предложивший наибольшую цену. Этот аукцион называ-

ется также аукционом первой цены (first price).

Аукцион Викрея (Vickrey auction) отличается от предыдущего тем, что победитель, предложивший наибольшую цену, платит сумму, равную второй по величине предложенной цене.

Английский аукцион начинается со стартовой (наименьшей) цены и прекращается, когда никто из участников больше не увеличивает предложение. Победитель получает ресурс по достигнутой цене.

Датский аукцион подобен английскому, но торг начинается с наибольшей цены, которая затем снижается.

Двойной аукцион (double auction) типичен для фондовых бирж. Этот механизм предусматривает, что продавец может предлагать цену только ниже текущей, в то время как покупатель имеет право только поднимать цену. Сделка осуществляется в момент, когда цены продавца и покупателя сравняются.

Исследования в области приложения рыночных механизмов к вычислительным гридам можно классифицировать по трем направлениям. Первое направление занимается непосредственно изучением различных моделей рынка для систем грид. В этом случае тот или иной механизм (аукцион, товарный рынок и т.д.) призван стимулировать потребителей и собственников ресурсов участвовать в работе грида [80-84]. Например, в работе [84] сравнивается модель товарного рынка и аукционная модель, причем сравнение производится по таким параметрам, как устойчивость цен, эффективность выполнения приложений, эффективность использования ресурсов. По результатам сравнения модель товарного рынка оказалась результативней. В работе [81] сравниваются некоторые типы аукционов по таким показателям, как плата потребителей, доходы собственников ресурсов, структура платы, загрузка ресурсов. Делается вывод о том, что аукцион первой цены является предпочтительным с точки зрения собственников ресурсов, аукцион Ви-

крея — с точки зрения потребителей, зато двойной аукцион превосходит оба упомянутых с обеих точек зрения. Но, к сожалению, двойной аукцион требует кооперации между провайдерами ресурсов, что трудно осуществимо на практике. Различные сценарии двойного аукциона были изучены в работе [83].

Второе направление исследований занимается исследованием стратегий планирования (scheduling) для провайдеров ресурсов. Например, в работе [85] изучается стратегия, по которой задания принимает сервер. Для исследования строится контрактная модель. В момент получения спецификации нового задания ресурс должен определить, принимать задание или нет. Ресурс может послать запрос другим потребителям в поисках наиболее выгодного, однако, в процессе этого поиска для некоторых заданий может наступить дедлайн. Для управления ситуацией вводится штраф, который предупреждает провайдера от обслуживания большего количества заданий, чем он может обслужить. В работе [53] аналогичная постановка задачи изучается с помощью функции полезности. Работа [82] также использует для анализа алгоритмов планирования заданий контрактную модель, но в качестве критериев выступают коэффициент нагрузки, доход, процент выполненных или потерянных заданий, коэффициент загрузки ресурсов.

В рамках исследования стратегий планирования для провайдеров ресурсов также изучается так называемая проблема составления расписаний с прерываниями (preemptive scheduling [86]), суть которой в общих чертах такова: провайдер ресурса должен в реальном времени определить, в каком случае необходимо прервать выполнение того или иного задания и вместо него поставить на ресурс вновь поступившее задание, возможно более неотложное, более важное (доходное) или с более коротким временем до наступления дедлайна. Для решения этой задачи в [87, 88] вводится функция полезности, указывающая,

в каких случаях следует прервать выполнение задания с тем, чтобы завершить его в дальнейшем, а в каких выгоднее выполнить имеющееся задание полностью, и лишь потом перейти к новому. Показано, что использование соответствующего алгоритма позволяет значительно улучшить производительность ресурсов. В [89] схожая проблема решается исходя из параметра $\alpha(J_i) \geq 0$, определяющего, какова может быть задержка до начала выполнения задания J_i, при условии, что оно должно быть выполнено до наступления соответствующего дедлайна.

Третье направление исследований связано с попытками выбора оптимальной стратегии для потребителей. Была разработана система (Nimrod-G), способная поддерживать различные экономические модели, такие как рынок товаров (commodity market), рынок наличного товара (spot market), и т.д. В рамках этих моделей предложены стратегии выбора ресурсов, оптимизирующие время выполнения заданий и бюджет. Недостаток подхода в том, что здесь предполагается существование единственного центрального агента, который осуществляет планирование для всех пользователей, и что эти стратегии не могут быть применены для аукционной модели.

В большинстве указанных работ предполагается однородность ресурсов, то есть делается предположение, что все ресурсы имеют одинаковую производительность и одинаковую стоимость. В этом случае задача выбора ресурсов значительно упрощается, поскольку существует взаимозаменяемость ресурсов. Однако в реальной обстановке ресурсы являются гетерогенными и отличаются друг от друга архитектурой, производительностью, памятью, ценой, пропускной полосой и т.д. Выбор подходящего ресурса для каждого приложения становится многопараметрической задачей, причем отдельные параметры являются взаимозависимыми. Соответственно усложняется и задача организации и исследования аукционов.

Организация рынка ресурсов посредством аукционов представляется во многом предпочтительной. В частности:

модель с аукционами позволяет ввести схему продажи типа "один со многими" (один провайдер — много потребителей), что делает процесс установления цены однозначным;

аукционы не требуют глобальной информации о ценах и могут быть легко внедрены в систему грид;

в отличие от товарного рынка аукцион является полностью децентрализованной системой.

Рассмотрим модель, в которой каждый потребитель имеет брокера ресурсов, управляющего возникающими заданиями (по существу отождествленного с потребителем) [90]. Задача брокера ресурсов — для каждого задания найти подходящего провайдера ресурсов, который осуществил бы размещение задания. Предполагается, что для размещения заданий используется модель аукциона первой цены.

Каждый из N потребителей, участвующих в гриде, имеет одно или несколько заданий, в спецификацию которых входит "длина" (length), время жизни (deadline) и бюджет (budget). Длина задания измеряется в миллионах инструкций. Каждое задание должно быть выполнено за время, не превосходящее время жизни, включая время проведения аукциона. На выполнение каждого задания должно быть затрачено денег не больше, чем величина бюджета.

Каждый из M ресурсов представляет собой единственный процессор, который характеризуется скоростью, измеряемой в миллионах инструкций в секунду, и ценой за одну секунду его использования. Каждый ресурс проводит собственный аукцион, и потребитель, желающий использовать данный ресурс, должен в нем участвовать.

Процесс взаимодействия потребителя и ресурса выглядит следующим образом. Потребитель посылает брокеру ресурсов

спецификацию возникшего у него задания. Брокер ресурсов запрашивает информацию об имеющихся в данный момент аукционах в информационном центре грида (Grid Information Service). В соответствии с выбранной пользователем стратегией (о стратегии ниже) брокер ресурсов выбирает провайдера и предлагает цену за ресурс (bid), которая не должна быть ниже, чем "забронированная"(reserved) цена, определенная для данного ресурса. После окончания выбранного аукциона провайдер информирует о его результатах. Если аукцион выигран, то брокер ресурсов пересылает задание для выполнения на соответствующий ресурс. Иначе процесс повторяется, при условии, что время жизни задания не истекло. По окончании выполнения задания, оно возвращается потребителю.

Предположим, что потребитель определяет одну из стратегий выбора ресурсов, а брокер ресурсов руководствуется ею. Опишем кратко эти стратегии.

Случайная стратегия (Random Policy) предписывает потребителю случайный выбор ресурса (аукциона) из множества тех, что могут обеспечить выполнение задания до истечения времени его жизни и за цену, укладывающуюся в бюджет задания.

Стратегия, оптимизирующая время выполнения задания (Time Optimized Policy), предписывает выбирать тот ресурс (аукцион), который быстрее всего выполнит задание, не нарушив при этом упомянутые ограничения по времени жизни и бюджету. При этом для формирования предлагаемой на аукцион цены используется весь бюджет.

Стратегия, оптимизирующая бюджет (Budget Optimized Policy), предписывает выбирать тот ресурс (аукцион), который является наиболее дешевым, но при этом удовлетворяет заданным ограничениям по времени жизни и по бюджету. При этом предлагаемая на аукцион цена — меньше бюджета задания. Коэффициент уменьшения связан с предполагаемым качеством

обслуживания. Если ресурс высокоскоростной, то этот коэффициент невелик, а если производительность ресурса мала, то предлагаемая цена значительно меньше бюджета задания.

Обобщенная стратегия, оптимизирующая время выполнения задания (New Time Optimized Policy), действует следующим образом. Допустимые ресурсы, то есть ресурсы, удовлетворяющие требования по времени жизни и по бюджету, упорядочиваются в порядке убывания времени выполнения задания. Ресурс (аукцион) выбирается равновероятно среди первых k элементов в этом списке. При $k = 1$ эта стратегия совпадает со стратегией Time Optimized Policy, при k равном числу допустимых ресурсов — со стратегией Random Policy.

Обобщенная стратегия, оптимизирующая бюджет (New Budget Optimized Policy), определяется аналогичным образом.

В следующей модели представлен единственный тип ресурсов — процессорный [76]. Его подразделяют на три категории в зависимости от относительной производительности. С точки зрения выполнения заданий ресурсы считаются взаимозаменяемыми, хотя потребители оценивают их по-разному. Имея в виду данный тип ресурса и три его категории, будем говорить о "товарном рынке" ресурсов (Commodity Market).

На аукционном рынке (Auction Market) все ресурсы выставляются индивидуальным образом на аукцион. Это позволяет потребителю сделать более точную, чем на рынке товаров, оценку. Все ресурсы по-прежнему оцениваются с помощью одного показателя — относительной производительности.

Каждый потребитель имеет очередь заданий вычислительного характера, для выполнения которых он должен с помощью провайдера и участия в рынке получить соответствующие ресурсы. Предполагается, что доступ задания к процессору осуществляется немедленно после того, как ресурс выделен, каждое задание имеет номинальное время выполнения (кото-

рое, однако, "неизвестно" обсуждаемому ниже алгоритму).

Каждый потребитель располагает определенным периодически пополняемым бюджетом. Сам по себе источник финансирования потребителя в модели не рассматривается. На практике такими источниками могут служить системные администраторы, собственные средства потребителей, а также перераспределение кредитов, полученных провайдерами некоторой виртуальной организации, между участниками этой организации. На каждом этапе моделирования потребители "привязаны" к арендным ценам тех ресурсов, на которых размещены их задания. Предполагается, что потребители не пытаются "сэкономить" бюджетные деньги, а, напротив, намерены их потратить. При этом расходы должны распределяться в промежутках между пополнением бюджета равномерно, поскольку у потребителей нет надежной оценки времени выполнения заданий.

Действуя на "товарном рынке" ресурсов, потребитель исходит из вектора цен C, который сформировался на рынке, где компонента C_i означает цену за единицу ресурса, имеющего относительную производительность p_i. В зависимости от характера заданий потребитель должен определить, какие категории ресурсов для него предпочтительнее. Предпочтение выражается коэффициентами k_i. Таким образом, скорректированная цена D_i каждой категории ресурсов выражается формулой

$$D_i = \frac{C_i}{p_i k_i}.$$

Использование коэффициента k_i является простым выражением довольно сложной логики, которой может следовать потребитель, отдавая предпочтение тому или иному ресурсу. Например, потребитель может быть готов заплатить более чем двойную цену за процессор первой категории, всего в два раза более быстрый, чем процессор категории 2. Или, например,

если граф задания содержит некоторые "критические" узлы, то, желая минимизировать общее время выполнения задания, потребитель готов заплатить больше номинальной стоимости за размещения именно этой его части, являющейся наиболее важной с точки зрения времени выполнения.

На аукционном рынке ресурсов потребители должны определять свое поведение по отношению к конкретным процессорам и решать, какие средства они могут потратить на аренду необходимых мощностей, а также в каких аукционах им следует участвовать. Они вычисляют базовый уровень предложения цены (bid), зависящий от оставшейся части бюджета и соотносят его с характеристиками конкретного ресурса. Получается следующее соотношение для предложенной цены:

$$b_c = Bp_ik_i,$$

где B — базовая цена, p_c и k_c — соответственно относительная производительность и оценочный коэффициент, относящиеся к конкретному процессору c.

Каждый провайдер располагает некоторым количеством ресурсов, которые он предоставляет на товарный рынок, где цены для различных категорий процессоров устанавливаются в процессе моделирования динамически. На каждом шаге имитационного процесса решается специальная оптимизационная задача, которая определяет равновесные цены. При этом все участники рынка опрашиваются с целью выяснения уровня предложения и спроса для каждой категории процессоров. Эта информация используется для определения так называемой поверхности избыточного спроса (excess demand surface), устанавливающей разницу между спросом и предложением как функцию вектора цен. Соответствующий алгоритм изложен в [75].

На аукционном рынке каждый провайдер располагает некоторым количеством "единичных аукционов Викрея" (single-unit Vickrey auctions), по одному на каждый доступный в данный

момент процессор. Потребители предлагают аукционистам свои цены за интересующие их процессоры. Аукцион Викрея отдает процессор потребителю, который предложил наибольшую цену, но по договорной цене, соответствующей второму по величине предложению (или бесплатно, если на торгах выступал один покупатель). Договорная цена для потребителя не зависит от той цены, которую он сам предлагает, что обеспечивает так называемую "поощрительную совместимость" аукциона Викрея (incentive compatibility of the Vickrey auction). Действительно, у потребителя нет стимула предлагать цену (bid), отличную от истинной цены за данный процессор, поскольку такая стратегия не приносила бы никакой выгоды.

Оба предложенных способа организации рынка, на котором взаимодействуют потребители и поставщики вычислительных ресурсов были проанализированы с помощью компьютерного моделирования с позиций стабильности цен, справедливости распределения ресурсов, а также коммуникационных и вычислительных затрат. Товарный рынок обнаружил большую стабильность цен и характера распределения ресурсов. В то же время этот способ управления имеет свой недостаток, он менее приспособлен для детальной оценки ситуации участниками (fine-grained valuations). Причина в том, что коммуникационные затраты слишком высоки. Организация аукционов дает менее устойчивые параметры, однако сопряжена с меньшими коммуникационными требованиями и позволяет участникам лучше ориентироваться в ситуации на рынке.

§4. Архитектурные решения задачи

Одним из предлагаемых архитектурных решений рассматриваемой задачи является архитектура грид-федерации (Grid Federation). Она определяется как логическая совокупность кластерных ресурсов, соединенных по принципу P2P [91] и принадлежащих различным организациям, различным административным и временным зонам [92, 93]. Единичной составной частью в модели является "вычислительное устройство" (machine), под которым подразумевается одно- или многопроцессорная система, обладающая памятью, устройствами ввода-вывода и операционной системой. Под "кластером" в модели понимается совокупность одинаковых вычислительных устройств, которые соединены высокоскоростной сетью, наподобие мегабайтного или гигабайтного Ethernet. В грид-федерации имеются системы управления ресурсами (Resource Management System, RMS), которые выполняют как локальное, так и глобальное планирование. Каждый кластер имеет свою RMS, которая осуществляет локальное планирование. Кроме этого определяется еще одна RMS, которая называется агентом грид-федерации (Grid Federation Agent, GFA). Эта система регулирует взаимодействие кластера с остальной частью грид-федерации.

Средняя интенсивность потока заданий, поступающих на RMS кластера i, обозначается через $\lambda_{C_i}, i = 1, \ldots, n$, где n — количество кластеров в системе. Средняя интенсивность обслуживания в RMS i-го кластера обозначается через μ_{C_i}.

Потоки заданий разделяются на локальные с интенсивностью λ_{P_i} и удаленные, поступающие с кластеров $j \neq i$ с интенсивностью $\lambda_{G_j^{out}}$. Обозначм среднюю интенсивность заданий, поступающих в систему GFS i-го кластера через λ_{G_i}, тогда

$$\lambda_{G_i} = \sum_{j=1, j\neq i}^{n} \lambda_{G_j^{out}} + \lambda_{P_i} + \mu_{C_i},$$

$$\lambda_{C_i} = \lambda_{G_i} - \lambda_{G_j^{out}} - \mu_{P_i},$$

где μ_{P_i} — интенсивность выполнения локальных заданий.

Задание состоит из конечного набора задач, последовательность выполнения которых определяется направленным ациклическим или циклическим графом. Каждый кластер имеет собственный набор ресурсов, характеризующихся следующими параметрами: тип процессора, тип операционной системы, количество доступной физической памяти, доступный объем вспомогательных запоминающих устройств, количество узлов в кластере, доступная библиотека программ. Тот же набор параметров, но только необходимый для выполнения собственного приложения, задает потребитель, описывая задание.

Экономическая модель грид-федерации предполагает наличие собственника каждого кластера, а также наличие цены за использование каждого ресурса в единицу времени.

Перейдем к описанию следующей модели. Она использует двухуровневую схему планирования [94]. На первом уровне брокер ресурсов грида осуществляет распределение заданий между параллельными компьютерами. Как правило, брокер отвечает за поиск и отбор нужного ресурса для данного задания, так, чтобы удовлетворялись основные условия, выдвинутые потребителем, а также цели владельца ресурса. Задача брокера — обеспечить централизованный доступ к распределенным ресурсам. Второй уровень планирования относится непосредственно к вычислительным узлам грида, в которых осуществляется локальное распределение заданий.

Предположим, что имеется n заданий $J_1, \ldots, J_n$, а также m узлов грида $N_1, \ldots, N_m$, которые характеризуются набором $M = [m_1, \ldots, m_m]$, где m_i — количество (идентичных) процессоров в узле N_i. Предполагается, что любое задание может выполняться в любое время, в любой последовательности и в любом узле. Каждое задание характеризуется парой чисел

$(s_j, p^j_{s_j})$, где s_j — указывает количество процессоров, необходимых для выполнения задания J_j, а $p^j_{s_j}$ — время выполнения данного задания. Объем работы, соответствующий заданию J_j, определяется как $W_j = p^j_{s_j} \cdot s_j$.

Каждое задание должно быть выполнено на одном узле, отсюда вытекает необходимость того, чтобы размер задания не превосходил количества процессоров в узле.

Сравнительный анализ стратегий планирования будем осуществлять по следующему критерию. Пусть $C_{opt}(I)$ означает время выполнения всех заданий при оптимальном распределении для задачи I, а $C_A(I)$ — значение того же функционала для стратегии A той же задачи I. Стратегия A характеризуется отношением

$$\rho_A = \sup_I C_A(I)/C_{opt}(I).$$

При этом стратегия A называется "ρ-аппроксимацией", и если указанное число ρ существует, то о стратегии A говорят, как о стратегии, доставляющей постоянную аппроксимацию (constant approximation). Как отмечалось выше, схема планирования является двухступенчатой. На первом этапе выбирается узел, в котором будет выполняться задание, а на втором этапе осуществляется локальное планирование.

Для выбора узлов, в которых будут выполняться задания, используется вся доступная информация о требованиях потребителей и возможностях ресурсов. В качестве компонент такой информации выступает нагрузка — номера заданий в каждой локальной очереди, параллельная нагрузка — сумма размеров заданий, объем нагрузки — сумма объемов заданий, и т.д.

Предположим, что все задания упорядочены по возрастанию их размеров: $m_1 \leq m_2 \leq \ldots \leq m_m$. Пусть $first(J_j)$ означает минимум таких значений i, что $m_i \geq s_j$, а $last(J_j)$ — максимум значений i, что $m_i \geq s_j$. Если $last(J_j) = m$, то узлы N_i, для

которых $i = first(J_j), \ldots, last(J_j)$, объявляются допустимыми (available), а их множество обозначается через *M-avail*.

Если $last(J_j)$ является минимумом таких значений r, что

$$\sum_{j=first(J_j)}^{r} m_i \geq \frac{1}{2} \sum_{j=first(J_j)}^{m} m_i,$$

то узлы N_i, для которых $i = first(J_j), \ldots, last(J_j)$, объявляются приемлемыми (admissible), а их множество обозначается через *M-admis*.

Брокер выбирает узел для выполнения задания, руководствуясь следующими стратегиями:

стратегия ML (Min-Load) — выбирается узел с наименьшей нагрузкой на процессор (количество заданий, отнесенное к количеству процессоров в узле);

стратегия MPL (Min-Parallel-Load) — выбирается узел с наименьшей параллельной нагрузкой на процессор (сумма размеров заданий, отнесенная к количеству процессоров в узле);

стратегия MLB (Min-Lower-Bound) — выбирается узел с наименьшей нижней оценкой времени выполнения для ранее назначенных заданий, то есть узел с наименьшим объемом работ в расчете на один процессор;

стратегия MCT (Min-Completion-Time) — отличается от предыдущего варианта способом оценки времени выполнения. Детали можно найти в [95-97].

Локальное планирование рассматривается как задача "упаковки прямоугольников", ассоциируемых с заданиями, в полосы различной ширины. Соответствующие алгоритмы изложены в [98-100]. В частности, используется вариант стратегии упаковки, который называется LSF (Lager Size First). Для описанных модификаций стратегии брокера в сочетании с локальной стратегией LSF получены следующие результаты:

Для грида с идентичными процессорами и заданным множеством заданий стратегии MCT-LSF и MLB-LSF, вообще говоря, не обеспечивают постоянную аппроксимацию.

Для любого множества заданий и любого грида с идентичными процессорами стратегии MCT-LSF и MLB-LSF, использующие множество приемлемых узлов M-admis, являются 10-аппроксимацией.

В [101] предлагается архитектура грид, состоящая из трех ярусов: яруса брокера ресурсов (Resource Broker Tier), яруса управляющих ресурсами грид (Grid Head Tier) и вычислительного яруса (Computing Tier).

Ярус брокера ресурсов выступает как координатор между потребителями и провайдерами ресурсов. В этом ярусе выделяются два модуля: центральный модуль (Kernel Module) и модуль планирования нагрузки (Load Balancing Module). Центральный модуль включает три уровня. Первый уровень представляет собой интерфейс командных строк пользователя (Command-Line User Interface) и обеспечивает дружеский интерфейс пользователя при подаче заявок. Второй уровень — управляющий центр, который получает запросы с первого уровня и выбирает для их выполнения подходящего менеджера на третьем уровне. Третий уровень включает в себя три типа менеджеров: менеджер выполнения (Execution Manager), менеджер передачи (Transfer Manager) и менеджер информации (Information Manager).

Модуль планирования нагрузки состоит из блока измерений нагрузки с помощью нечеткой логики (Fuzzy Logic Workload Measurement) и блока обучаемой нейронной сети (Neural Network Training Component). В задачу первого из этих блоков входит оценка степени нагрузки каждого вычислительного узла. Второй блок осуществляет автоматическую настройку используемой функции предпочтения нечетких переменных.

Ярус управляющих ресурсами грид является ответственным за доставку заданий соответствующим вычислительным узлам и за взаимодействие с брокерами ресурсов. Каждый такой управляющий (head) состоит из четырех компонент системы Globus Toolkit (GRAM, RFT, MDS, PBS) и из блока наблюдения за нагрузкой (Workload Observer). Компонента GRAM (Grid Resource Allocation Manager) используется для получения команд от брокера ресурсов. Компонента RFT (Reliable File Transfer) собирает результаты вычислений в узлах и сообщает о них брокеру ресурсов. Компонента MDS (Monitoring and Discovery System) накапливает интегрирующую системную информацию о вычислительных узлах и также сообщает ее брокеру ресурсов. Компонента PBS (Portable Batch System) является локальным планировщиком, который направляет задания в вычислительные узлы. Наконец, блок наблюдения за нагрузкой предназначен для дополнительного сбора информации, если компонентой MDS ее собрано недостаточно.

Вычислительный ярус состоит из вычислительных узлов, в которых происходит выполнение заданий.

Блок измерения нагрузки с помощью нечеткой логики использует в качестве входных нечетких переменных "использование центрального процессора" (CPU Utilization) и "использование памяти" (Memory Utilization). При этом использование центрального процессора включает скорость процессора, загрузку процессора и очередь доступа к процессору. Использование памяти учитывает общую память и загрузку памяти. Оба типа нечетких переменных нормализованы и характеризуются функцией предпочтения, заданной соотношениями:

$$Light(x) = \begin{cases} 1, & \text{если } x < 0, \\ 1 - 2x, & \text{если } 0 \le x \le 0,5; \end{cases}$$

$$Medium(x) = \begin{cases} 2x, & \text{если } 0 \le x < 0,5, \\ 2 - 2x, & \text{если } 0,5 \le x < 1; \end{cases}$$

$$Heavy(x) = \begin{cases} 2x - 1, & \text{если } 0,5 \leq x < 1, \\ 1, & \text{если } x > 1. \end{cases}$$

Правила вывода, которые используются для вычисления нагрузки показаны в табл. 1. С помощью методов нечеткой логики с их помощью может быть получена степень нагрузки каждого вычислительного узла.

Табл. 1

Memory / CPU	Light	Medium	Heavy
Light	Very Light	Medium	Heavy
Medium	Light	Medium	Heavy
Heavy	Medium	Heavy	Very Heavy

Блок обучаемой нейронной сети включает 5 уровней: входной уровень, скрытый уровень X, скрытый уровень Y, скрытый уровень Z и выходной уровень. Функционирование блока распадается на две фазы: прямого и обратного действия. В фазе прямого действия (Feed-Forward Phase) информация о нагрузке, включающая использование центрального процессора и памяти, поступает на входной уровень.

Входные переменные обрабатываются индивидуально с помощью трех нечетких множеств, заданных на скрытом уровне X. Скрытый уровень Y отражает уже девять нечетких правил. Например, правило R1 указывает, что если использование центрального процессора является "слабым" (Light), также как и использование памяти является слабым, то нагрузка является "очень слабой" (Very Light). На этом уровне нейрон вычисляет минимальное значение сигнала, полученного с уровня X, который представляет собой степень нечеткого совпадения условия, заключенного в правиле вывода. Данное минимальное значение посылается затем на примыкающий нейрон уровня Z, где для завершения процесса вывода согласно максиминному правилу применяется оператор максимума. Выходной уровень за-

вершает вычисление степени нагрузки с помощью метода "центра гравитации". Аналогичным образом устроена фаза обратного действия (Back-Propagation Phase).

§5. Проблема моделирования нагрузки

Изучение динамики нагрузки в кластерах и гридах проводилось в работах [102, 103]. Оказалось, что характер нагрузки для этих сред существенно отличается от нагрузки, обнаруживаемой в обычных суперкомпьютерах. В частности, потоки заданий обнаруживают ряд эффектов, таких, как краткосрочную или долгосрочную зависимость (short and long range dependence), а также псевдопериодичность (pseudo-periodicity). Простые модели, основанные на пуассоновском распределении не в состоянии отразить свойства второго порядка, необходимые для изучения рассматриваемых объектов. Ниже рассматриваются некоторые модели нагрузки в гриде, способны сравнительно адекватно отразить реальные траектории.

Для разработки и изучения новых алгоритмов распределения ресурсов в гриде необходимы две принципиально важные вещи: наличие правдоподобных траекторий нагрузки и надежная среда, в которой можно было бы проводить эксперименты. Говоря о моделировании нагрузки, следует сразу отметить, что в гриде практически всегда выделяется нагрузка локальная, или фоновая (local or background), и нагрузка глобальная, характерная собственно для грида. Моделирование последней представляет основные трудности из-за ее недостаточной изученности. При изучении алгоритмов распределения ресурсов часто делают упрощающие предположения. Например, в работах [32, 104] при анализе указанных алгоритмов, предназначенных для работы в режиме off-line, предполагалось наличие совокупности заданий в количестве от 200 до 1000. Разновид-

ности таких алгоритмов, работающих в режиме on-line, изучались в предположении, что задания поступают через равные промежутки времени [105], либо строго последовательно [106].

Более реалистичный подход связан с использованием реально замеренных траекторий нагрузки. Он применялся, например, в работах [92, 107-109]. Однако этот подход связан с трудностями получения реальных траекторий в количестве, необходимом для статистического изучения алгоритмов распределения нагрузки.

Поступление заданий может быть описано как точечный процесс. Для его описания используют два основных приема. В первом случае задается распределение времени между событиями (Interarrival Time Process). Второй способ заключается в дискретном разбиении временной оси и задании распределения количества событий на каждом промежутке (Count Process).

Довольно часто процесс поступления заданий описывается дважды стохастическим пуассоновским процессом (Doubly Stochastic Poisson Process). Он отличается от обычного пуассоновского процесса тем, что параметр (интенсивность) зависит от времени и в общем случае является самостоятельным случайным процессом с непрерывным временем и положительными значениями. В частном случае характер изменения параметра регулируется конечной марковской цепью (Markov Modulated Poisson Process). Последняя модель допускает сравнительно легкий математический анализ и способна описать ряд фактических эффектов, таких как корреляция промежутков между поступлениями заданий или их кратко- и среднесрочную зависимость [110].

Другой важный, экспериментально обнаруженный при изучении нагрузки в гриде, эффект называется псевдопериодичностью. Моделирование этого явления осуществляется с помощью техники "поиска с подгонкой" (matching pursuit) [111].

Процесс $X(t)$ обладает долгосрочной зависимостью (Long Range Dependent, LRD), если его автокорреляционная функция удовлетворяет условию

$$R(k) \sim c_r k^{\alpha-1}, k \to \infty.$$

В этом случае автокорреляционная функция убывает так медленно, что

$$\sum_{k=-\infty}^{\infty} R(k) = \infty.$$

Указанный процесс является одним из вариантов так называемых самоподобных (selfsimilar) процессов. Долгосрочная зависимость, которую отражает данная модель, возникает на разных уровнях града, в частности, в кластерах и виртуальных организациях [103].

2. Оперативное адаптивное управление потоком заданий в подсистеме распределенных вычислительных ресурсов

§1. Общее описание модели оперативного распределения заданий в замкнутой подсистеме ресурсов

В данном разделе излагается модель распределения ресурсов в системе грид. Предполагается, что уже реализован "этап предварительного планирования", итогом которого явилось закрепление потребителей за определенными подсистемами ресурсов, и который осуществлялся с учетом самых общих интегральных потребностей потребителей. Модель предварительного планирования здесь не описывается, примеры такой модели содержатся в работах [112, 113].

После осуществления процедуры предварительного планирования начинается непосредственное выполнение заданий. Моделируя этот "этап оперативного управления", исходим из предположения о том, что каждый потребитель является источником потока заданий, и что эти задания должны распределяться строго внутри той подсистемы ресурсов, которая была определена на этапе планирования.

Поток заданий в модели предполагается случайным и необрывающимся (или достаточно продолжительным), а его интегральные параметры соответствуют тем характеристикам потребителей, которые были использованы на этапе предварительного распределения ресурсов. В то же время для описания потоков заданий необходимы дополнительные параметры, поскольку речь идет уже о динамической модели.

Рассматриваемая задача относится к определенной замкнутой подсистеме ресурсов. При этом подчеркнем, что алгоритм предварительного планирования допускает закрепление за подсистемами ресурсов нескольких потребителей.

Задача заключается в том, чтобы предложить стратегию распределения ресурсов в подсистеме в процессе поочередного выполнения поступающих заданий. Эта стратегия должна учитывать интересы всех потребителей, присутствующих в данной подсистеме. Еще более важным является то, что эта стратегия не должна опираться на знание параметров входных потоков заданий, поскольку общие показатели, использованные на предварительном уровне планирования, являются очень приблизительными, а более точные параметры на практике просто неизвестны.

Вся информация, которую может использовать алгоритм распределения заданий в рассматриваемой модели — это более или менее детальное отражение состояния системы (текущее количество заданий, загрузка ресурсов и т.п.), а также возможность в какой-то мере запоминать предысторию этих состояний. По существующей терминологии подобного рода алгоритмы принято называть адаптивными стратегиями.

Перейдем к неформальному описанию замкнутой системы, включающей в себя распределенные вычислительные ресурсы и потребителей этих ресурсов. Хотя эта система является абстрактным, гипотетическим объектом, есть основания считать, что выделенные черты, которые являются предметом анализа — в данном случае это особенности процесса загрузки вычислительных мощностей, присущи если не всем, то, по крайней мере, большинству реальных объектов. В данной работе не ставилась задача сопоставления модели с конкретными приложениями, но все же в качестве примера, не единственно возможного, укажем хотя бы на систему X-COM [114].

Рассмотрим гипотетическую информационно-вычислительную систему, основными элементами которой являются вычислительные ресурсы и потребители ресурсов. Эти элементы соединены между собой каналами связи, которые, вообще говоря,

могут быть частью неких общих, даже глобальных сетей.

Вычислительный ресурс будем трактовать либо как компьютер (процессор), способный выполнять операции вычислительного характера, либо как совокупность компьютеров (процессоров), объединенных в локальную сеть (кластер). В любом случае вычислительный ресурс является "элементарной" составной частью рассматриваемой системы, внутреннее устройство такого элемента не рассматривается. Вычислительные ресурсы обладают, вообще говоря, разными характеристиками, в том числе различной производительностью, разной удаленностью от пользователей, разной степенью надежности, разной стоимостью доступа к ним.

Потребители ресурсов являются источниками потоков заданий. Каждое задание должно быть выполнено путем обращения к одному или нескольким вычислительным ресурсам. Время существования задания в системе ограничено, и если за это время задание не выполнено, то оно считается потерянным и покидает систему.

Каждое задание состоит из некоторого линейно упорядоченного набора задач. Выполнение задания заключается в обработке отдельных задач на вычислительных ресурсах в последовательности, строго соответствующей их упорядоченности, т.е. задача может поступить на обработку в процессоре только в том случае, если обработаны все предшествующие ей задачи. Задачи из задания могут посылаться на обработку в виде пакетов. Пакет может состоять только из задач, относящихся к одному заданию, причем включать произвольный начальный отрезок из совокупности еще нерешенных задач.

Каждая задача характеризуется параметром, который будем называть объемом задачи. Объем пакета определяется, как сумма объемов входящих в него задач. Объемы пакета и входящих в него задач определяют такие показатели, как временные

задержки при передаче по сети и при обработке на ресурсах, и, аналогично, стоимостные затраты.

Пакеты задач поступают на вычислительные ресурсы, где образуют очереди, обслуживаемые по принципу "первый пришел — первый обслуживается". Один ресурс одновременно может обрабатывать один пакет, выполняя по очереди все задачи из этого пакета. Полностью выполненный пакет возвращается потребителю, после чего он считается покинувшим систему.

Может возникать ситуация, когда пребывание пакета в ресурсе прерывается до окончания его обработки. Это может происходить вследствие двух обстоятельств. Во-первых, предполагается наличие механизма, ограничивающего очередь в ресурсе. Он заключается в том, что время пребывания пакета в ресурсе ограничивается заданным промежутком, по истечении которого пакет возвращается потребителю. Во-вторых, существует вероятность прерывания обслуживания пакета, не превысившего допустимое время пребывания — это отражает такие ситуации, как возможность появления более приоритетных, с точки зрения владельца ресурса, заданий или же технический сбой. В обоих случаях пакет покидает ресурс независимо от того, находится он в очереди ожидания или обрабатывается на процессоре, причем все задачи из этого пакета считаются невыполненными. Невыполненный возвращенный пакет расформировывается, а входившие в него задачи ожидают выполнения в одинаковом режиме с остальными задачами.

Таким образом, время, затрачиваемое на посылку одного пакета, складывается из сетевых задержек при передаче пакетов задач от потребителей к исполнителям и обратно, из времени пребывания в очереди ресурса, а также из времени непосредственной обработки на процессоре. Выполнение задания сопряжено, следовательно, с посылкой одного или более пакетов, в зависимости от того, сколько задач включается в пакеты,

и от наличия возвращенных невыполненных пакетов.

Функционирование описанной системы предполагает наличие алгоритма выбора двух параметров, связанных с отправкой пакетов. Посылая пакет, необходимо указать объем этого пакета, а именно — количество включенных в него задач, и определить ресурс, на который данный пакет будет направлен. Априори можно предположить, что стратегия выбора обоих параметров отразится на качестве функционирования.

Воздействие выбора ресурса на качество работы системы представляется наиболее очевидным. Действительно, если все потребители посылают пакеты на один и тот же ресурс, то это может привести к его перегрузке, при этом остальные ресурсы будут простаивать. Но достаточно правдоподобно и то, что объемы пакетов также могут оказывать существенное влияние на качество работы системы. Слишком большие пакеты, адресованные ресурсам с жесткими ограничениями на размер очереди или подверженные сбоям, с высокой вероятностью будут возвращаться невыполненными, а посылка пакетов маленького объема может порождать дополнительные потери, связанные с ожиданием в очереди и транспортными издержками.

Оценка качества функционирования системы может осуществляться весьма разнообразными способами. Прежде всего, имеется большой выбор показателей, которые заслуживают внимания с точки зрения их оптимизации. Эти показатели можно разделить на две группы, условно называемые операционные и стоимостные. К операционным показателям отнесем количество выполненных заданий, время выполнения заданий, объем выполненных заданий, количество потерянных (невыполненных) заданий, загрузка ресурсов. К стоимостным — затраты на аренду ресурсов, затраты на транспортировку по сети, доход от использования ресурсов, штраф за невыполненные задания или за задержку в их выполнении.

В терминах перечисленных показателей возможны разнообразные постановки задач, связанных с безусловной или условной оптимизацией.

§2. Формализация модели управления распределением ресурсов

Рассматривается система, состоящая из M *потребителей* и N *ресурсов*. Система функционирует в дискретном времени $t = 0, 1, ...$ Каждый потребитель получает входной *поток заданий*, которые выполняются путем посылки их ресурсам. Механизм образования потока заданий для потребителя $i = 1, \dots M$, задается случайным индикаторным процессом, где значение $\xi_t^{(i)} = 1$ соответствует поступлению в момент t нового задания. Хотя это не является принципиальным ограничением для модели, но в обсуждаемом варианте полагаем, что $\xi_t^{(i)}$ являются независимыми в совокупности процессами с независимыми значениями, причем $Pr\{\xi_t^{(i)} = 1\} = \alpha_i, 0 \le i \le 1$ — *интенсивность входного потока*.

Задание, появившееся в момент t у потребителя i, содержит некоторое количество $\nu_t^{(i)}$ задач. Случайные величины $\nu_t^{(i)}$ являются условно независимыми в совокупности, принимающими значения из множества $\{1, \dots, K_i\}, K_i$ — *максимальное число задач в задании*. В качестве конкретного варианта можно предположить, что распределения

$Pr\{\nu_t^{(i)} = n \mid$ потребитель i получил задание в момент $t\ \}$

являются равномерными на указанном множестве. Каждая задача имеет свой *объем*, который определяется каждый раз независимо как реализация случайной величины, распределенной на множестве $\{1, \dots, Vi\}, Vi$ — *максимальный объем задачи*. Объем задания представляет собой сумму объемов всех входящих в него задач.

Максимальное время пребывания произвольного задания в системе для пользователя i считается детерминированным и равным T_i — *максимальное время жизни задания.*

Таким образом, потребитель i характеризуются набором параметров $U_i = (\alpha_i, K_i, V_i, T_i)$. Обозначим $U = (U_1, \ldots, U_M)$.

Все задачи задания упорядочены и ни одна, кроме первой, не может быть выполнена, если перед этим не выполнены все предыдущие. Полученные в разные моменты времени задания могут накапливаться у потребителя, вообще говоря, в неограниченном количестве. Задания, имеющиеся в данный момент у потребителя, группируются в двух очередях, O_1 и O_2. Очередь O_1 содержит задания, ожидающие отправки для выполнения. Вновь поступившее задание помещается в конец этой очереди. Очередь O_2 составляют те задания, которые полностью или частично находятся либо в процессе передачи, либо в процессе обслуживания на ресурсе.

В каждый момент времени, если очередь O_1 не пуста, потребитель отправляет один *пакет* задач, который может быть адресован любому из имеющихся ресурсов. Этот пакет формируется из задания, находящего в голове очереди O_1. Предположим, что такое задание содержит ν задач, из которых первые κ задач уже к данному моменту выполнены. Тогда пакет может быть сформирован одним из следующих способов: $\{\kappa+1\}, \{\kappa+1, \kappa+2\}, \ldots, \{\kappa+1, \ldots, \nu\}$. Здесь в фигурных скобках указаны номера задач, включаемых в пакет, а сами номера соответствуют упорядоченной последовательности задач, образующей задание.

Сформированному пакету назначается адресат — тот ресурс, на котором предполагается его выполнение, после чего пакет направляется в сеть. Задание, из которого был сформирован пакет, покидает очередь O_1 и переходит в очередь O_2. Упорядоченность в этой очереди не имеет значения.

Пакет, возвращенный после обслуживания, может быть либо полностью выполнен, либо полностью не выполнен. В первом случае, если задание, из которого сформирован пакет, выполнено (с его доставкой) полностью, то оно покидает систему. Если же в данном задании остались невыполненные задачи, то оно переходит из очереди O_2 в начало очереди O_1. Во втором случае задание определенно совершает переход из очереди O_2 в начало очереди O_1.

Каждый ресурс получает на каждом такте пакеты заданий от потребителей, не более M пакетов на каждом такте. Эти пакеты образуют единую очередь, образованную по принципу FIFO. *Производительность ресурса j* на одном такте равняется a_j. Это означает, что на каждом такте ресурс j выполняет ровно $\min\{a_j, g_j\}$ объема заданий, последовательно обрабатывая задачи в соответствии с их упорядочением внутри заданий. Здесь g_j — суммарный объем заданий в очереди ресурса в данный момент. Полностью выполненный пакет заданий отправляется в сеть для обратной передачи потребителю.

Размер буфера входной очереди для ресурса задан косвенно. Предполагается, что *максимальное время обслуживания одного пакета в ресурсе j* не должно превосходить величины d_j, которая является вторым постоянным параметром, характеризующим ресурс. Если время пребывания пакета в очереди ресурса достигло указанного значения и не все задания из него выполнены, то пакет отправляется обратно потребителю как полностью невыполненный. Там он присоединяется к общему объему невыполненных и неразличимых заданий.

В каждый момент времени существует *вероятность сбоя ресурса β_j*, то есть вероятность того, что обслуживание всех пакетов, находящихся на ресурсе j, будет прекращено, и они будут отправлены обратно потребителю как полностью невыполненные.

Наконец, будем предполагать, что для каждого ресурса j задана величина c_j — *стоимость единицы ресурса* , назначаемая за выполнение единицы задания пользователя.

Таким образом, каждый ресурс j характеризуется набором параметров $R_j = (a_j, d_j, \beta_j, c_j)$. Обозначим $R = (R_1, \ldots, R_N)$.

Сетевой аспект задается с помощью *расстояний* l_{ij} между пользователем i и ресурсом j. С их помощью задержка на передачу пакета объемом w между пользователем i и ресурсом j в любом направлении определяется как kwl_{ij}, где k — некоторая константа, *сетевой коэффициент*. Кроме того, обозначим через C — *тариф,* т.е. стоимость передачи по сети единицы объема задания.

Таким образом, сетевые параметры представляют собой набор $S = (\mathbf{l}, \mathrm{k}, \mathrm{C})$, где $\mathbf{l}$ — матрица с элементами l_{ij}.

Итак, вся система описывается набором параметров

$$G = G_{M,N} = (M, N, R, U, S).$$

Рассмотрим систему с набором параметров G. Для описания возможных состояний системы обозначим упорядоченный список заданий, одновременно в ней присутствующих, через $x = (Z_1, \ldots, Z_\mu)$, здесь — μ количество заданий в системе, а Z_i — задания. Этот список является переменной величиной.

Каждое задание в системе можно описать в виде набора

$$Z = (\upsilon, \tau, \tau_{loc}, H, \rho, \sigma, \delta, o, \iota, pack),$$

составляющие которого имеют следующий смысл:

υ — номер потребителя, которому принадлежит задание, $1 \leq \upsilon \leq M$;

τ — таймер, время с момента появления задания в системе;

τ_{loc} — локальный таймер, время пребывания в очереди или в сети;

H — упорядоченный список задач, составляющих задание;

ρ — номер ресурса, на который направлен пакет, сформированный из задач данного задания. $\rho = j > 0$, если задание

направлено на ресурс j, и $\rho = 0$, если задание еще находится в очереди O_1;

σ — статус задания относительно сети; $\sigma = 0$, если задание не находится в состоянии передачи по сети; $\sigma = 1$, если задание передается от потребителя к ресурсу; σ = -1, если задание передается от ресурса к потребителю;

δ — сетевая задержка; $\delta = 0$, если задание не находится в состоянии передачи по сети;

o — номер очереди, в которой находится задание; $o = 1$, если задание находится в очереди O_1, и $o = 2$, если задание находится в очереди O_2;

ι— номер в очереди, если $\iota \geq 0$, и $\iota = -1$, если $\iota = -1$;

pack — указывает на наличие и состав пакета задач из задания, отправленных для выполнения. Будем считать, что *pack* не определено (*pack*=[ ]), если задание находится в очереди O_1; и $pack = [n1, n2]$, если сформирован пакет, состоящий из задач с номерами от n_1 до n_2 из списка H.

Упорядоченный список задач имеет вид $H = (A_1, \ldots, A_\nu)$, где ν — количество задач в задании, а A_k — задачи. Каждая задача может быть описана в виде, где $A = (v, w)$, где v — объем задачи, подлежащий выполнению, а w — выполненный к данному моменту объем.

Функционирование системы разбивается на такты. В течение каждого такта происходят процессы образования новых заданий, формирование и рассылка пакетов, выполнение заданий на ресурсах и пр. Алгоритм выполнения этих процессов одинаков для всех тактов, и его описание проведем в терминах перехода от состояния системы к концу такта с номером t, которое обозначим через x_t, к состоянию в конце такта с номером $t+1$. Пусть $x_t = x, x_{t+1} = x'$. Далее излагаются процедуры, последовательное выполнение которых приводит к переходу системы из состояния x в состояние x'.

Процедура удаления заданий, превысивших допустимое время пребывания в системе. Для всех заданий из списка x, которые находятся к этому моменту в очереди $O_1(o = 1)$ делается проверка выполнения условия $\tau > T_v$. Если это условие выполнено, то данное задание удаляется из списка, т.е. покидает систему. Одновременно для оставшихся в списке заданий происходит коррекция номеров ι, указывающих на местоположение заданий в очередях системы. Получившийся в результате процедуры список обозначим через $\tilde{x}$.

Процедура поступления новых заданий. Для всех потребителей независимо разыгрываются бернуллиевские случайные величины с распределениями $\alpha_i, i = 1, \ldots, M$. "Успех" соответствует образованию нового задания, которое добавляется к списку $\tilde{x}$. Таким образом, на каждом такте для данного потребителя может появиться не более одного задания. В момент образования нового задания у потребителя i, с помощью параметров N_i и V_i определяются случайное число задач в задании ν и случайный объем каждой задачи v. Это новое задание описывается в виде $(i, 0, 0, ((v_1, 0), \ldots, (v_\nu, 0)), -1, 0, 1, 1, \iota, [\ ])$, где ι — номер после помещения в конец очереди O_1 для заданий, ожидающих отправки на выполнение.

З а м е ч а н и е. С точки зрения написания и последующего исполнения программного компьютерного кода очередь O_1 должна, конечно, фигурировать как самостоятельный объект. Однако с точки зрения описания системы, непосредственное задание этой очереди избыточно, поскольку она восстанавливается путем просмотра списка заданий, т.е. однозначно восстанавливается по состоянию системы.

Получившийся в результате процедуры список обозначим через x''. После завершения формирования этого списка для всех заданий, входящих в него, увеличивается на единицу значение компоненты τ.

Процедура образования пакетов. На каждом такте каждый потребитель может сформировать и отправить в сеть не более одного пакета. Для образования пакета берется задание, стоящее в голове очереди O_1, если эта очередь не пуста. Данное задание переходит в очередь O_2. Относительно этой очереди справедливо замечание, сделанное относительно очереди O_1.

Пакет образовывается путем включения в него одной или более задач среди первых невыполненных задач из указанного задания. Напомним, что список задач в задании упорядочен в отношении последовательности их выполнения.

Выбор количества задач, включаемых в пакет, осуществляется специальной процедурой, которая будет описана в следующем пункте. В результате этого выбора заполняется компонента *pack* в описании задания.

После формирования размера пакета, происходит выбор ресурса, на который этот пакет будет направлен. Выбор ресурса также осуществляется специальной процедурой, которая также будет описана в следующем пункте. Компонента ρ принимает значение номера выбранного ресурса. Компоненте σ присваивается значение 1 — это означает, что сформированный пакет поступил в сеть для передачи от потребителя к ресурсу.

Процедура определения сетевых задержек при передаче пакетов от потребителя к ресурсам. Эта процедура выполняется для каждого потребителя над всеми заданиями, которые находятся в очереди O_2 и у которых $\sigma = 1, \delta = 0$. В соответствии со сказанным на стр. 52, сетевая задержка определяется как произведение сетевого коэффициента на расстояние от потребителя до выбранного ресурса и на объем пересылаемого пакета. Заметим, что данная процедура является вполне автономной, и может быть заменена другим алгоритмом определения сетевых задержек.

Процедура проверки завершения передачи по сети от потребителя к ресурсу. Эта процедура производится для каждого потребителя, над каждым заданием из очереди O_2, у которого $\sigma = 1$. Значение счетчика τ_{loc} увеличивается на 1. Затем проверяется условие $\tau_{loc} > \delta$. Если оно выполнено, то компонентам τ_{loc}, σ и δ присваивается значение 0. Компоненте ι присваивается значение номера, на единицу большее, чем номер последнего элемента в очереди заданий, обслуживаемых на ресурсе с номером ρ. Эти очереди, подобно очередям O_1 и O_2, однозначно восстанавливаются путем просмотра всех заданий с аналогичным значением ρ.

Процедура проверки времени пребывания заданий на ресурсе. Задания, находящиеся на обслуживании в ресурсе j, определяются по признакам $o = O_2,\ \sigma = 0,\ \rho = j$. Для каждого такого задания значение счетчика τ_{loc} увеличивается на 1. Затем проверяется условие $\tau_{loc} > d_j$, где d_j — параметр ресурса. Если неравенство выполнено, то параметру σ присваивается значение -1, задание покидает ресурс и возвращается потребителю.

Процедура определения готовности ресурса. Для каждого ресурса j разыгрывается бернуллиевская случайная величина с параметром β_j. "Успех" означает, что ресурс вышел из строя. В этом случае все задания, которые обслуживались на этом ресурсе, получают значение параметра $\sigma = -1$, после чего они возвращаются потребителям.

Процедура обслуживания заданий на ресурсах. Выполнение заданий происходит в очередности их поступления на ресурс. На каждом такте ресурс j может выполнить не более a_j единиц объема заданий в очереди, где a_j — параметр ресурса. Вначале выполняется первая задача из первого задания в очереди. Если ее объем $v_1 < a_j$, то выполняется вторая задача первого задания, или первая задача второго задания,

если первое задание состояло из одной задачи. Если ее объем $v_2 < a_j - v_1$, то выполняется следующая задача и т.д. до завершения списка задач, либо до исчерпания ресурса, выделенного на один такт. Все выполнявшиеся задачи, кроме последней, выполнены полностью. Они получают значение параметров $w = 0$. Последняя выполнявшаяся задача, возможно, будет выполнена не до конца ($w > 0$) и останется в ресурсе на следующий такт. Если в некотором задании выполнены все задачи, то оно получает значение параметра $\sigma = -1, \iota = 0$ и возвращается потребителю.

Процедура определения сетевых задержек при передаче пакетов от ресурсов к потребителям. Эта процедура тождественна описанной выше аналогичной процедуре для передачи в обратном направлении, за исключением того, что обрабатываются задания с признаком $\sigma = -1$.

Процедура проверки завершения передачи по сети от потребителя к ресурсу. Эта процедура совпадает с описанной выше аналогичной процедурой для передачи в обратном направлении, за исключением того, что обрабатываются задания с признаком $\sigma = -1$, а не $\sigma = 1$, значение индекса ι не меняется, индекс ρ получает значение 0.

Процедура обработки доставленных пакетов. Обрабатываются те задания, находящиеся в очередях O_2, для которых $\rho = 0$, т.е. задания с вернувшимися пакетами. Каждое такое задание проходит следующую обработку.

Анализируются задачи из прибывшего пакета,их номера лежат в диапазоне, заданном в признаке *pack*. Если не все задачи из пакета выполнены полностью, т.е. если не для всех задач выполняются условия $v = w$, то для всех задач из пакета восстанавливается признак $w = 0$. Параметр *pack* получает значение [ ]. Если все задачи из задания выполнены, то задание считается выполненным и покидает систему, т.е. удаляется из списка

x''. Если задание не выполнено, то оно переходит в начало очереди O_1. Тогда o получает значение 1, индекс ι — 0, для всех остальных заданий из очереди O_1 индекс ι увеличивается на 1.

Полученный после завершения данной процедуры список заданий является новым состоянием системы, которое будет являться исходным для выполнения всей цепочки описанных процедур на следующем такте $t + 2$.

Подведем итог этого параграфа. Рассматриваемый объект, состоящий из m потребителей и n ресурсов, задается системой параметров $G_{M,N} = (M, N, U, R, S)$, составляющих три группы: параметры, описывающие потребителей (U), параметры, описывающие ресурсы (R) и параметры, задающие сетевое взаимодействие (S). Функционирование системы $G_{M,N}$ представляет собой чередование состояний, причем каждое состояние представляет собой набор дискретных характеристик, а все множество возможных состояний является не более, чем счетным. Переход от одного состояния к другому осуществляется за один такт, в течение которого выполняются алгоритмически точно заданные процедуры, имеющие как детерминированный, так и вероятностный характер.

В ходе трансформации состояния системы в течение одного такта необходимо осуществлять выбор *управлений*, под которыми понимается размер пакета задач, передаваемых на ресурс и номер ресурса, на который этот пакет будет отправлен. Сказанное позволяет трактовать построенную модель, как управляемую (конечную) марковскую цепь. В следующем параграфе этот аспект раскрывается подробнее, и описываются процедуры выбора управлений.

§3. Стратегии управления

Суть управления в рассматриваемой модели заключается в формировании пакетов для всех заданий, которые готовы к отправке на данном такте, если таковые существуют. На каждом такте отправляются задания, находящиеся в голове очередей O_1 всех потребителей. В свою очередь, формирование пакета заключается в двух действиях: 1) выборе ресурса, на который будет отправлен пакет и 2) определении объема пакета, то есть количества задач из задания, которые будут включены в пакет.

Предположим, что выбор ресурса осуществляется с помощью вероятностных распределений на множестве номеров ресурсов $\{1, \ldots, N\}$. Такое распределение для потребителя i обозначается через $p^{(i)} = (p_1^{(i)}, ..., p_N^{(i)})$. Стало быть, если задание, находится в голове очереди O_1 потребителя i, то вероятность того, что пакет, сформированный из этого задания, будет отправлен на ресурс j, равна $p_j^{(i)}$.

Предположим также, что выбор объема пакета для потребителя i осуществляется с помощью вероятностных распределений $q^{(i)} = (q_1^{(i)}, ..., q_{n_i}^{(i)})$, где n_i выше трактовалось как максимальное количество задач, которое может быть в задании. Происходит этот выбор следующим образом. Пусть, например, в обрабатываемом задании осталось κ невыполненных задач. Разыгрывается значение случайной величины с распределением $q^{(i)}$. Если это значение оказалось равным m, то в пакет включаются $\min(\kappa, m)$ задач, взятых в соответствии с упорядочением задач в задании.

Таким образом, векторы $\mathbf{p} = (p^{(1)}, \ldots, p^{(M)})$ и $\mathbf{q} = (q^{(1)}, \ldots, q^{(M)})$ отвечают за характер функционирования системы.

В качестве одношагового показателя качества функционирования системы выберем объем задач, возвращенных на данном такте t как выполненные. Этот объем обозначается через z_t

и может рассматриваться как состоящий из вкладов отдельных потребителей: $z_t = (z_t^{(1)}, \ldots, z_t^{(M)})$. Тогда можно предложить в качестве одного из возможных критериев качества функционирования i-го потребителя величину, которую назовем предельной средней производительностью, и которая зависит от векторов $\mathbf{p}$ и $\mathbf{q}$:

$$W_i(\mathbf{p}, \mathbf{q}) = \liminf_{T \to \infty} T^{-1} \sum_{t=1}^{T} M z_t.$$

Средняя производительность всей системы составит

$$W(\mathbf{p}, \mathbf{q}) = \frac{1}{M} \sum_{i=1}^{M} W_i(\mathbf{p}, \mathbf{q}).$$

Если понимать z_t не как последовательность объемов, а как последовательность одношаговых финансовых затрат или как количество потерянных необслуженных заданий на шаге t, то аналогично получаются целевые функции соответственно *предельных средних затрат* и *предельных средних отказов*.

Естественно ставить вопрос о максимизации или минимизации того или иного критерия, как для системы в целом, так и для отдельных потребителей. В терминологии управляемых марковских цепей это соответствует широко известной постановке задачи о нахождении экстремума предельного среднего дохода (штрафа). Укажем на [115, 116], как на возможные источники основных сведений по этому вопросу и многочисленных дальнейших библиографических ссылок. Хотя эта задача достаточно хорошо изучена, существуют объективные препятствия для непосредственного применения на практике известных методов ее решения. Трудности, если говорить кратко, связаны с тремя обстоятельствами: большим количеством

состояний, ограниченной возможностью их наблюдения и отсутствием априорной информации о переходных вероятностях. Относительно новое направление развития теории — адаптивное управление частично наблюдаемыми марковскими цепями, дает некоторые способы преодоления этих трудностей. Фундаментальное изложение материала содержится в [116]. За рубежом эта тематика интенсивно развивается в настоящее время также под общим названием reinforcement learning. Более подробно см. [117, 118].

Применительно к рассматриваемой задаче адаптивное управление предполагает, что для нахождения значений векторов $\mathbf{p}$ и $\mathbf{q}$, максимизирующих *целевые функции* $W(\mathbf{p}, \mathbf{q})$ (или $Wi(\mathbf{p}, \mathbf{q})$), нельзя пользоваться информацией о распределении случайных величин, входящих в модель. К тому же такая информация на практике, как правило, отсутствует. При этом распределения $\mathbf{p}$ и $\mathbf{q}$ зависят, естественно, от времени и от доступной наблюдению части траектории системы в фазовом пространстве. Эта зависимость должна быть такова, чтобы асимптотически выходить на оптимальные значения $\mathbf{p}$ и $\mathbf{q}$.

Для построения рекуррентного алгоритма пересчета векторов $\mathbf{p}$ и $\mathbf{q}$ по результатам наблюдений был использован градиентный подход к оптимизации конечных частично наблюдаемых марковских цепей [119]. Далее будем называть этот алгоритм *адаптивной стратегией*. Основная идея указанного подхода основана на использовании специфических вероятностных оценок градиента функции предельного среднего дохода, которые, в свою очередь базируются на точных аналитических выражениях для градиента.

Для того чтобы описать идею построения адаптивной стратегии градиентного типа выберем в качестве целевой функции производительность первого потребителя $W_1(\mathbf{p}, \mathbf{q})$. Предположим, что все компоненты вектора $\mathbf{p}$, кроме первой, фиксирова-

ны, и что вектор $\mathbf{q}$ также фиксирован. Будем максимизировать функцию $W_1 = w(p^{(1)})$ по первой компоненте вектора $\mathbf{p}$. Значения переменного вектора $p^{(1)} = (p_1^{(1)}, ..., p_N^{(1)})$ — вектора выбора ресурсов для первого потребителя, на последовательных тактах функционирования системы для краткости обозначим через $\mathbf{b_t} = (b_{t1}, ..., b_{tN}), t = 0, 1, \ldots$

Пусть ε — произвольное число, и $0 < \varepsilon < N^{-1}$. Обозначим

$$B_\varepsilon = \{\mathbf{b} = (b_1, ..., b_K) : \sum_{j=1}^{K} b_j = 1; b_j \geq \varepsilon, j = \overline{1, N}\}.$$

Пусть значение вектора $\mathbf{b_0}$ выбрано произвольно из множества B_ε. Рассмотрим рекуррентную последовательность

$$\mathbf{b_{t+1}} = \prod_\varepsilon (\mathbf{b_t} + a_t \varsigma_t), t = 0, 1, \ldots,$$

где через $\prod_\varepsilon$ обозначен оператор проектирования на множество B_ε, a_t — числовая последовательность, ς_t — векторы одинаковой размерности с вектором $\mathbf{b}$.

Если вектор ς_t представляет собой градиент функции $w(\mathbf{b})$, то данная последовательность представляет собой алгоритм проекции градиента для нахождения максимума функции w на множестве B_ε. Однако точное значение градиента найти довольно трудно даже при известных значениях параметров системы G, которые не предполагаются известными для управляющего органа. Вместо этого используются оценки градиента по результатам наблюдений за траекторией процесса. В данном случае достаточно траектории выполненных объемов задач для первого потребителя. Обозначим через $z_t = z_t^{(1)}$ объем задач, который был возвращен как выполненный первому потребителю на такте номер t.

Для каждого $j = 1, ..., N$, рассмотрим последовательные случайные моменты $\tau_n^{(j)}, n = 0, 1, \ldots$, в которые первый потребитель выбирал в качестве адреса для отправки пакета на

ресурс $j(\tau_0^{(j)} = 0$ для всех $j)$. Обозначим

$$Z_n^{(j)} = \{ \sum_{k=\tau_n^{(j)}}^{\tau_n^{(j)}+\Delta} z_k \}.$$

Величина $Z_n^{(j)}$ обозначает объем задач, которые первый потребитель получил как выполненные в течение Δ тактов после n-го выбора ресурса j в качестве исполнителя очередного пакета задач (Δ — натуральное число, являющееся параметром алгоритма). Положим

$$\vartheta_n^{(j)} = \tau_n^{(j)} + \Delta,$$

и определим кусочно-постоянные функции $\varsigma_t^{(j)}$, которые изменяют значения лишь в моменты $\vartheta_n^{(j)}$. Обозначим $\varsigma_{\vartheta_n^{(j)}}^{(j)} = \xi_n^{(j)}$ значение, которое функция $\varsigma_t^{(j)}$ получает в момент $\vartheta_n^{(j)}$. Последовательность $\xi_n^{(j)}$, которая служит оценкой частной производной функции $w(\mathbf{b})$ по j-ой компоненте вектора $\mathbf{b}$, задается рекуррентным соотношением

$$\xi_{n+1}^{(j)} = \frac{a_n c_n \xi_n^{(j)} + Z_{n+1}^{(j)}}{c_n}, \xi_0^{(j)} = 0.$$

В этом соотношении:

$a_n = 1 - n^{-a}, 0 < a < 1$, ($a = const$ — параметр алгоритма);

$c_{n+1} = a_n c_n + 1, c_0 = 0$.

Аналогичным образом можно корректировать в процессе функционирования системы любую из компонент векторов $\mathbf{p}$ и $\mathbf{q}$. В качестве последовательности z_t можно выбирать объем выполненных заданий для любого пользователя или любой группы пользователей. Кроме того, в роли z_t может выступать последовательность одношаговых затрат, или же количество потерянных задач для отдельных или для всех потребителей. В этом случае адаптивный алгоритм будет призван оптимизировать соответственно затраты или отказы для выбранной группы потребителей.

В следующем параграфе приводятся некоторые результаты компьютерного моделирования, проведенные с описанной в §1 моделью. Для управления формированием пакетов заданий используются адаптивные стратегии, которые сравниваются с некоторыми статическими неадаптивными стратегиями. Краткое описание программной системы дано в §5.

З а м е ч а н и е. Компьютерная реализация в основном совпадает с описанием модели, приведенном выше. Небольшое отличие, не меняющее существа дела, имеется только в способе дискретизации выбора размера пакета. Кроме того, к параметрам потребителей добавлен еще один стоимостной параметр — штраф за задание, которое превысило максимальное время пребывания в системе, но осталось невыполненным.

§4. Вычислительные эксперименты

Все имитационные эксперименты, описанные в этом параграфе, имели продолжительность 2000000 тактов.

Пример 1. Один ресурс, один потребитель.

Этот вариант системы — простейший по составу. Понятно, что в такой ситуации речь может идти только об управлении объемом посылаемых пакетов, и в проводившихся экспериментах были использованы различные стратегии выбора этого объема. Поэтому назначение этого примера — показать, оказывает ли размер пакета вообще какое-либо влияние на показатели качества системы.

Параметры потребителя:

α — интенсивность входного потока положим равной 10,

K — максимально количество задач в задании — 10,

V — максимальный объем задачи — 10,

T — максимальное время жизни задания — 10.

Параметры ресурса:

a — производительность положим равной 2,
d — максимальное время обслуживания задания — 100,
β — вероятность сбоя — 0.
c — стоимость единицы ресурса — 1.

Удаленность потребителя от ресурса была принята равной 1. Тариф за пересылку также составлял 1. Штраф за просроченное невыполненное задание равнялся 100.

Напомним, что согласно выбранной модели пакет формируется из одного задания. В этот пакет можно включить либо все задачи из задания, либо некоторую часть из них, если задач в задании больше чем одна. Управление размером пакета заключается в указании количества задач, которые следует включить в пакет. Это достигается заданием и использованием вектора $\mathbf{q}$ (см. §3). Ниже описаны результаты 9 экспериментов, далее они упоминаются также просто по номерам от 1 до 9.

В экспериментах 1-6 использовались статические стратегии, причем стратегии 1-5 являются детерминированными. Последнее означает, что вектор $\mathbf{q}$, который остается неизменным на протяжении всего эксперимента, имеет одну из компонент равную 1, а остальные — 0.

Векторы $\mathbf{q}$ в статических стратегиях 1-4 были выбраны так, что размер пакетов увеличивается с увеличением номера стратегии. Стратегия 1 предписывает выбор наименьшего размера пакета, ..., стратегия 4 предписывает выбор наибольшего размера пакета.

Статическая стратегия в эксперименте 5 предписывает включать в задание все задачи из задания.

Статическая рандомизированная стратегия 6 означает, что количество задач, включаемых в пакет, выбирается, исходя из равномерного распределения.

В экспериментах 7-9 применялись адаптивные стратегии, при которых вектор $\mathbf{q}$ корректируется в процессе имитации

функционирования системы

Были рассмотрены адаптивные стратегии с различными целевыми функциями. В случае 7 — это производительность, в случае 8 — затраты, а в случае 9 — отказы.

Основные численные результаты представлены в табл. 2, в ней три нижние строки относятся к адаптивным стратегиям. Из этой таблицы, прежде всего, следует сделать вывод о том, что управляемый параметр — размер посылаемых пакетов — оказывает заметное влияние на все основные характеристики качества.

Табл. 2

№ эксперимента	Производительн.	Затраты	Отказы, %
1	1,58	11,38	6,6
2	1,69	10,67	5,6
3	1,38	9,32	5,3
4	1,26	8,68	4,9
5	1,23	8,51	4,8
6	1,62	10,05	5,2
7	1,79	10,91	5,5
8	1,24	8,51	4,8
9	1,27	8,68	4,9

Рассмотрим вначале статические стратегии.

Наилучшая по производительности статическая стратегия соответствует случаю 2, когда в пакет включается в среднем немного меньше половины задач. Наименее эффективными среди статических стратегий для этого случая оказываются те, при которых в пакет включается все задание (5) или "почти все" задание (3,4). Относительно невыгодно также формировать совсем маленькие пакеты (1). Хороший результат дает стратегия равновероятного выбора размера пакета (6).

С точки зрения затрат и отказов выбор статической стра-

тегии иной. Наиболее эффективна по обоим этим показателям стратегия 5, а наименее эффективна стратегия 1 с минимальным размером пакетов.

Сравним адаптивные и статические стратегии.

Стратегия 7, максимизирующая производительность, дает равномерно наилучший результат по этому показателю среди всех испытывавшихся стратегий. На рис. 1 показаны графики производительности для стратегии 7, а также для лучшей среди статических стратегий — вариант 2. В то же время стратегия 7 не является лучшей ни по затратам, ни по отказам.

Изменяя целевую функцию, можно добиться того, чтобы адаптивная стратегия достигала наилучших результатов по показателям затрат (8) и отказов (9).

На рис. 2, 3 изображены сравнительные траектории процесса при использовании адаптивных стратегий, минимизирующих соответственно затраты и отказы в сравнении с лучшей по этим показателям статической стратегией 5.

В связи с рассмотренным примером следует обратить внимание на следующее. Даже в этом минимальном по размерности случае было не очевидно, что показатели качества окажутся чувствительными к регулировке размера пакета и, тем более, не ясно, как априори выбирать оптимизирующий (невырожденный) вектор $\mathbf{q}$, даже при наличии полной информации о параметрах системы. Подчеркнем еще раз, что адаптивные алгоритмы такой информацией "не располагали".

Пример 2. 5 ресурсов, 10 потребителей.

В этом примере система содержит максимум участников, которые позволяет существующая компьютерная реализация: 10 потребителей и 5 ресурсов. Большинство параметров системы было выбрано случайным образом.

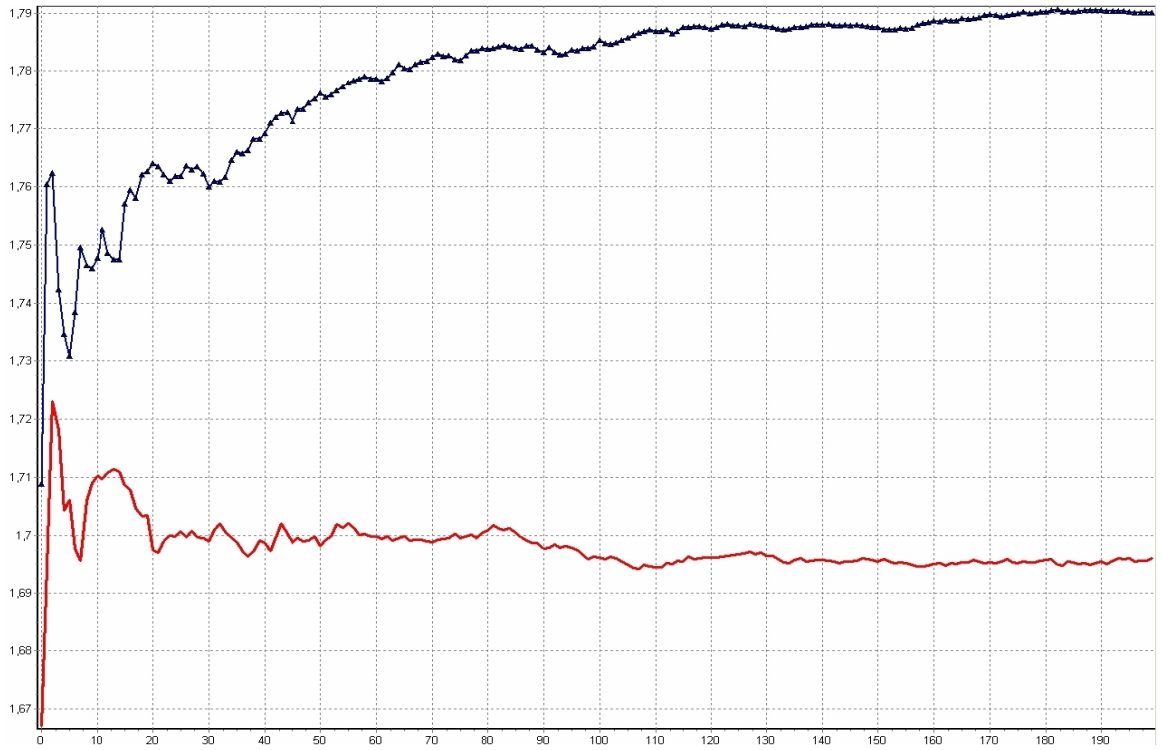

Рис. 1. (——) – эксперимент 2, (—△—) – эксперимент 7

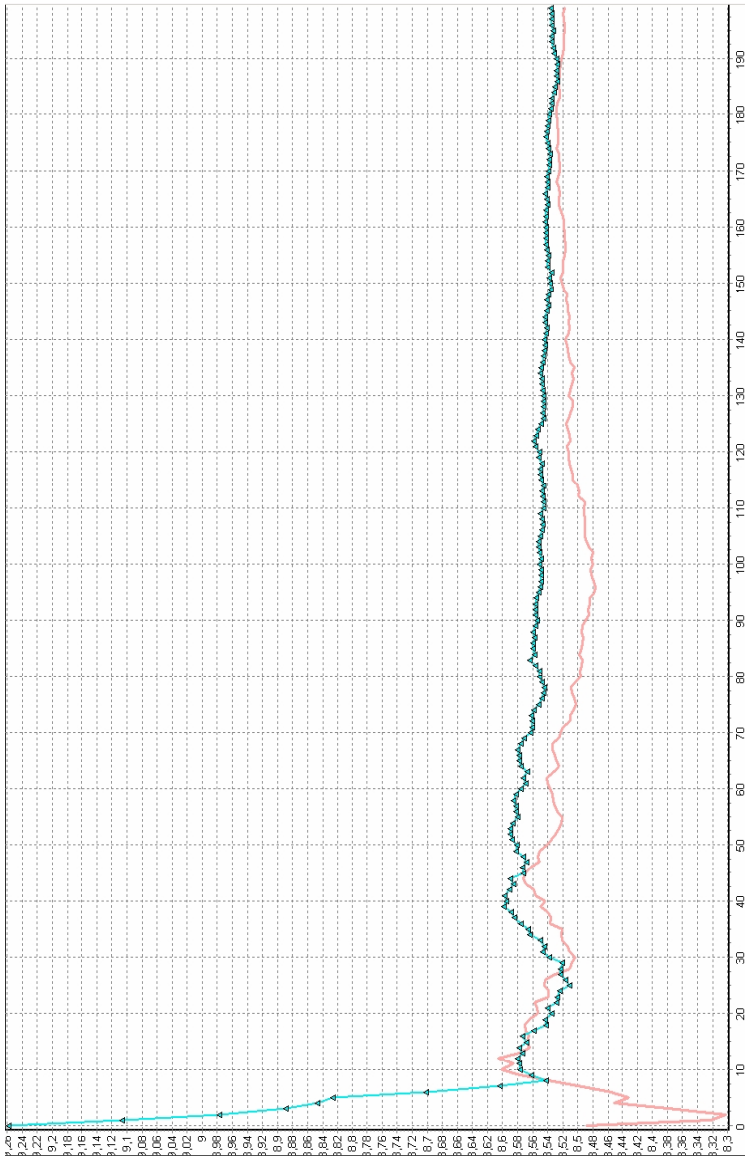

Рис. 2. (——) – эксперимент 5, (–△–) – эксперимент 8

Рис. 3. (——) – эксперимент 5, (–△–) – эксперимент 9

Параметры ресурсов представлены в табл. 3, параметры потребителей — в табл. 4 и сетевые параметры — в табл. 5. При этом для всех потребителей $T = 10, s = 100$, для всех ресурсов $d = 100$, и $k = C = 1$.

Табл. 3

Ресурсы	a	c	β
1	23	5	0,045
2	79	4	0,137
3	28	3	0,118
4	33	2	0,093
5	26	1	0,173

Табл. 4

Потребители	α	K	V
1	11,29	12	18
2	10,02	16	10
3	23,66	12	16
4	21,13	10	18
5	22,37	14	13
6	10,38	16	10
7	23,08	16	19
8	12,46	19	18
9	28,26	13	12
10	22,45	12	19

В данном случае рассматривается только управление выбором ресурсов, а управление размером пакетов отсутствует — в пакет всегда включаются все задачи из задания. Картина распределения параметров выглядит слишком запутанной, чтобы можно было априори предложить для всех потребителей какую-либо стратегию, которая выглядела бы предпочтительной с точки зрения производительности всей системы. Поэто-

му в качестве исходной была выбрана статическая стратегия, в которой каждый потребитель выбирал ресурсы равновероятно (эксперимент 1). Эта стратегия сравнивалась со стратегией, в которой все потребители действовали с помощью адаптивного алгоритма, причем в качестве целевой функции была взята производительность всей системы (эксперимент 2). Результаты сопоставления двух стратегий согласно рис. 4 свидетельствуют о преимуществе адаптивного поиска вектора **p**.

Табл. 5

Ресурсы	1	2	3	4	5
Потребитель 1	8	33	8	2	26
Потребитель 2	19	46	25	17	33
Потребитель 3	5	27	4	7	39
Потребитель 4	7	36	14	1	42
Потребитель 5	45	42	11	14	48
Потребитель 6	12	4	14	2	45
Потребитель 7	35	25	31	45	41
Потребитель 8	27	1	21	26	32
Потребитель 9	44	47	43	10	24
Потребитель 10	44	48	9	24	23

Следующие четыре эксперимента в этом примере касаются потребителя 7, который, как следует из табл. 2, порождает наибольший поток задач (так как произведение KV для этого потребителя наибольшее). В экспериментах 3-6 все потребители, кроме седьмого, неизменно выбирают ресурсы равновероятно, а для потребителя 7 рассмотрены четыре способа поведения.

В одном случае (эксперимент 3) потребитель 7 посылает все пакеты в ресурс 2 (статическая стратегия). Такой выбор обусловлен анализом параметров в табл. 4,5. В самом деле, ресурс 2 расположен ближе всего к потребителю 7 и имеет наибольшую производительность среди всех ресурсов.

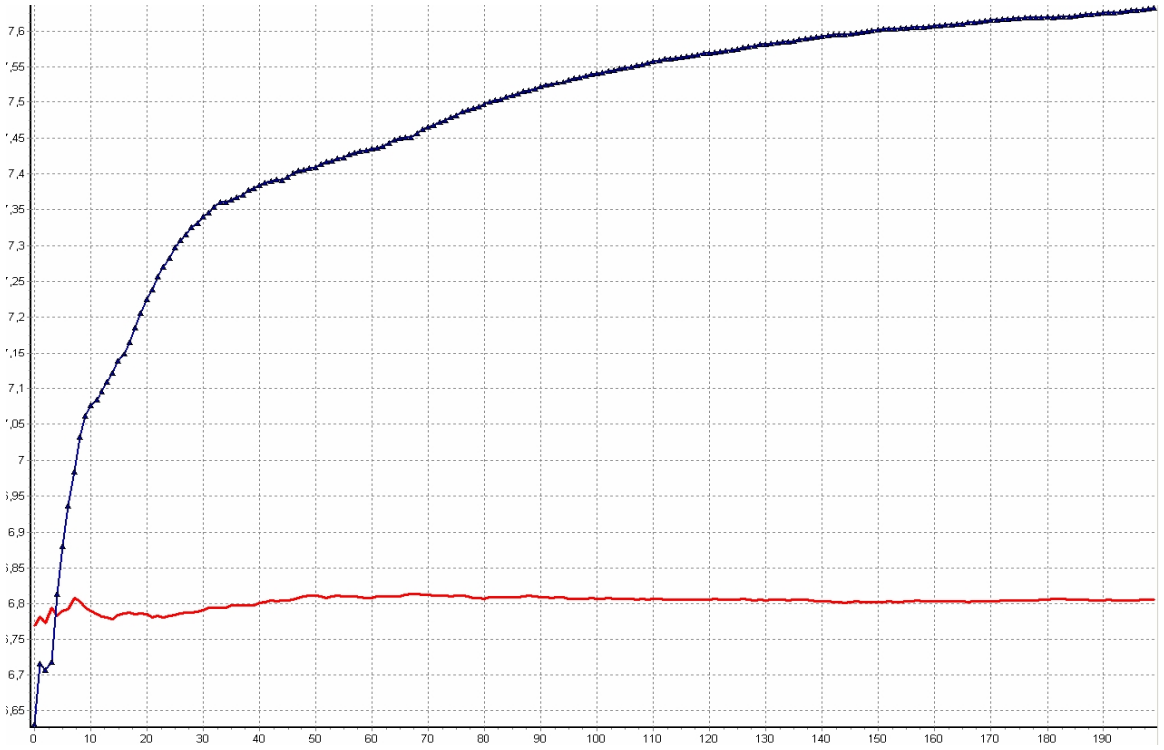

Рис. 4. (——) – эксперимент 1, (—△—) – эксперимент 2

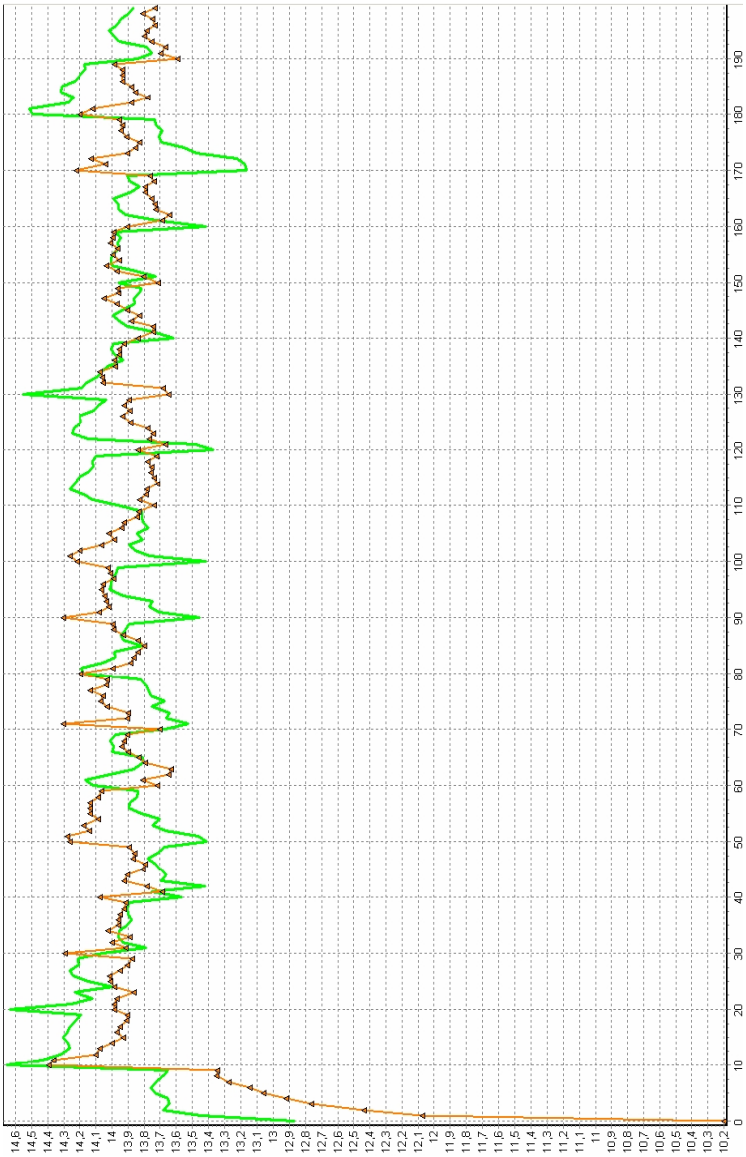

Рис. 5. (——) – эксперимент 3, (–△–) – эксперимент 4

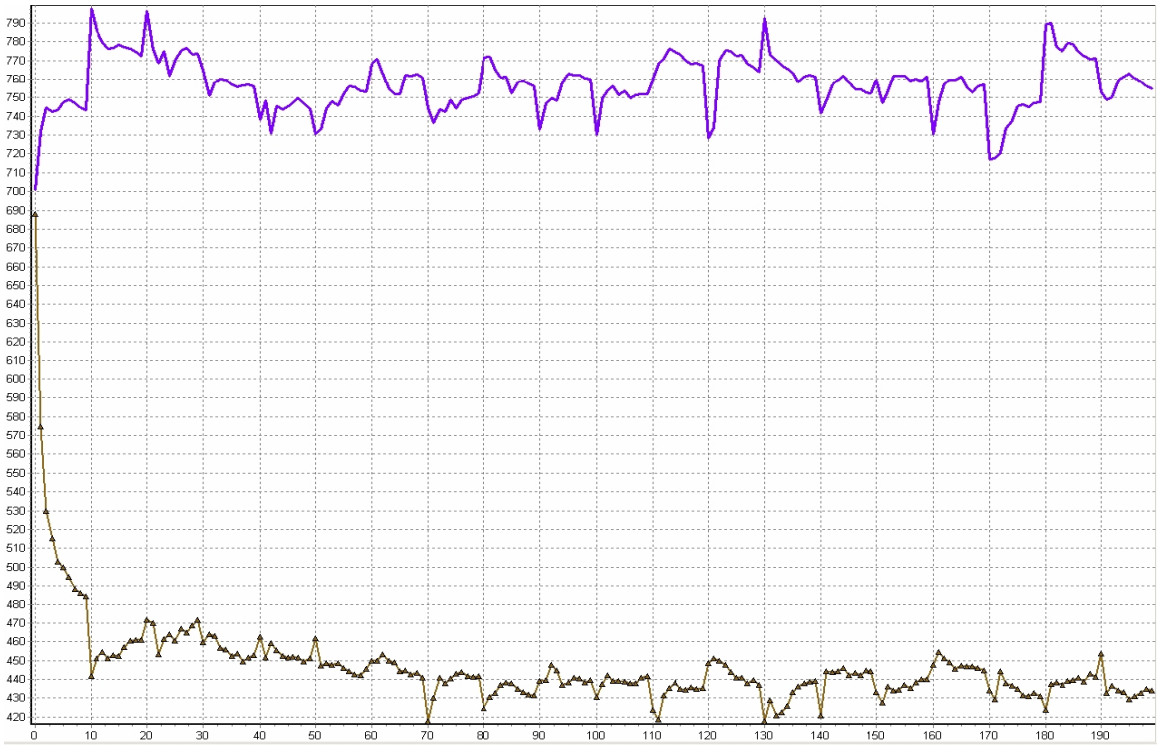

Рис. 6. (——) – эксперимент 3, (—△—) – эксперимент 5

Остальные варианты поведения потребителя 7 — адаптивные (эксперименты 4-6). Они отличаются целевыми функциями, в качестве которых были выбраны: собственная производительность (4), собственные затраты (5) и собственные отказы (6). С помощью рис. 5, 6 можно сравнить показатели качества функционирования потребителя 7 (производительность и затраты) для интуитивно предпочтительной статической стратегии 3 и адаптивных стратегий 4 и 5.

На рис. 5 изображены графики локальных производительностей потребителя 7 в экспериментах 3 и 4. Здесь локальная производительность определяется как средний объем выполненных задач, который подсчитывается в течение каждых 100000 тактов и обнуляется в конце каждого такого промежутка. Из рисунка видно, что производительность адаптивной стратегии по прошествии некоторого "периода адаптации" совпадает с производительностью статической стратегии 3. Это может служить подтверждением того, что стратегия 3 эффективна с точки зрения производительности.

На рис. 6 аналогичные графики представлены для локальных затрат. В этом случае картина иная: адаптивная стратегия находит вектор p, приносящий почти в два раза меньшие затраты. Это, скорей всего, связано с тем, что ресурс 2 является одним из самых дорогих, и, исходя из экономических соображений, посылать на него весь поток нецелесообразно. Средние отказы потребителя 7 в случае использования статической стратегии 3 и адаптивной стратегии 6 практически одинаковы, после окончания периода моделирования (2000000 тактов) они составили соответственно 6,0% и 6,1%.

Пример 3. Неоднородный случай. 2 ресурса, 5 потребителей.

Выше везде предполагалось, что параметры системы неизменны во времени, однако в этом примере предположим, что это не так. Исследуем, как ведут себя адаптивные алгоритмы,

если параметры ресурсов изменяются во времени.

Параметры потребителей отличаются значениями интенсивностей потоков заданий, которые заданы в табл. 6. Остальные параметры одинаковы для всех потребителей: $K = 10$, $V = 10, T = 10$.

Табл. 6

Потребители	1	2	3	4	5
α	10	15	20	25	30

Оба ресурса находятся на расстоянии 1 от всех потребителей и имеют равное, неизменное значение параметра $d = 100$. Производительность и вероятность сбоя изменяются в течение имитационного эксперимента так, как это указано в табл. 7. Таким образом, эксперимент разбивается на четыре промежутка однородности.

Табл. 7

Промежуток времени в тактах	от 0 до $5 \cdot 10^5$	от $5{\cdot}10^5$ до 10^6	от 10^6 до $1,5 \cdot 10^6$	от $1,5 \cdot 10^6$ до $2 \cdot 10^6$
Ресурс 1	$a = 20$ $\beta = 0$	$a = 25$ $\beta = 0$	$a = 10$ $\beta = 0$	$a = 40$ $\beta = 0$
Ресурс 2	$a = 30$ $\beta = 0,4$	$a = 25$ $\beta = 0$	$a = 40$ $\beta = 0$	$a = 10$ $\beta = 0$

На втором промежутке оба ресурса одинаковы, поэтому статическая стратегия равновероятного выбора ресурсов, по-видимому, является на этом участке оптимальной для всех потребителей. На первом промежутке меньшая производительность первого потребителя компенсируется ненадежностью второго ресурса, поэтому можно предположить, что и здесь та же статическая стратегия будет достаточно эффективна.

На последних промежутках времени ситуация явно несимметрична, причем в противоположные стороны. Однако если

предположить, что потребители могут не знать о состоянии ресурсов, то и здесь допустимо предложить равновероятную стратегию. Возьмем эту статическую стратегию за эталон, хотя заведомо не оптимальный (эксперимент 1), и сравним ее с адаптивными стратегиями. Были выбраны стратегии в которых целевая функция — производительность системы, при этом в одном случае — без управления размером пакетов (эксперимент 2), а во другом — с управлением (эксперимент 3).

Результаты экспериментов представлены на рис. 7-9.

На рис. 7 сравниваются производительности статической стратегии и адаптивной стратегии без управления размером пакетов. Последняя имеет почти везде преимущество, которое невелико на первых двух промежутках однородности (статическая стратегия здесь достаточно эффективна), но становится подавляющим на последних двух. В то же время в начале второго млн. тактов, когда адаптивный алгоритм выбирает неэффективные управления в поисках оптимального значения вектора $\mathbf{p}$, статическая стратегия дает большую производительность. Здесь появляется резкая ассиметрия ресурсов.

На рис. 8 приведены графики локальной производительности и хорошо видны участки, где происходит активное "обучение" (интервалы от 1000000 до 1200000 и от 1500000 до 1600000 тактов).

Рис. 9 дает еще одно свидетельство того, что управление размером пакетов является целесообразным. Адаптивное регулирование объема пакетов в дополнение к адаптивному выбору ресурсов дает стратегию, которая уже на всем промежутке моделирования превосходит статическую стратегию. Следует отметить, однако, что это преимущество возникло в данном случае исключительно за счет первого промежутка однородности. На отрезке после 500000 такта управление размером пакетов не приносит увеличения производительности.

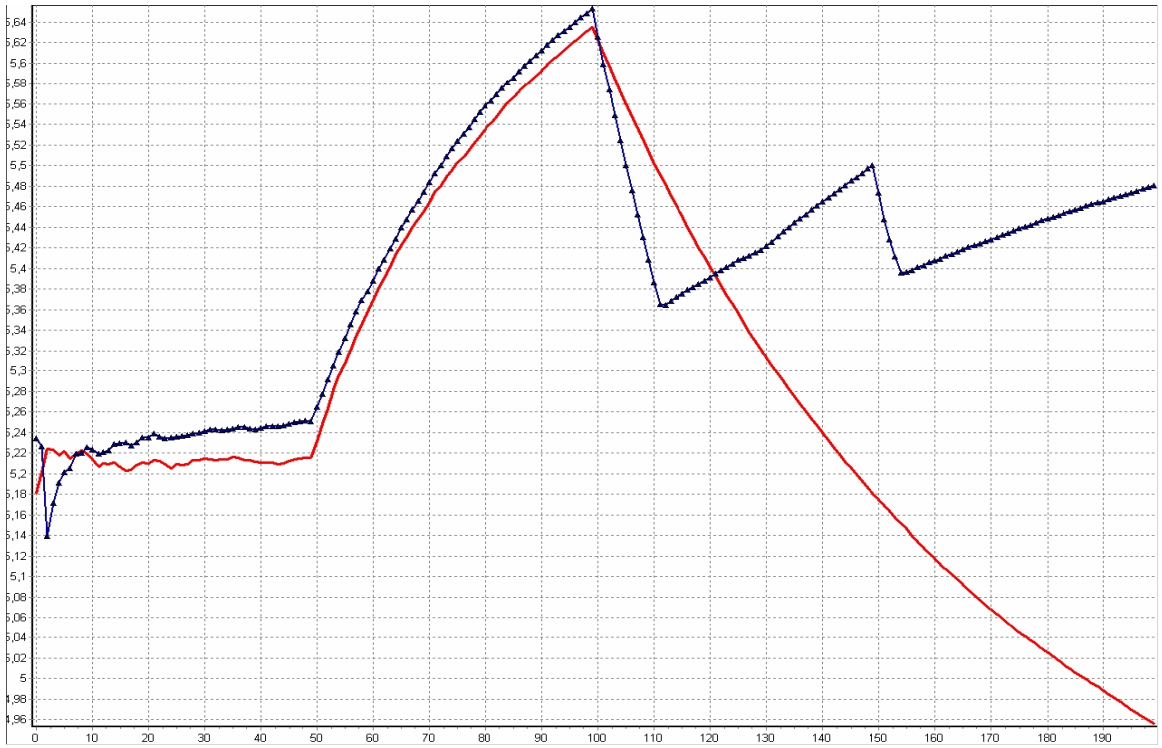

Рис. 7. (——) – эксперимент 1, (—△—) – эксперимент 2

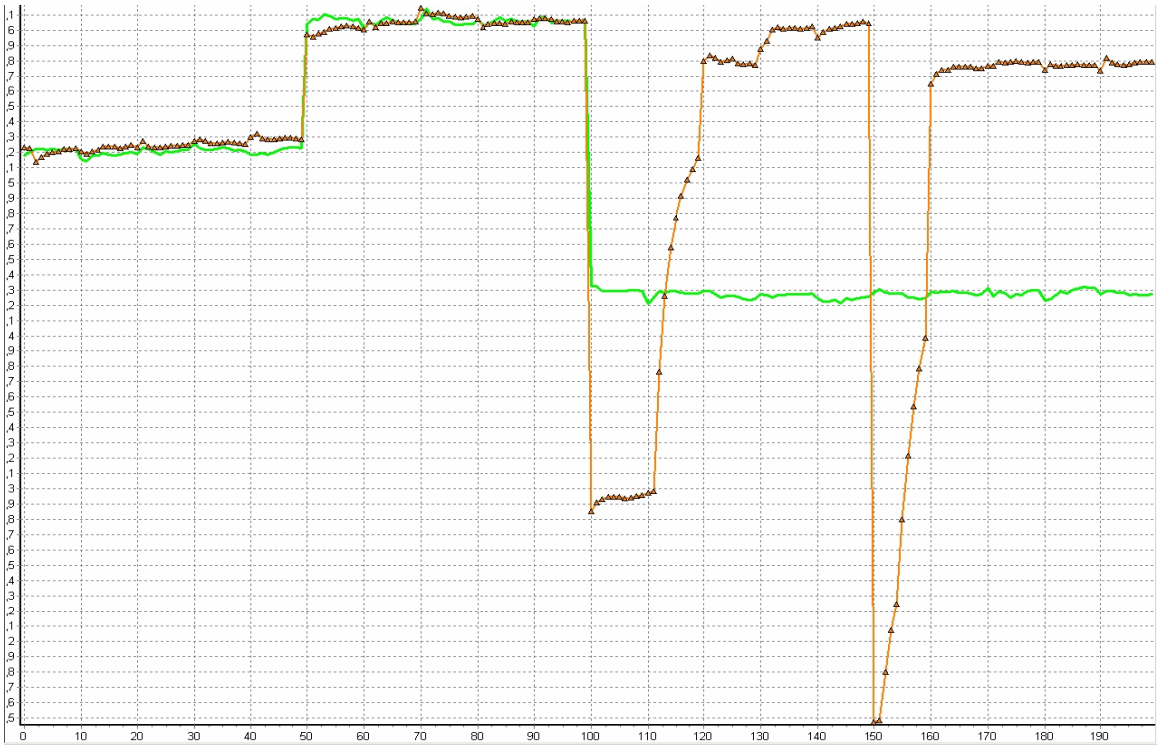

Рис. 8. (——) – эксперимент 1, (—△—) – эксперимент 2

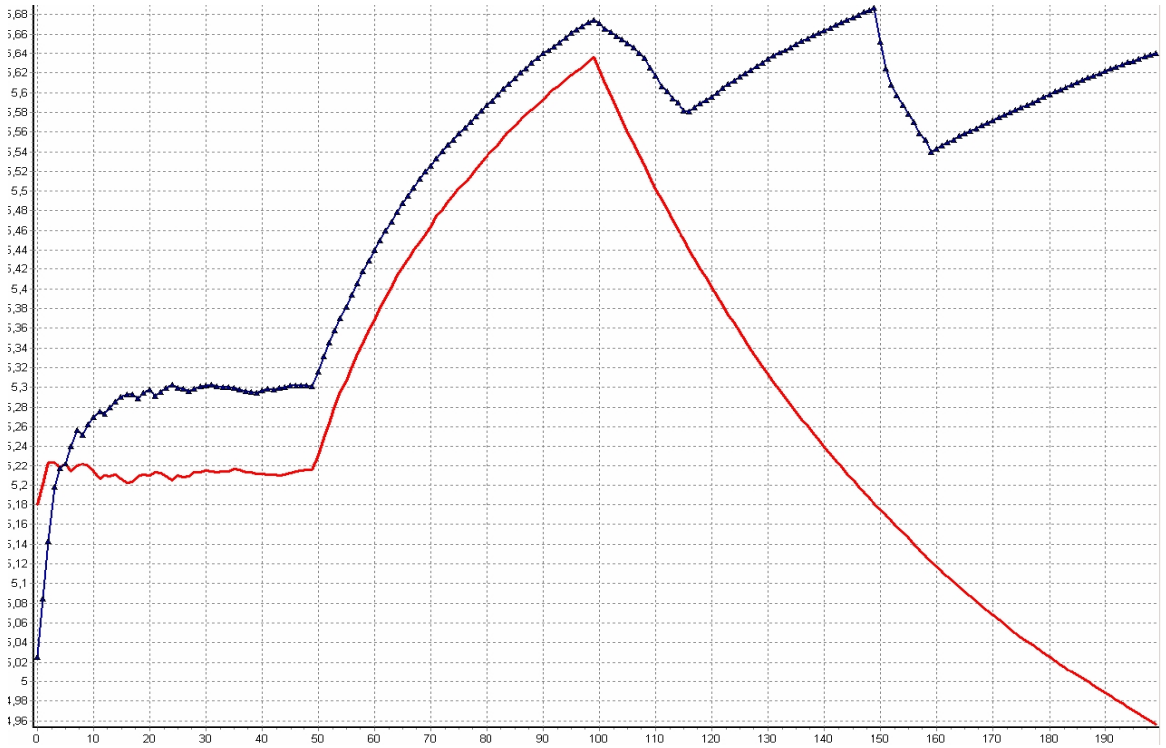

Рис. 9. (——) – эксперимент 1, (—△—) – эксперимент 3

§5. Компьютерная реализация модели

Описанная в настоящей работе модель коллективного взаимодействия удаленных потребителей и ресурсов реализована в программной системе. Она носит исследовательский характер и является инструментом для экспериментального изучения поведения системы и зависимости ее показателей качества от всех параметров, задающих модель. Программа позволяет задавать в модели стратегии управления назначением ресурсов и выбором объема пакетов. Наиболее важной особенностью программы является возможность проводить экспериментальное сравнение качества различных стратегий, как статических, так и адаптивных.

С помощью программы можно:

— создавать модель системы, состоящую из потребителей (от 1 до 10) и ресурсов (от 1 до 5), взаимодействующих через телекоммуникационную сеть;

— задавать и корректировать в широком диапазоне значения всех параметров модели, относящихся к потребителям, к ресурсам, а также сетевых параметров;

— выбирать режим функционирования системы: под управлением статической стратегии назначения ресурсов или под управлением адаптивной стратегии назначения ресурсов;

— подключать дополнительно функцию статического или адаптивного управления выбором размера пакетов;

— осуществлять настройку некоторых параметров стратегий управления;

— проводить эксперименты, имитируя поведение системы для определенной конфигурации и заданном способе управления;

— осуществлять визуальное наблюдение за характеристиками системы в ходе эксперимента;

— просматривать результаты экспериментов в формате данных и в графическом формате;

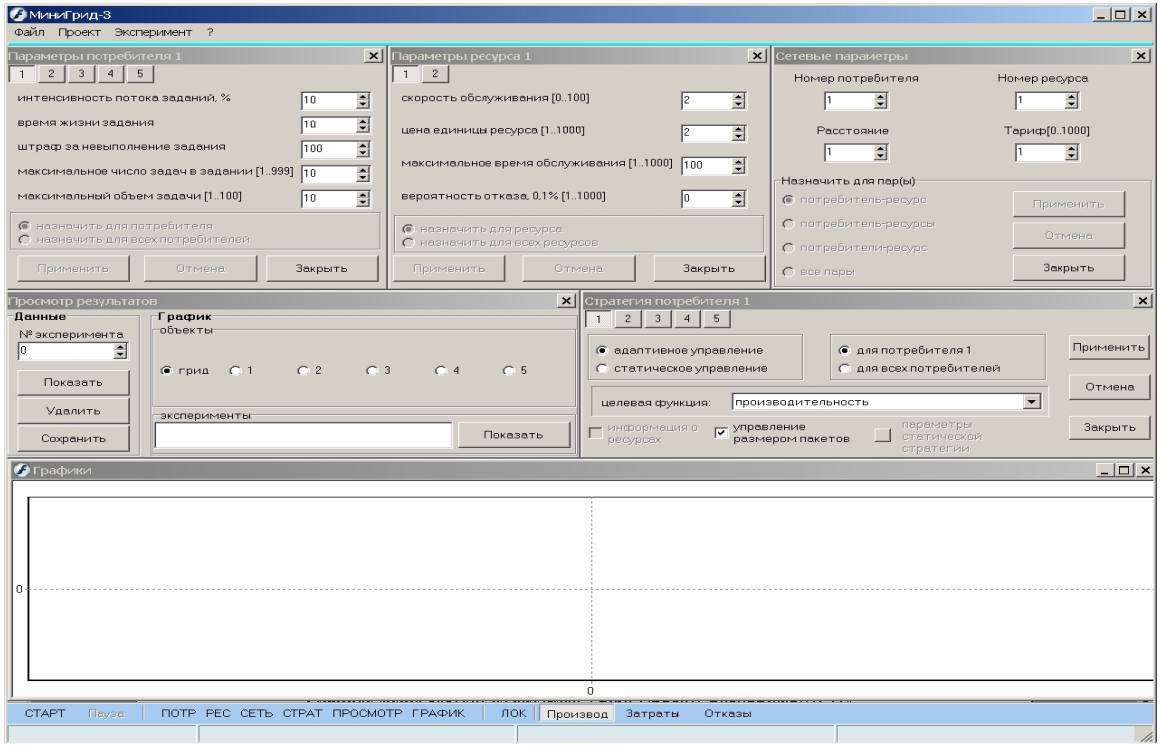
МиниГрид-3
Файл Проект Эксперимент ?
Параметры потребителя 1
1 2 3 4 5
интенсивность потока заданий, %
время жизни задания
штраф за невыполнение задания
максимальное число задач в задании [1..999]
максимальный объем задачи [1..100]
назначить для потребителя
назначить для всех потребителей
Применить Отмена Закрыть
Параметры ресурса 1
1 2
скорость обслуживания [0..100]
цена единицы ресурса [1..1000]
максимальное время обслуживания [1..1000]
вероятность отказа, 0,1% [1..1000]
назначить для ресурса
назначить для всех ресурсов
Применить Отмена Закрыть
Сетевые параметры
Номер потребителя
Номер ресурса
Расстояние
Тариф[0..1000]
Назначить для пар(ы)
потребитель-ресурс
потребитель-ресурсы
потребители-ресурс
все пары
Применить Отмена Закрыть
Просмотр результатов
Данные
№ эксперимента
Показать Удалить Сохранить
График
объекты
грид 1 2 3 4 5
эксперименты
Показать
Стратегия потребителя 1
1 2 3 4 5
адаптивное управление
статическое управление
для потребителя 1
для всех потребителей
целевая функция: производительность
информация о ресурсах
управление размером пакетов
параметры статической стратегии
Применить Отмена Закрыть
Графики
0
0
СТАРТ Пауза ПОТР РЕС СЕТЬ СТРАТ ПРОСМОТР ГРАФИК ЛОК Производ Затраты Отказы

Рис. 10

— сохранять на диск текущую конфигурацию модели, а также загружать в программу ранее созданные варианты модели;
— сохранять на диск результаты экспериментов.

Остановимся более подробно на интерфейсе программы. После запуска программы на экране появляются (рис. 10):
— основная форма с главным меню, панелью инструментов для управления окнами, ходом эксперимента и просмотром результатов, а также со строкой состояния для отображения показателей качества системы в ходе имитационного эксперимента;
— шесть независимых окон для редактирования параметров и наблюдения за ходом экспериментов и их результатами.

В процессе работы пользователь может видеть:
— модальное окно для создания модели с новым количеством потребителей и ресурсов;

— модальное окно для выбора продолжительности эксперимента;
— модальное окно для редактирования параметров статических стратегий;
— окна с результатами экспериментов.

Главное меню состоит из разделов Файл, Проект, Эксперимент.

Раздел Файл содержит команды Открыть, Создать, Сохранить, Сохранить результаты, Выход, раздел Проект — Потребители, Ресурсы, Сеть, раздел Эксперимент — Продолжительность, Стратегия, График, Просмотр, Старт.

Программа содержит следующие **Окна редактирования параметров**.

Окно "Параметры потребителя"(рис. 11) предназначено для контроля и изменения параметров каждого потребителя.

Окно "Параметры ресурса"(рис. 12) предназначено для контроля и изменения параметров каждого ресурса.

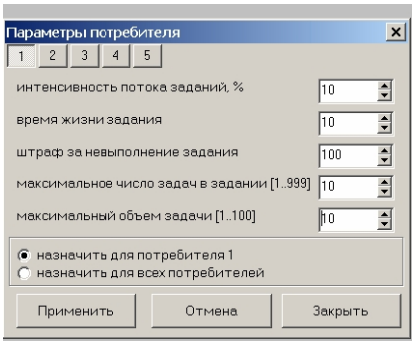

Рис. 11

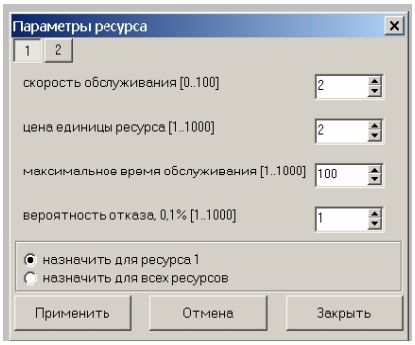

Рис. 12

Окно "Сетевые параметры"(рис. 13) предназначено для задания тарифа и расстояний между парами потребитель—ресурс.

Окно "Стратегия потребителей"предназначено для выбора способа управления (статический или адаптивный), для задания и отключения режима управления размером пакетов, а также для настройки некоторых параметров стратегий управления. В случае если выбрана адаптивная стратегия, можно

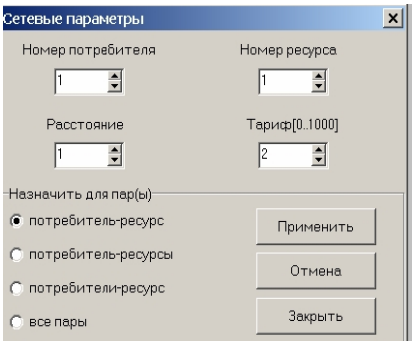

Рис. 13

дополнительно выбрать и задать цель управления (рис. 14).

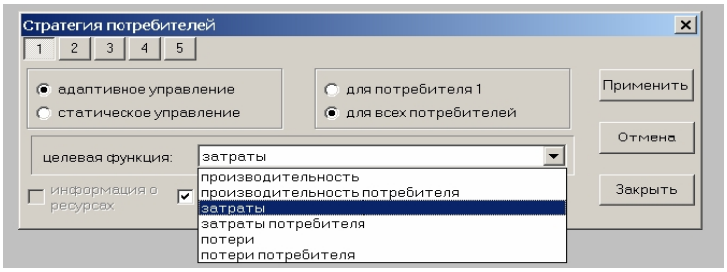

Рис. 14

Если выбрана статическая стратегия управления, можно дополнительно отредактировать ее параметры в модальном окне "Стратегия"(рис. 15), которое отрывается после нажатия кнопки "Параметры статической стратегии".

В окне указаны компоненты векторов, отвечающих за выбор ресурса и за управление потоком (последний только в случае, если выставлен флаг "управление размером потока"). Для

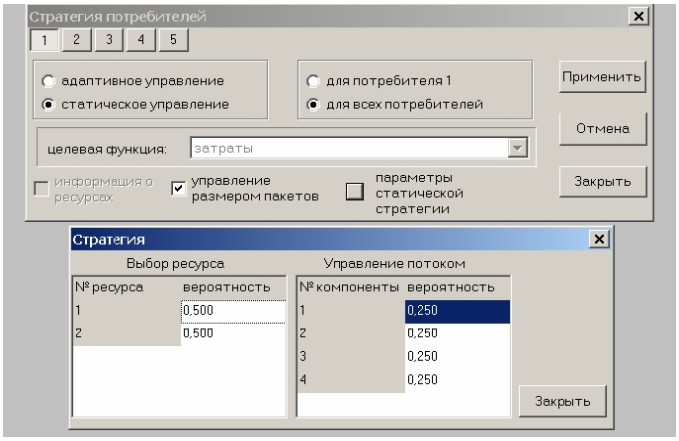

Рис. 15

задания новых значений этих векторов достаточно указать весовые коэффициенты компонент, а нормировка произойдет автоматически. Все окна могут быть скрыты или вновь вызваны с помощью команд главного меню или с помощью панели инструментов в нижней части основной формы.

Для проведения эксперимента и наблюдения за его ходом прежде всего необходимо посредством специального модального окна, которое вызывается с помощью команды главного меню Продолжительность, установить его продолжительность.

После того, как выбраны и отредактированы все параметры, с помощью команды Старт главного меню или с помощью одноименной кнопки на панели инструментов можно осуществить запуск имитационного эксперимента.

Во время всего эксперимента на панели состояния выводится количество выполненных тактов моделирования и значения трех основных характеристик процесса: производительности системы, затрат и отказов. Кроме того, в специальном

окне можно видеть графическое отображение динамики любой из указанных характеристик, либо их локальных (подсчитываемых в течение каждых 100000 тактов) вариантов. Переключение между типом графика осуществляется с помощью кнопок на панели инструментов, в частности, режим просмотра локальных характеристик включается и отключается кнопкой ЛОК.

Процесс моделирования можно приостановить с помощью кнопки Пауза, а затем вновь продолжить.

Процесс моделирования можно завершить досрочно с помощью команды Стоп главного меню или с помощью одноименной кнопки на панели инструментов. Заметим, что команды Пауза и Стоп имеют некоторое запаздывание: после их задания процесс (при)останавливается только в ближайший, кратный 10^5 тактов момент.

В процессе эксперимента доступны описанные выше окна редактирования параметров. Это дает, в частности, возможность моделировать поведение системы в неоднородной среде.

Для работы с результатами экспериментов имеется окно "Просмотр результатов". Вновь проведенный, закончившийся эксперимент становится доступным в этом окне под очередным порядковым номером (рис. 16).

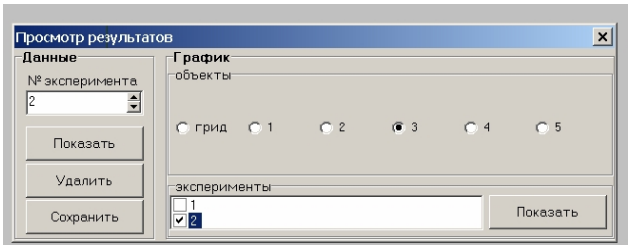

Рис. 16

Просмотр результатов можно осуществить в двух форматах: в виде текста (панель "Данные") и виде графиков (панель "Графики"). Для вывода на экран текста с данными следует на панели "Данные" выбрать нужный номер, затем нажать кнопку "Показать". Появится окно отчета (рис. 17).

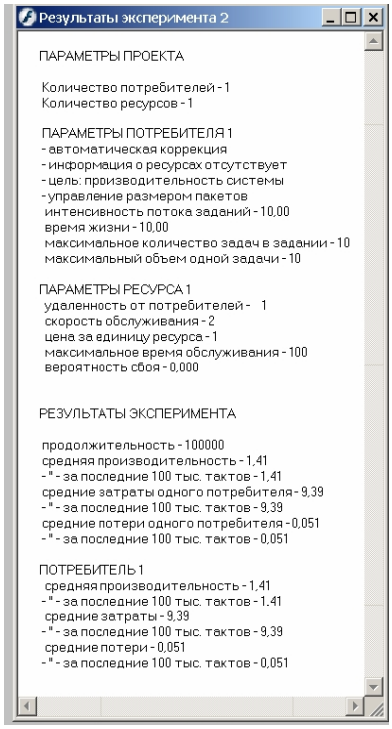

Рис. 17

Для просмотра графиков нужно выполнить следующие действия:

на панели "Графики" в окне "эксперименты" выделить один или два номера эксперимента;

на той же панели переключатель "объекты" установить в положение, отвечающее либо за всю систему, либо за одного из потребителей;

на панели инструментов выбрать тип графика с помощью кнопок ЛОК, Производ, Затраты, Отказы;

нажать кнопку "Показать" на панели "Графики".

После этого в специальном окне появляется график изменения во времени выбранного показателя для заданного объекта. Подобным образом были получены графики с рис. 1-9.

Сохранение результатов эксперимента осуществляется с помощью панели "Данные". Здесь нужно выделить номер эксперимента и нажать кнопку "Сохранить".

Удаление результатов эксперимента также осуществляется с помощью панели "Данные". Выбираем номер эксперимента и нажимаем кнопку "Удалить". При этом результаты выбранного эксперимента удаляются, а следующие за ним эксперименты получают номера на единицу меньше прежних.

Заключение

Данная работа посвящена проблеме комплексного управления потоками заданий и ресурсами в больших распределенных компьютерных системах (гридах). Проведенный в работе анализ литературы по этому вопросу свидетельствует, что проблема актуальна, но далека от окончательного решения и является предметом активных исследований. Для ее решения применяются самые разнообразные подходы и методы. Широко используется математическое моделирование, причем наиболее интенсивный поиск ведется с помощью средств теории массового обслуживания и методов оптимизации. Большое внимание уделяется разработке эвристических алгоритмов и развитию "экономических" методов. Неизменным и важным инструментом исследований остается имитационное моделирование. Существенное значение имеет создание новой архитектуры системы управления ресурсами, которая все чаще имеет многоуровневый характер.

В работе описана оригинальная модель оперативного управления размещением заданий в локальной подсистеме грида, которая отражает принципиальные особенности информационных процессов в современных системах коллективного использования распределенных вычислительных ресурсов. Предложенная модель является имитационной и опирается на подробное алгоритмическое описание процесса коллективного взаимодействия через телекоммуникационную сеть взаимно удаленных ресурсов и пользователей. Управление потоками заданий включает в себя назначение ресурсов для их выполнения и выбор объема текущих пакетов заданий. Цели управления формализуются в терминах производительности системы, количества отказов в ней, а также стоимостных показателей. Проблема сводится к задаче оптимального управления марковскими последовательностями. Для решения последней используются

адаптивные алгоритмы, оптимизирующие предельные средние характеристики.

Предложенные адаптивные стратегии слабо зависят от структуры и особенностей конкретной модели. Поэтому в случае изменения деталей, например, в сторону большего отражения особенностей конкретной системы, работоспособность алгоритмов сохранится.

Имитационная модель реализована в виде программной системы, которая позволяет в широком диапазоне настраивать параметры модели, включая стратегии управления, и проводить вычислительные эксперименты. Для исследования модели была проведена серия численных экспериментов для различных конфигураций систем и разных сценариев организации потоков заданий. Их описание и соответствующие графики можно найти в §4 раздела 2. По результатам экспериментов был проведен анализ свойств различных стратегий управления, который показал высокую эффективность разработанных адаптивных алгоритмов.

Таким образом, численный эксперимент показал, что построенная в работе информационная технология является полезным инструментом для экспериментального исследования свойств системы коллективного использования распределенных вычислительных ресурсов.

Литература

1. Foster I., Kesselman C., Nick J. et al. The physiology of the grid: An open grid services architecture for distributed systems integration. // http://www.globus.org/research/papers.html.

2. Коновалов М. Г. Адаптивное управление процессом размещения заданий в модели коллективного использования распределенных вычислительных ресурсов // Матер. 2-й междунар. конф. "Распределенные вычисления и Грид-технологии в науке и образовании". Дубна, 2006. С. 124-127.

3. Антонов С. В., Душин Ю. А., Коновалов М. Г. и др. Разработка математических моделей и методов распределения заданий в системе распределённых вычислений // Системы и средства информатики. М.: Наука. 2006. С. 32-45.

4. Gaeta M., Konovalov M., Shorgin S. Development of mathematical models and methods of task distribution in distributed computing system // Reliability: Theory & Applications 2006. V. 1. N. 4. P. 16-21.

5. Foster I., Kesselman C., Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. // International J. of Supercomputer Applications. 2001. V. 15. N. 3. P. 200-222.

6. Baker M., Buyya R., Laforenza D. Grids and Grid Technologies for Wide-Area Distributed Computing // Software: Practice and Experience. 2002. V. 32. N. 15. P. 1437-1466.

7. Berman F., Fox G., Hey T. Grid Computing: Making the

Global Infrastructure a Reality. Chichester: John Wiley & Sons, 2003.

8. Foster I., Kesselman C., Nick J. et al. Grid Services for Distributed System Integration // IEEE Computer. 2002. V. 35. N. 6. P. 37-46.

9. Sabin G., Sahasrabudhe V., Sadayappan P. On fairness in distributed job scheduling across multiple sites // IEEE INT. Conf. on Cluster Computing. IEEE, 2004. P. 35-44.

10. Gentzsh W. Sun grid engine: Towards creating a compute power grid // Proc. 1st IEEE/ACM Int. Symp. on Cluster Computing and the Grid. IEEE, 2002. P. 35-36.

11. Belalem G., Bouhraoua F. Dynamic strategy of placement of the replicas in data grid // Lecture Notes in Computer Science. 2007. V. 4671. P. 496-506.

12. Cho S., Lee M., In J. et al. Policy based scheduling for resource allocation on grid // Lecture Notes in Computer Science. 2007. V. 4537. P. 229-234.

13. Iosup A., Jan M., Sonmez O. et al.The characteristics and performance of groups of jobs in grids // Lecture Notes in Computer Science. 2007. V. 4641. P. 382-393.

14. Ishii R. P., De Mello R. F., Yang L.T. A complex network-based approach for job scheduling in grid environments // Lecture Notes in Computer Science. 2007. V. 4782. P. 204-215.

15. Krasnotcshekov V., Vakhitov A. Adaptive scheduling and esource assessment in grid // Lecture Notes in Computer Science. 2007. V. 4671. P. 240-244.

16. Lewis R., Torczon V. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds // SIAM J. on Optimization. 2002. V. 12. N. 4. P. 1075-1089.

17. Li B., Zhao D. Online algorithms for single machine schedulers to support advance reservations from grid jobs // Lecture Notes in Computer Science. 2007. V. 4782. P. 239-248.

18. Nou R., Kounev S., Torres J. Building online performance models of grid middleware with fine-grained load-balancing: a globus toolkit case study // Lecture Notes in Computer Science. 2007. V. 4748. P. 125-140.

19. Quetier B., Cappello F. A survey of Grid research tools: simulators, emulators and real life platforms // http://sab1.sscc.ru/imacs2005/papers/T1-I-58-1069.pdf.

20. Snavely A., Chun G., Casanova H. et al. Benchmarks for for Grid Computing: A Review of Ongoing Efforts and Future Directions // ACM Sigmetrics Performance Evaluation Review. 2003. V. 30. N. 4. P. 27-32.

21. Song E., Jeon Y., Han S.et al. Hierarchical and Dynamic Information Manage-ment Framework on Grid Computing // Lecture Notes in Computer Science. 2006. V. 4096. P. 151-161.

22. Mohammadi A., Akl S. Scheduling Algorithms for Real-Time Systems. Technical Report N. 2005-499. Queen's University Kingston, Ontario, 2005.

23. Ousterhout J. K. Scheduling techniques for concurrent systems // Proc. 3rd IEEE Int. Conf. on Distributed Computer Systems. IEEE Comp. Sc. Press, 1982. P. 22-30.

24. Paranhos D., Cirne W., Brasileiro F. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids // Proc. Int. Conf. Parallel and Distributed Computing, Euro-Par. 2003. P. 169-80.

25. Parra-Hernandez R., Vanderster D., Dimopoulos N. J. Resource management and knapsack formulations on the grid // Proc. 5th IEEE/ACM Int. Workshop on Grid Computing. IEEE Comp. Sc. Press, 2004. P. 94-101.

26. Frey J., Tannenbaum T., Livny M. et al. Condorg: A computation management agent for multi-institutional grids // Cluster Computing. 2002. V. 5. Iss. 3. P. 237-246.

27. Vadhiyar S.S., Dongarra J.J. A Metascheduler for the Grid // Proc. of 11-th IEEE Symp. High Performance Distributed Computing. IEEE Comp. Sc. Press, 2002. P. 343-351.

28. Li C., Li L. Utility-based QoS optimisation strategy for multi-criteria scheduling on the grid // J. of parallel and distributed computing. 2007. V. 67. N. 2. P. 142-153.

29. Salmani V., Ensafi R., Khatib-Astaneh N. et. al. A Fuzzy-

Based Multi-criteria Scheduler for Uniform Multiprocessor Real-Time Systems // Proc. 10th Int. Conf. on Information Technology. IEEE Comp. Sc. Press, 2007. P. 179-184.

30. Salmani V., Naghibzadeh M., Kahani M. et al. Multi-criteria Scheduling of Soft Real-time Tasks on Uniform Multiprocessors Using Fuzzy Inference // Advances and Innovations in Syst. Comp. Sciences and Software Engineering. Springer, 2007. P. 439-444.

31. Baraglia R., Klusaček D., Rudova H. et al. Comparison Of Multi-Criteria Scheduling Techniques // Grid Computing. Springer, 2008. P. 173-184.

32. Song S., Hwang K., Kwok Y.-K. Trusted grid computing with security binding and trust integration // J. of Grid Computing. 2005. V. 3. P. 53-73.

33. Yagoubi B., Slimani Y. Dynamic Load Balancing Strategy for Grid Computing // Transactions on Engineering, Computingand Technology. 2006. N. 13. P. 260-265.

34. Elmroth E., Tordsson J. An Interoperable, Standards-based Grid Resource Broker and Job Submission Service // 1st Int. Conf. on e-Science and Grid Computing. IEEE Comp. Sc. Press, 2005. P. 212 - 220.

35. Frumkin M., Wijngaart R. NAS Grid Benchmarks: A Tool or Grid Space Exploration // Cluster Computing. 2002. V. 5. Iss. 3. P. 247 - 255.

36. Herrera J., Huedo E., Montero R.S. et al. Loosely-coupled

Scheduling in Computational Grids // Proc. 20th IEEE Int. Parallel & Distributed Processing Symp. IEEE Comp. Sc. Press, 2006. P. 6.

37. Montero R.S., Huedo E., Llorente I.M. Benchmarking of high throughput computing applications on Grids // Parallel Computing. 2006. V. 32. Iss. 4. P. 267-279.

38. Phatanapherom S., Uthayopas P., Kachitvichyanukul V. Dynamic scheduling II: fast simulation model for grid scheduling using HyperSim // Proc. 35th conf. on Winter simulation. New Orleans, 2003. P. 1494-1500.

39. Zhou J., Yu K., Chou C. et al. Dynamic Resource Broker and Fuzzy Logic Based Scheduling Algorithm in Grid Environment // Proc. Int. Conf. on Adaptive and Natural Computing Algorithms. ICANNGA, 2007. P. 604–613.

40. Poddar H., Singh J., Sekhon G.S. Dynamic Scheduling for Hard Real-Time Multiprocessor Systems // Proc. of Natio-National Conf. on Challenges & Opportunities in Inf. Technology. Mandi Gobindgarh, 2007. P. 76-80.

41. Giersh A., Rober, Y., Vivien F. Scheduling tasks sharing files on heterogeneous master-slave platforms // Proc. 12th Euromicro Workshop in Par., Dist. and Network-based. IEEE Comp. Sc. Press, 2004. P. 364-371.

42. Senger H., Hruschka E.R., Silva F.A.B. et al. In-hambu: Data Mining Using Idle Cycles in Clusters of PCs // Proc. Int. Conf. on Network and Parallel Computing. Lecture Notes in Computer Science. 2004. V. 3222. P. 213-220.

43. Silva F.A.B., Carvalho S., Hruschka E.R. A Scheduling Algorithm for Running Bag-of-Tasks Data Mining Application on the Grid // Lecture Notes in Computer Science. 2004. V. 3419. P. 254-262.

44. Silva F.A.B., Carvalho S., Senger H. et al. Running Data Mining Applications on the Grid: a Bag-of-Tasks Approach // Int. Conf. on Computational Science and its Applications. Lecture Notes in Computer Science. 2004. V. 3044. P. 168-177.

45. Slegers J., Mitrani I., Thomas N. Optimal dynamic server allocation in systems with on/off sources // Lecture Notes in Computer Science. 2007. V. 4748. P. 186-199.

46. Marchal L., Yang Y., Casanova H. et al. Steady-state scheduling of multiple divisible load applications on wide-area distributed computing platforms // Int. J. of High Performance Comp. Appl. 2006. V. 20. N. 3. P. 365-381.

47. Banino C., Beaumont O., Carter L. et al. Scheduling strategies for master-slave tasking on heterogeneous processor platforms // IEEE Trans. Parallel Distributed Systems. 2004. V. 15. N. 4. P. 319-330.

48. Beaumont O., Casanova H., Legrand A. et al. Scheduling divisible loads for star and tree networks: main results and open problems // IEEE Trans. Parallel and Distributed Systems. 2005. V. 16 N. 3. P. 207-218.

49. Vanderster D. C., Dimopoulos N. J., Sobie R. J. Meta-sche-

duling multiple resource types using the MMKP // Proc. 7th IEEE / ACM Int. Conf. on Grid Computing. IEEE Comp. Sc. Press, 2006. P. 231-237.

50. Cappello F., Caron E., Dayde M. et al. Grid'5000: a large scale, reconfigurable, controlable and monitorable Grid platform // Proc. 6th IEEE/ACM Int. Workshop on Grid Computing. EEE Comp. Sc. Press, 2006. P. 99–106.

51. The Distributed ASCI Supercomputer (DAS) // http: //www.cs.vu.nl/das3

52. Bucur A. I. D., Epema D. H. J. The Performance of Processor Co-Allocation in Multi-cluster Systems // Proc. 3rd IEEE / ACM Int. Symp. on Cluster Computing and the GRID. IEEE Comp. Sc. Press, 2003. P. 302-309.

53. Ernemann C., Hamscher V., Schwiegelshohn U. et al. On Advantages of Grid Computing for Parallel Job Scheduling // Proc. 2nd IEEE / ACM Int. Symp. Cluster Computing and the GRID. IEEE Comp. Sc. Press, 2002. P. 39-46.

54. Mohamed H.H., Epema D. Experiences with the koala Co-Allocating Scheduler in Multiclusters // Proc. 5th IEEE / ACM Int. Symp. on Cluster Computing and the GRID. IEEE Comp. Sc. Press, 2005. V. 2. P. 784-791.

55. Mohamed H.H., Epema D. The design and implementation of the KOALA co-allocating grid scheduler // Lecture Notes in Computer Science. 2005. V. 3470. P. 640-650.

56. Ballier A., Caron E., Epema D. et al. Simulating grid sche-

dulers with deadlines and co-allocation // Research report [inria-00077051 — version 1]. Laboratoire de l'Informatique, du Parallelisme, 2006. http://www.ens-lyon.fr/LIP

57. Wolski R., Brevik J., Plank J.S. et al. Grid Resource Allocation and Control Using Computational Economics // Grid Computing: Making the Global Infrastructure a Reality. Chichester: Wiley & Sons, 2003. P. 747-772.

58. Berten V., Gaujal B. Brokering strategies in computational grids using stochastic prediction models // Parallel Computing 2007. V. 33. P. 238-249.

59. England D., Weissman J. B. Costs and benefits of load sharing in the computational grid // Lecture Notes in Computer Science. 2004. V. 3277. P. 160-175.

60. Ernemann C., Hamscher V., Yahyapour R. Economic Scheduling in Grid Computing // Lecture Notes in Computer Science. 2002. V. 2537. P. 128-152.

61. Li J., Sim K. M., Yahyapour R. Negotiation strategies considering opportunity functions for grid scheduling // Lecture Notes in Computer Science. 2007. V. 4641. P. 447-456.

62. Vanmechelen K., Broeckhove J. A comparative analysis of single-unit vickrey auctions and commodity markets for realizing grid economies with dynamic pricing // Lecture Notes in Computer Science. 2007. V. 4685. P. 98-111.

63. Combinatorial Auctions. Cramton P., Shoham Y., Steinberg R. (eds.) Cambridge: MIT Press, 2006.

64. Foster I., Jennings N.R., Kesselman C. Brain meets brawn: Why grid and agents need each other // Lecture Notes in Computer Science. 2005. V. 3394. P. 8-15.

65. Li J., Yahyapour R. Negotiation strategies for grid scheduling // Lecture Notes in Computer Science. 2006. V. 3947. P. 42-52.

66. Li J., Yahyapour R. Learning-based negotiation strategies for grid scheduling // Proc. Int. Symp. on Cluster Computing and the Grid. IEEE Comp. Sc. Press, 2006. P. 567-583.

67. Nguyen T.D., Jennings N.R. A heuristic model of concurrent bilateral negotiations in incomplete information settings // Proc. 18th Int. Joint Conf. on AI. IEEE Comp. Sc. Press, 2003. P. 1467-1469.

68. Kraus S. Strategic Negotiation in Multi-Agent Environments. Cambridge: MIT Press, 2001.

69. Abramson D., Buyya R., Giddy J. A computational economy for Grid computing and its implementation in the nimrod-g resource broker // Future Generation Computer Systems. 2002. V. 18 N. 8. P. 1061-1074.

70. AuYoung A., Chun B.N., Snoeren A.C. et al. Resource allocation in federated dis-tributed computing infrastructures // Proc. 1st Workshop on Operating System and Architectural Support for the On-demand IT Infra-Structure. Boston, 2004. CDROM.

71. Buyya R., Abramson D., Venugopal S. The Grid Economy // Proc. IEEE. 2005. V. 93. N. 3. P. 698-714.

72. Chun B.N., Culler D.E. User-centric performance analysis of market-based cluster batch schedulers // Proc. 2nd IEEE / ACM Int. Symp. on Cluster Computing and the Grid. IEEE Comp. Sc. Press, 2002. P. 22-30.

73. Feldman M., Lai K., Zhang L. A price-anticipating resource allocation mechanism for distributed shared clusters // Proc. ACM Conf. Electronic Commerce. New York: ACM, 2005. P. 127-136.

74. Gomoluch J., Schroeder M. Market-based resource allocation for Grid computing: A model and simulation // Proc. Int. Middleware Conference. ACM Press, 2005. P. 211-218.

75. Stuer G., Vanmechelen K., Broeckhove J. A Commodity Market Algorithm for Pricing Substitutable Grid Resources // Fut. Gen. Comp. Sys. 2007. V. 23. N. 5. P. 688-701.

76. Vanmechelen K., Stuer G., Broeckhove J. Pricing Substitutable Grid Resources using Commodity Market Models // Proc. 3th Int. Workshop on Grid Economics and Business Models. Singapore: World Scientific, 2006. P. 103-112.

77. Bredin J., Maheswaran R.T., Cëagri Imer T. et al. Computational markets to regulate mobile-agent systems // Autonomous Agents and Multi-Agent Systems. 2003. V. 6. N. 3. P. 235-263.

78. Dash R., Parkes D., Jennings N. Computational Mechanism

Design: A Call to Arms // IEEE Intelligent Systems. 2003. V. 18. N. 6. P. 40-47.

79. Joita L., Rana O., Freitag F. et al. A catallactic market for data mining services // Future Generation Computer Systems. 2007. V. 23. N. 1. P. 146-153.

80. Chen M., Yang G., Liu X. Gridmarket: A Practical, Efficient Market Balancing Resource for Grid and P2P Computing // Lecture Notes in Computer Science. 2003. V. 3033. P. 612-619.

81. Grosu D., Das A. Auction-Based Resource Allocation Protocols in Grids // Proc. 16th IASTED Int. Conf. on Parallel and Distributed Computing and Systems. Cambridge: MIT, 2004. P. 20-27.

82. Kale L. V., Kumar S., Potnuru M. et al. Faucets: Efficient Resource Allocation on the Computational Grid // Proc. Int. Conf. on Parallel Processing. IEEE Comp. Sc. Press, 2004. P. 396-405.

83. Kant U., Grosu D. Double Auction Protocols for Resource Allocation in Grids // Int. Conf. on Information Technology: Coding and Computing. IEEE Comp. Sc. Press, 2005. V. 1. P. 366-371.

84. Wolski R., Plank J. S., Brevik J. et al. Analyzing Market-Based Resource Allocation Strategies for the Computational Grid // Int. J. of High Performance Computing Applications. 2001. V. 15. N. 3. P. 258-281.

85. Xiao L., Zhu Y., Lionel M. et al. GridIS: An Incentive-Based Grid Scheduling // 19th IEEE Int. Parallel and Distributed Processing Symp. IEEE Comp. Sc. Press, 2005. V. 1. P. 65.

86. McLaughlin D., Sardesai S., Dasgupta P. Preemptive Scheduling for Distributed Systems // Proc. 11th Int. Conf. on Parallel and Distributed Computing Systems. IEEE Comp. Sc. Press, 1998. P. 222-227.

87. Ahmad I., Shamala S., Othman M. et al. Preemptive Utility Accrual Scheduling Algorithm for Adaptive Real Time System // Int. J. of Computer Science and Network Security. 2008. V. 8. N. 5. P. 57-61.

88. Li P. Utility Accrual Real Time Scheduling: Models and Algorithms. Ph.D. dissertation. Blacksburg: Virginia Polytechnic Institute and State University, 2004.

89. Gupta B., Palis M. Online real-time preemptive scheduling of jobs with deadlines on multiple machines // J. Scheduling. 2001. N. 4. P. 297–312.

90. Reddy S. Market Economy Based Resource Allocation in Grids. A thesis for the degree of Master of Technology in Information Technology. Kharagpur: School of Inf. Tech. Indian Institute of Technology, 2006.

91. Iamnitchi A., Foster I. On fully decentralized resource discovery in grid environments // Lecture Notes in Computer Science. 2005. V. 2242. P. 51-62 .

92. Ranjan R., Harwood A., Buyya R. Sla-based cooperative superscheduling algorithms for computational grids // Proc. 8th IEEE Int. Conf. on Cluster Computing. IEEE Computer Society Press, 2006. abs/cs/0605057.

93. Ranjan R., Harwood A., Buyya R. Grid Federation: An Economy Based, Scalable Distributed Resource Management System for Large-Scale Resource Coupling. Technical Report, GRIDS-TR-2004-10. Grid Computing and Distributed Systems Laboratory, University of Melbourne, 2004.

94. Tchernykh A., Ramirez J. M., Avetisyan A. et al. Two Level Job-Scheduling Strategies for a Computational Grid // Lecture Notes in Computer Science. 2006. V. 3911. P. 774-781.

95. Sabin G., Kettimuthu R., Rajan A. et al. Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment // Lecture Notes in Computer Science. 2003. V. 2862. P. 87-104.

96. Zhuk S., Chernykh A., Kuzjurin N. et al. Comparison of Scheduling Heuristics for Grid Resource Broker // Proc. 3rd Int. Conf. on Parallel Comp. Systems. IEEE Comp. Sc. Press, 2004. P. 388-392.

97. //http: //www.clusterresources.com

98. Hopper E., Turton B.C.H. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem // Europian Journal of Operational Research. 2001. V. 128. P. 34–57.

99. Jansen K. Scheduling malleable parallel tasks: an asymptotic fully polynomial-time approximation scheme // Algorithmica. 2004. V. 39. N. 1. P. 59–81.

100. Kenyon C., Remila E. A near optimal solution to a two dimensional cutting stock problem // Math. of Operations Res. 2000. V. 25. P. 645-656.

101. Yu K.-M., Luo Z.-J., Chou C.-H. et al. A fuzzy neural network based scheduling algorithm for job assignment on computational grids // Lecture Notes in Computer Science. 2007. V. 4658. P. 533-542.

102. Li H., Buyya R. Model-Driven Simulation of Grid Scheduling Strategies // Proc. 3rd IEEE Int. Conf. on e-Science and Grid Computing. IEEE Comp. Sc. Press, 2007. P. 287-294.

103. Li H., Wolters L. Towards a better understanding of workload dynamics on data-intensive clusters and grids // Proc. 21st IEEE Int. Parallel and Distributed Processing Symp. IEEE Comp. Sc. Press, 2007. P. 60.

104. Buyya R., Murshed M., Abramson D. et al. Scheduling parameter sweep applications on global grids: a deadline and budget constrained costtime optimization algorithm // Software Practice and Experience. 2005. V. 35. P. 491-512.

105. Dumitrescu C., Raicu I., Foster I. Di-gruber: A distributed approach to grid resource brokering // Proc. 2005

ACM / IEEE Conf. on Supercomputing. IEEE Comp. Sc. Comp. Sc. Press, 2005. P. 38.

106. Ranganathan K., Foster I. Decoupling computation and data scheduling in distributed data-intensive applications // Proc. 11th IEEE Symp. on High Performance Distributed Comp. IEEE Comp. Sc. Press, 2002. P. 352-358.

107. Bucur A., Epema D. Trace-based simulations of processor co-allocation policies in multiclusters // Proc. 12th IEEE Symp. on High Performance Distributed Comp. IEEE Comp. Sc. Press, 2003. P. 70-79.

108. He L., Jarvis S., Spooner D. et al. Mapping dag-based applications to multiclusters with background workload // Proc. 5th IEEE Symp. on Cluster Comp. and The Grid. IEEE Comp. Sc. Press, 2005. P. 855-862.

109. Ramakrishnan A., Singh G., Zhao H. et al. Scheduling data intensive workflows on to storage-constrained distributed resources // Proc. 7th IEEE Symp. on Cluster Comp. and The Grid. IEEE Comp. Sc. Press, 2007. P. 401-409.

110. Li H., Muskulus M. Analysis and modeling of job arrivals in a production grid // ACM SIGMETRICS Performance Evaluation Review. 2007. V. 34. P. 59-70.

111. Li H., Heusdens R., Muskulus M. et al. Analysis and synsynthesis of pseudo-periodic job arrivals in grids: A matching pursuit approach // Proc. 7th IEEE Symp. on Cluster Comp. and The Grid. IEEE Comp. Sc. Press, 2007. P. 183-196.

112. Душин Ю. А. Модель оценки стоимости гетерогенных ресурсов в Грид // Системы и средства информатики: Спец. вып. "Математические модели в информационных технологиях". М.: ИПИ РАН, 2006. С. 163-172.

113. Коновалов М. Г., Душин Ю. А., Малашенко Ю. Е. и др. Модель взаимодействия потребителей с удаленными вычислительными ресурсами через посредников // Системы и средства информатики. Вып. 19. М.: Наука, 2009. С. 5-33.

114. http://x-com.parallel.ru

115. Puterman M.L. Markov decision processes: Discrete stochastic dynamic programming. New York: Wiley, 1994.

116. Sragovich V.G. Mathematical Theory of Adaptive Control. Singapore: World Scientific, 2006.

117. Mahadevan S. Average reward reinforcement learning foundations, algorithms, and empirical results // Machine Learning. 1996. V. 22. P. 159-195.

118. Bartlett P.L. An Introduction to Reinforcement Learning Theory: Value Function Methods // Lecture Notes in Computer Science. 2003. V. 2600. P. 184-202.

119. Коновалов М. Г. Методы адаптивной обработки информации и их приложения. М.: ИПИ РАН, 2007.

СОДЕРЖАНИЕ

Список литературы

[1] Foster I., Kesselman C., Nick J. et al. The physiology of the grid: An open grid services architecture for distributed systems integration. // http://www.globus.org/research/papers.html.

[2] Коновалов М. Г. Адаптивное управление процессом размещения заданий в модели коллективного использования распределенных вычислительных ресурсов // Материалы 2-й международной конференции "Распределенные вычисления и Грид-технологии в науке и образовании". Дубна, 2006. С. 124-127.

[3] Антонов С. В., Душин Ю. А., Коновалов М. Г., Шоргин С. Я. Разработка математических моделей и методов распределения заданий в системе распределённых вычислений // Системы и средства информатики. М.: Наука. 2006. С. 32-45.

[4] Gaeta M., Konovalov M., Shorgin S. Development of mathematical models and methods of task distribution in distributed computing system // Reliability: Theory & Applications. 2006. V.1. N.4. P. 16-21.

[5] Foster I., Kesselman C., Tuecke S. The Anatomy of the Grid:Enabling Scalable Virtual Organizations. // International J.of Supercomputer Applications. 2001. V.15. N. 3. P. 200-222.

[6] Baker M., Buyya R., Laforenza D. Grids and Grid Technologies for Wide-Area Distributed Computing // Software: Practice and Experience. 2002. V. 32. N. 15. P. 1437-1466.

[7] Berman F., Fox G., Hey T. Grid Computing: Making the Global Infrastructure a Reality.Chichester: John Wiley & Sons, 2003.

[8] Foster I., Kesselman C., Nick J. et al. Grid Services for Distributed System Integration // IEEE Computer. 2002. V. 35. N. 6. P. 37-46.

[9] Sabin G., Sahasrabudhe V., Sadayappan P. On fairness indistributed job scheduling across multiple sites // IEEE Int.Conf. on Cluster Computing. IEEE Comp. Sc. Press, 2004. P. 35-44.

[10] Gentzsh W. Sun grid engine: Towards creating a compute power grid // Proc. 1-st IEEE/ACM Int. Symp. on Cluster Computing and the Grid. IEEE Comp. Sc. Press, 2002. P. 35-36.

[11] Belalem G., Bouhraoua F. Dynamic strategy of placement of the replicas in data grid // Lecture Notes in Computer Science. 2007. V. 4671. P. 496-506.

[12] Cho S., Lee M., In J. et al. Policy based scheduling for resource allocation on grid // Lecture Notes in Computer Science. 2007. V. 4537. P. 229-234.

[13] Iosup A., Jan M., Sonmez O. et al.The characteristics and performance of groups of jobs in grids // Lecture Notes in Computer Science. 2007. V. 4641. P. 382-393.

[14] Ishii R. P., De Mello R. F., Yang L.T. A complex network-based approach for job scheduling in grid environments // Lecture Notes in Computer Science. 2007. V. 4782. P. 204-215.

[15] Krasnotcshekov V., Vakhitov A. Adaptive scheduling and esource assessment in grid // Lecture Notes in Computer Science. 2007. V. 4671. P. 240-244.

[16] Lewis R., Torczon V. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds // SIAM J. on Optimization. 2002. V. 12. N. 4. P. 1075-1089.

[17] Li B., Zhao D. Online algorithms for single machine schedulers to support advance reservations from grid jobs // Lecture Notes in Computer Science. 2007. V. 4782. P. 239-248.

[18] Nou R., Kounev S., Torres J. Building online performance models of grid middleware with fine-grained load-balancing: a globus toolkit case study // Lecture Notes in Computer Science. 2007. V. 4748. P. 125-140.

[19] Quetier B., Cappello F. A survey of Grid research tools: simulators, emulators and real life platforms // http://sab1.sscc.ru/imacs2005/papers/T1-I-58-1069.pdf.

[20] Snavely A., Chun G., Casanova H. et al. Benchmarks for Grid Computing: A Review of Ongoing Efforts and Future Directions // ACM Sigmetrics Performance Evaluation Review. 2003. V. 30. N. 4. P. 27-32.

[21] Song E., Jeon Y., Han S.et al. Hierarchical and Dynamic Information Manage-ment Framework on Grid Computing // Lecture Notes in Computer Science. 2006. V. 4096. P. 151-161.

[22] Mohammadi A., Akl S. Scheduling Algorithms for Real-Time Systems. Technical Report No. 2005-499. Queen's University Kingston, Ontario, 2005.

[23] Ousterhout J. K. Scheduling techniques for concurrent systems // Proc. 3rd IEEE Int. Conf. on Distributed Computer Systems. IEEE Comp. Sc. Press, 1982. P. 22-30.

[24] Paranhos D., Cirne W., Brasileiro F. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids // Proc. Int. Conf. Parallel and Distributed Computing, Euro-Par. 2003. P. 169-80.

[25] Parra-Hernandez R., Vanderster D., Dimopoulos N. J. Resource management and knapsack formulations on the grid // Proc. 5th IEEE/ACM Int. Workshop on Grid Computing. IEEE Comp. Sc. Press, 2004. P. 94-101.

[26] Frey J., Tannenbaum T., Livny M. et al. Condorg: A computation management agent for multi-institutional grids // Cluster Computing. 2002. V. 5. Iss. 3. P. 237 - 246.

[27] Vadhiyar S.S., Dongarra J.J. A Metascheduler for the Grid // Proc. of 11-th IEEE Symp. High Performance Distributed Computing. IEEE Comp. Sc. Press, 2002. P. 343- 351.

[28] Li C., LI L. Utility-based QoS optimisation strategy for multi-criteria scheduling on the grid // J. of parallel and distributed computing. 2007. V. 67. N. 2. P. 142-153.

[29] Salmani V., Ensafi R., Khatib-Astaneh N. et. al. A Fuzzy-Based Multi-criteria Scheduler for Uniform Multiprocessor Real-Time Systems // Proc. 10th Int. Conf. on Information Technology, (ICIT 2007). IEEE Comp. Sc. Press, 2007. P. 179-184.

[30] Salmani V., Naghibzadeh M., Kahani M. et al. Multi-criteria Scheduling of Soft Real-time Tasks on Uniform Multiprocessors Using Fuzzy Inference // Advances and Innovations in Systems, Computing Sciences and Software Engineering. Springer Netherlands, 2007. P. 439-444.

[31] Klusáček D., Rudová H., Baraglia R. et al. Comparison Of Multi-Criteria Scheduling Techniques // Grid Computing. Springer, 2008. P. 173-184.

[32] Song S., Hwang K., Kwok Y.-K. Trusted grid computing with security binding and trust integration // J.l of Grid Computing. 2005. V. 3. P.53-73.

[33] Yagoubi B., Slimani, Y. Dynamic Load Balancing Strategy for Grid Computing // Transactions on Engineering, Computing and Technology. 2006. N. 13. P. 260-265.

[34] Elmroth E., Tordsson J. An Interoperable, Standards-based Grid Resource Broker and Job Submission Service // 1-st Int. Conf. on e-Science and Grid Computing. IEEE Comp. Sc. Press, 2005. P. 212 - 220.

[35] Frumkin M., Wijngaart R. NAS Grid Benchmarks: A Tool for Grid Space Exploration // Cluster Computing. 2002. V. 5. Iss. 3. P. 247 - 255.

[36] Herrera J., Huedo E., Montero R.S. et al. Loosely-coupled Loop Scheduling in Computational Grids // Proc. 20th IEEE International Parallel & Distributed Processing Symposium. IEEE Comp. Sc. Press, 2006. P. 6.

[37] Montero R.S., Huedo E., Llorente I.M. Benchmarking of high throughput computing applications on Grids // Parallel Computing. 2006. V. 32. Iss. 4. P. 267 - 279.

[38] Phatanapherom S., Uthayopas P., Kachitvichyanukul V. Dynamic scheduling II: fast simulation model for grid scheduling using HyperSim // Proc. 35th conf. on Winter simulation. New Orleans, 2003. P. 1494-1500.

[39] Zhou J., Yu K.-M., Chou C.-H. et al. Dynamic Resource Broker and Fuzzy Logic Based Scheduling Algorithm in Grid Environment // Proc. Int. Conf. on Adaptive and Natural Computing Algorithms. ICANNGA, 2007. P. 604–613.

[40] Poddar H., Singh J., Sekhon G.S. Dynamic Scheduling for Hard Real-Time Multiprocessor Systems // Proc. of National Conf. on Challenges & Opportunities in Information Technology (COIT-2007). Mandi Gobindgarh, 2007. P. 76-80.

[41] Giersh A., Rober, Y., Vivien F. Scheduling tasks sharing files on heterogeneous master-slave platforms // Proc. 12th Euromicro Workshop in Par., Dist. and Network-based. IEEE CS Press, 2004. P. 364-371.

[42] Senger H., Hruschka E.R., Silva F.A.B. et al. In-hambu: Data Mining Using Idle Cycles in Clusters of PCs // Proc. IFIP Intl. Conf. on Network and Parallel Computing (NPC'04). Lecture Notes in Computer Science. 2004. V. 3222. P. 213-220.

[43] Silva F.A.B., Carvalho S., Hruschka E.R. A Scheduling Algorithm for Running Bag-of-Tasks Data Mining Application on the Grid // Lecture Notes in Computer Science. 2004. V. 3419. P. 254-262.

[44] Silva F.A.B., Carvalho S., Senger H. et al. Running Data Mining Applications on the Grid: a Bag-of-Tasks

Approach // Int. Conf. on Computational Science and its Appllications (ICCSA). Lecture Notes in Computer Science. 2004. V. 3044. P. 168-177.

[45] Slegers J., Mitrani I., Thomas N. Optimal dynamic server allocation in systems with on/off sources // Lecture Notes in Computer Science. 2007. V. 4748. P. 186-199.

[46] Marchal L., Yang Y., Casanova H. et al. Steady-state scheduling of multiple divisible load applications on wide-area distributed computing platforms // Int. J. of High Performance Computing Applications. 2006. V. 20. N. 3. P. 365-381.

[47] Banino C., Beaumont O., Carter L. et al. Scheduling strategies for master-slave tasking on heterogeneous processor platforms // IEEE Trans. Parallel Distributed Systems. 2004. V. 15. N. 4. P. 319-330.

[48] Beaumont O., Casanova H., Legrand A. et al. Scheduling divisible loads for star and tree networks: main results and open problems // IEEE Trans. Parallel and Distributed Systems. 2005. V. 16 N. 3. P. 207-218.

[49] Vanderster D. C., Dimopoulos N. J., Sobie R. J. Metascheduling multiple resource types using the MMKP // Proc. 7th IEEE/ACM Int. Conf. on Grid Computing. EEE Comp. Sc. Press, 2006. P. 231-237.

[50] Cappello F., Caron E., Dayde M. et al. Grid'5000: a large scale, reconfigurable, controlable and monitorable Grid platform // Proc. 6th IEEE/ACM Int. Workshop on Grid Computing. EEE Comp. Sc. Press, 2006. P. 99–106.

[51] The Distributed ASCI Supercomputer (DAS) //http: //www.cs.vu.nl/das3

[52] Bucur A. I. D., Epema D. H. J. The Performance of Processor Co-Allocation in Multi-cluster Systems // Proc. 3rd IEEE/ACM Int'l Symp. on Cluster Computing and the GRID (CCGrid2003). IEEE Comp. Sc. Press, 2003. P. 302-309.

[53] Ernemann C., Hamscher V., Schwiegelshohn U. et al. On Advan-tages of Grid Computing for Parallel Job Scheduling // Proc. 2nd IEEE/ACM Int. Symp. on Cluster Computing and the GRID (CCGrid2002). IEEE Comp. Sc. Press, 2002. P. 39-46.

[54] Mohamed H.H., Epema D.H.J. Experiences with the koala Co-Allocating Scheduler in Multiclusters // Proc. 5th IEEE/ACM Int. Symp. on Cluster Computing and the GRID (CCGrid2005). IEEE Comp. Sc. Press, 2005. V. 2. P. 784 - 791.

[55] Mohamed H.H., Epema D.H.P. The design and implementation of the KOALA co-allocating grid scheduler // Lecture Notes in Computer Science. 2005. V. 3470. P. 640-650.

[56] Ballier A., Caron E., Epema D. et al. Simulating grid schedulers with deadlines and co-allocation // Research report (2006) [inria-00077051 — version 1]. Laboratoire de l'Informatique du Parallelisme. http://www.ens-lyon.fr/LIP.

[57] Wolski R., Brevik J., Plank J.S. et al. Grid Resource Allocation and Control Using Computational Economics

// Grid Computing: Making the Global Infrastructure a Reality. Chichester: Wiley and Sons, 2003. P. 747-772.

[58] Berten V., Gaujal B. Brokering strategies in computational grids using stochastic prediction models // Parallel Computing 2007. V. 33. P. 238-249.

[59] England D., Weissman J. B. Costs and benefits of load sharing in the computational grid // Lecture Notes in Computer Science. 2004. V. 3277. P. 160-175.

[60] Ernemann C., Hamscher V., Yahyapour R. Economic Scheduling in Grid Computing // Lecture Notes in Computer Science. 2002. V. 2537. P. 128-152.

[61] Li J., Sim K. M., Yahyapour R. Negotiation strategies considering opportunity functions for grid scheduling // Lecture Notes in Computer Science. 2007. V. 4641. P. 447-456.

[62] Vanmechelen K., Broeckhove J. A comparative analysis of single-unit vickrey auctions and commodity markets for realizing grid economies with dynamic pricing // Lecture Notes in Computer Science. 2007. V. 4685. P. 98-111.

[63] Combinatorial Auctions. Cramton P., Shoham Y., Steinberg R. (eds.) Cambridge: MIT Press, 2006.

[64] Foster I., Jennings N.R., Kesselman C. Brain meets brawn: Why grid and agents need each other // Lecture Notes in Computer Science. 2005. V. 3394. P. 8-15.

[65] Li J., Yahyapour R. Negotiation strategies for grid scheduling // Lecture Notes in Computer Science. 2006. V. 3947. P. 42-52.

[66] Li J., Yahyapour R. Learning-based negotiation strategies for grid scheduling // Proc. Int. Symp. on Cluster Computing and the Grid (CCGRID2006). IEEE Comp. Sc. Press, 2006. P. 567-583

[67] Nguyen T.D., Jennings N.R. A heuristic model of concurrent bi-lateral negotiations in incomplete information settings // Proc. 18th Int. Joint Conf. on AI. IEEE Computer Society Press, 2003. P. 1467-1469.

[68] Kraus S. Strategic Negotiation inMulti-Agent Environments. Cambridge: MIT Press, 2001.

[69] Abramson D., Buyya R., Giddy J. A computational economy for Grid computing and its implementation in the nimrod-g resource broker // Future Generation Computer Systems. 2002. V. 18 N. 8. P. 1061-1074.

[70] AuYoung A., Chun B.N., Snoeren A.C. et al. Resource allocation in federated dis-tributed computing infrastructures // Proc. 1st Workshop on Operating System and Architectural Support for the On-demand IT InfraStructure. Boston, 2004. CDROM.

[71] Buyya R., Abramson D., Venugopal S. The Grid Economy // Proc. IEEE. 2005. V. 93. N.3. P. 698-714.

[72] Chun B.N., Culler D.E. User-centric performance analysis of market-based cluster batch schedulers // Proc. 2nd IEEE/ACM Int. Symp. on Cluster Computing and the Grid. EEE Comp. Sc. Press, 2002. P. 22-30.

[73] Feldman M., Lai K., Zhang L. A price-anticipating resource allocation mechanism for distributed shared

clusters // Proc. ACM Conf. Electronic Commerce (EC). New York: ACM, 2005. P. 127-136.

[74] Gomoluch J., Schroeder M. Market-based resource allocation for Grid computing: A model and simulation // Proc. Int. Middleware Conference. ACM Press, 2005. P. 211-218.

[75] Stuer G., Vanmechelen K., Broeckhove J. A Commodity Market Algorithm for Pricing Substitutable Grid Resources // Fut. Gen. Comp. Sys. 2007. V. 23. N. 5. P. 688-701.

[76] Vanmechelen K., Stuer G., Broeckhove J. Pricing Substitutable Grid Resources using Commodity Market Models // Proc. 3th Int. Workshop on Grid Eco-nomics and Business Models. Singapore: World Scientific, 2006. P. 103-112.

[77] Bredin J., Maheswaran R.T., Cëagri Imer T. et al. Computational mar-kets to regulate mobile-agent systems // Autonomous Agents and Multi-Agent Systems. 2003. V. 6. N. 3. P. 235-263.

[78] Dash R., Parkes D., Jennings N. Computational Mechanism Design: A Call to Arms // IEEE Intelligent Systems. 2003. V. 18. N. 6. P. 40-47.

[79] Joita L., Rana O., Freitag F. et al. A catallactic market for data mining services // Future Generation Computer Systems. 2007. V. 23. N. 1. P. 146-153.

[80] Chen M., Yang G., Liu X. Gridmarket: A Practical, Efficient Market Balancing Resource for Grid and P2P

Computing // Lecture Notes in Computer Science. 2003. V. 3033. P. 612-619.

[81] Grosu D., Das A. Auction-Based Resource Allocation Protocols in Grids // Proc. 16th IASTED Int. Conf. on Parallel and Distributed Computing and Systems. Cambridge: MIT, 2004. P. 20-27.

[82] Kale L. V., Kumar S., Potnuru M. et al. Faucets: Efficient Resource Allocation on the Computational Grid // Proc. Int. Conf. on Parallel Processing (ICPP 2004). EEE Comp. Sc. Press, 2004. P. 396-405.

[83] Kant U., Grosu D. Double Auction Protocols for Resource Allocation in Grids // Int. Conf. on Information Technology: Coding and Computing (ITCC'05). EEE Comp. Sc. Press, 2005. V. 1. P. 366-371.

[84] Wolski R., Plank J. S., Brevik J. et al. Analyzing Market-Based Resource Allocation Strategies for the Computational Grid // Int. J. of High Performance Computing Applications. 2001. V. 15. N. 3. P. 258-281.

[85] Xiao L., Zhu Y., Lionel M. et al. GridIS: An Incentive-Based Grid Scheduling // 19th IEEE Int. Parallel and Distributed Processing Symp. (IPDPS'05). EEE Comp. Sc. Press, 2005. V. 1. P. 65.

[86] McLaughlin D., Sardesai S., Dasgupta P. Preemptive Scheduling for Distributed Systems // Proc. 11th Int. Conf. on Parallel and Distributed Computing Systems. EEE Comp. Sc. Press, 1998. P. 222-227.

[87] Ahmad I., Shamala S., Othman M. et al. Preemptive Utility Accrual Scheduling Algorithm for Adaptive Real

Time System // Int. J. of Computer Science and Network Security. 2008. V. 8. N. 5. P. 57-61.

[88] Li P. Utility Accrual Real Time Scheduling: Models and Algorithms. Ph.D. dissertation. Blacksburg: Virginia Polytechnic Institute and State University, 2004.

[89] Gupta B., Palis M. Online real-time preemptive scheduling of jobs with deadlines on multiple machines // J. Scheduling. 2001. N. 4. P. 297–312.

[90] Reddy S. Market Economy Based Resource Allocation in Grids. A thesis for the degree of Master of Technology in Information Technology. Kharagpur: School of Inf. Tech. Indian Institute of Technology, 2006.

[91] Iamnitchi A., Foster I. On fully decentralized resource discovery in grid environments // Lecture Notes in Computer Science. 2005. V. 2242. P. 51-62 .

[92] Ranjan R., Harwood A., Buyya R. Sla-based cooperative superscheduling algorithms for computational grids // Proc. 8th IEEE Int. Conf. on Cluster Computing (Cluster 2006). IEEE Computer Society Press, 2006. abs/cs/0605057.

[93] Ranjan R., Harwood A., Buyya R. Grid Federation: An Economy Based, Scalable Distributed Resource Management System for Large-Scale Resource Coupling. Technical Report, GRIDS-TR-2004-10. Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, 2004.

[94] Tchernykh A., Ramirez J. M., Avetisyan A. et al. Two Level Job-Scheduling Strategies for a Computational Grid

// Lecture Notes in Computer Science. 2006. V. 3911. P. 774-781.

[95] Sabin G., Kettimuthu R., Rajan A. et al. Scheduling of Parallel Jobs in a Heterogeneous Multi-Site Environment // Lecture Notes in Computer Science. 2003. V. 2862. P. 87-104.

[96] Zhuk S., Chernykh A., Kuzjurin N. et al. Comparison of Scheduling Heuristics for Grid Resource Broker // Proc. 3rd Int. Conf. on Parallel Computing Systems. IEEE Computer Society Press, 2004. P. 388-392.

[97] //http: //www.clusterresources.com

[98] Hopper E., Turton B.C.H. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem // Europian Journal of Operational Research. 2001. V. 128. P. 34–57.

[99] Jansen K. Scheduling malleable parallel tasks: an asymptotic fully polynomial-time approximation scheme // Algorithmica. 2004. V. 39. N. 1. P. 59–81.

[100] Kenyon C., Remila E. A near optimal solution to a two dimensional cutting stock problem // Math. of Operations Res. 2000. V. 25. P. 645-656.

[101] Yu K.-M., Luo Z.-J., Chou C.-H. et al. A fuzzy neural network based scheduling algorithm for job assignment on computational grids // Lecture Notes in Computer Science. 2007. V. 4658. P. 533-542.

[102] Li H., Buyya R. Model-Driven Simulation of Grid Scheduling Strategies // Proc. 3rd IEEE Int. Conf. on e-

Science and Grid Computing (eScience07). IEEE Comp. Sc. Press, 2007. P.287-294.

[103] Li H., Wolters L. Towards a better understanding of workload dynamics on data-intensive clusters and grids // Proc. 21st IEEE Int. Parallel and Distributed Processing Symp. (IPDPS). IEEE Comp. Sc. Press, 2007. P. 60.

[104] Buyya R., Murshed M., Abramson D. et al. Scheduling parameter sweep applications on global grids: a deadline and budget constrained costtime optimization algorithm // Software Practice and Experience. 2005. V. 35. P. 491-512.

[105] Dumitrescu C., Raicu I., Foster I. Di-gruber: A distributed approach to grid resource brokering // Proc. 2005 ACM/IEEE conference on Supercomputing. IEEE Comp. Sc. Press, 2005. P. 38.

[106] Ranganathan K., Foster I. Decoupling computation and data scheduling in distributed data-intensive applications // Proc. 11th IEEE Symp. on High Performance Distributed Computing (HPDC). IEEE Comp. Sc. Press, 2002. P. 352-358.

[107] Bucur A., Epema D. Trace-based simulations of processor co-allocation policies in multiclusters // Proc. 12th IEEE Symp. on High Performance Distributed Computing (HPDC). IEEE Comp. Sc Press, 2003. P. 70-79.

[108] He L., Jarvis S., Spooner D. et al. Mapping dag-based applications to multiclusters with background workload // Proc. 5th IEEE Symp. on Cluster Computing and The Grid (CCGrid). IEEE Comp. Sc. Press, 2005. P. 855-862.

[109] Ramakrishnan A., Singh G., Zhao H. et al. Scheduling data intensive workflows on to storage-constrained distributed resources // Proc. 7th IEEE Symp. on Cluster Computing and The Grid (CCGrid). IEEE Comp. Sc. Press, 2007. P. 401-409.

[110] Li H., Muskulus M. Analysis and modeling of job arrivals in a production grid // ACM SIGMETRICS Performance Evaluation Review. 2007. V. 34. P. 59-70.

[111] Li H., Heusdens R., Muskulus M. et al. Analysis and synthesis of pseudo-periodic job arrivals in grids: A matching pursuit approach // Proc. 7th IEEE Symp. on Cluster Computing and The Grid (CCGrid)). EEE Comp. Sc. Press, 2007. P. 183-196.

[112] Душин Ю. А. Модель оценки стоимости гетерогенных ресурсов в Грид // Системы и средства информатики: Спец. вып. "Математические модели в информационных технологиях". М.: ИПИ РАН, 2006. С. 163-172.

[113] Коновалов М. Г., Душин Ю. А., Малашенко Ю. Е. и др. Модель взаимодействия потребителей с удаленными вычислительными ресурсами через посредников // Системы и средства информатики. Вып. 19. М.: Наука, 2009. С. 5-33.

[114] http://x-com.parallel.ru

[115] Puterman M.L. Markov decision processes: Discrete stochastic dynamic programming. New York: Wiley, 1994.

[116] Sragovich V.G. Mathematical Theory of Adaptive Control. Singapore: World Scientific, 2006.

[117] Mahadevan S. Average reward reinforcement learning foundations, algorithms, and empirical results // Machine Learning/ 1996. V. 22. P. 159-195.

[118] Bartlett P.L. An Introduction to Reinforcement Learning Theory: Value Function Methods // Lecture Notes in Computer Science. 2003. V. 2600. P. 184-202.

[119] Коновалов М. Г. Методы адаптивной обработки информации и их приложения. М.: ИПИ РАН, 2007.