Федеральное государственное бюджетное учреждение науки
Институт прикладных математических исследований
Карельского научного центра Российской Академии Наук

На правах рукописи

Никитина Наталия Николаевна

Математические модели управления заданиями и защиты информации в высокопроизводительных вычислительных системах

05.13.18 – Математическое моделирование, численные методы и комплексы программ

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата технических наук

Научный руководитель
д. ф.-м. н., проф.
Мазалов Владимир Викторович

Петрозаводск – 2014

Содержание

Введение

Актуальность темы исследования. В ряде отраслей фундаментальной и прикладной науки возникают задачи, требующие выполнения значительных объемов вычислений, а также обработки, хранения, передачи и визуализации больших объемов данных. Для решения таких задач используются высокопроизводительные вычислительные системы, значительно превосходящие традиционные вычислительные средства по техническим характеристикам. Для эффективного использования высокопроизводительной вычислительной системы необходимо управлять ее ресурсами, то есть определять порядок их запуска и распределять между ними вычислительные ресурсы, в первую очередь процессорное время и оперативную память.

Критериями эффективности при управлении очередью заданий могут являться общее время выполнения набора заданий, среднее время ожидания задания в очереди, средняя загрузка системы в единицу времени и др., а также комбинации таких критериев. Выбор характеристик заданий и критериев эффективности во многом зависит от особенностей операционной системы и архитектуры вычислительной системы в целом, а также от специфики решаемых задач. В связи с этим разработке и оценке алгоритмов планирования заданий посвящено множество исследований на протяжении нескольких десятков лет.

Эффективное управление ресурсами включает в себя обеспечение безопасности информации в вычислительной системе. Сетевая атака типа DOS (от англ. Denial of Service — отказ в обслуживании) заключается в создании таких условий, в которых вычислительная система становится не в состоянии своевременно обслуживать все задания. При этом алгоритм управления заданиями, функционирующий в отсутствие атаки, становится малоэффективным или практически неприменимым.

Цель диссертационной работы заключается в построении и исследовании свойств математических моделей управления ресурсами высокопроизводи-

тельных вычислительных систем и защиты информации в них с применением методов теории вероятностей, теории оптимизации и теории некооперативных игр.

Достижение поставленной цели требует решения следующих **задач**:

1. Задача повышения эффективности алгоритма Backfill для управления заданиями на вычислительном кластере. В качестве критериев эффективности принимаются количество ошибок при применении алгоритма и изменение среднего времени ожидания задания в очереди.

2. Задача минимизации нагрузки на сервер в централизованной системе распределенных вычислений.

3. Задача определения момента начала DoS-атаки на вычислительную систему с последующим изменением протокола обслуживания заданий.

Научная новизна работы заключается в следующем:

1. Разработана модификация алгоритма Backfill с принятием решения на основе аналитического выражения вероятности ошибки.

2. Построена теоретико-игровая модель управления заданиями в централизованной системе распределенных вычислений, предназначенной для проведения виртуального скрининга лекарств. Найдены аналитические выражения оптимальных стратегий и функций выигрыша узлов.

3. Найдено аналитическое решение разностного уравнения, полученного целочисленным приближением уравнения, описывающего динамику изменения среднего количества ошибок I или II рода при обнаружении момента DoS-атаки на вычислительную систему методом кумулятивных сумм.

4. Разработан алгоритм обслуживания заданий в присутствии DoS-атаки, позволяющий выбирать из очереди регулярные (ассоциированные с нор-

мальной активностью пользователей системы) задания, игнорируя искусственные (ассоциированные с атакой).

Практическая значимость работы заключается в следующем:

1. Реализована программа обслуживания заданий с применением модифицированного алгоритма Backfill.

2. Реализована программа поиска решений задачи минимизации нагрузки на сервер в системе распределенных вычислений методом прямого перебора.

3. Реализован программный пакет для управления заданиями на основе построенной модели, предназначенный для системы распределенных вычислений. Программный пакет апробирован в системе, реализованной в ходе выполнения совместного научно-исследовательского проекта между Институтом прикладных математических исследований КарНЦ РАН и Любекским Институтом экспериментальной дерматологии при университете г. Любек (Германия).

4. Реализована программа обслуживания заданий в присутствии DoS-атаки, допускающая использование различных видов и параметров распределений регулярного и искусственного потоков заданий. Программа прошла государственную регистрацию в Федеральной службе по интеллектуальной собственности, патентам и товарным знакам.

На защиту выносятся следующие **положения**:

1. Разработана модификация алгоритма Backfill для управления заданиями на вычислительном кластере.

2. Предложена и исследована теоретико-игровая модель управления заданиями в централизованной системе распределенных вычислений, предназначенной для проведения поиска лекарств.

3. Разработан метод кумулятивных сумм обнаружения момента начала DoS-атаки в потоке заданий, характеризуемом распределением Бернулли.

4. Предложен и реализован протокол обслуживания заданий в присутствии DoS-атаки, позволяющий выбирать из потока заданий регулярные и игнорировать искусственные.

Связь работы с научными программами, темами: основные результаты диссертации были получены в рамках выполнения исследований при финансовой поддержке РФФИ (проекты 11-06-00165а, 12-07-31147 мол_а), Германской службы академических обменов DAAD (долгосрочная научно-исследовательская стипендия с октября 2013 г. по июль 2014 г.) и Программы стратегического развития ПетрГУ.

Апробация результатов. Основные результаты диссертации были представлены и обсуждены на следующих конференциях:

1. Тринадцатый всероссийский симпозиум по прикладной и промышленной математике, 2–9 июня 2012 г., г. Петрозаводск;

2. Международный семинар «Networking Games and Management», 30 июня–2 июля 2012 г., г. Петрозаводск;

3. Двенадцатая всероссийская конференция «Высокопроизводительные параллельные вычисления на кластерных системах», 26–28 ноября 2012 г., г. Нижний Новгород;

4. XIV Российская конференция с участием иностранных ученых «Распределенные информационные и вычислительные ресурсы» 26–30 ноября 2012 г., г. Новосибирск;

5. Международный семинар «Russian-Chinese Seminar on Asymptotic Methods in Probability Theory and Mathematical Statistics», 10–14 июня 2013 г., г. Санкт-Петербург;

6. Первая российская конференция «Высокопроизводительные вычисления на базе BOINC: фундаментальные исследования и разработки», 9–13 сентября 2013 г., г. Петрозаводск;

7. VIII Международная научно-практическая конференция «Научно-образовательная информационная среда XXI века», 15–18 сентября 2014 г., г. Петрозаводск.

Публикации. Материалы диссертации опубликованы в 14 печатных работах, из них 6 статей в рецензируемых журналах, рекомендованных ВАК [1–6], одна статья в рецензируемом журнале [7], 2 статьи в трудах конференций [8, 9] и 5 тезисов докладов [10–14]. Одна статья принята к публикации в издании, индексируемом в библиографической базе данных Scopus [15]. Получено свидетельство о государственной регистрации программы для ЭВМ [16].

Структура и объем диссертации. Диссертационная работа состоит из введения, четырех глав, заключения и списка литературы.

Во введении отражена актуальность работы, поставлена цель исследования, обоснована научная новизна работы, сформулированы положения, выносимые на защиту, кратко описаны полученные результаты и показана их практическая ценность.

В первой главе приведен обзор литературы по теме исследования, описаны основные принципы функционирования высокопроизводительных вычислительных систем и основные задачи, возникающие при управлении вычислительным процессом в них.

Во второй главе приводится модификация алгоритма Backfill управления потоком заданий на вычислительном кластере и исследуются оценки его характеристик. Для оценки времени выполнения в модифицированном алгоритме предлагается использовать параметры распределения, полученные анализом истории выполнения заданий. Принятие решения осуществляется с использованием вероятности ошибки, выраженной в аналитическом виде.

Ожидаемое время выполнения задания в исходном алгоритме Backfill выражено точечной оценкой, что позволяет повысить производительность базового алгоритма управления заданиями при условии, что оценка является достаточно точной [17]. Однако получить точную оценку времени выполнения на практике, как правило, сложно или невозможно в связи со значительным количеством факторов, влияющих на ход вычислительного процесса: в частности, выполнение заданий может прерываться из-за ошибок, пользователи могут отменять задачи и склонны запрашивать значительно большее количество ресурсов (в том числе процессорного времени), чем необходимо в действительности. В работе [18] точечная оценка заменяется оценкой распределения времени выполнения заданий и приводятся результаты экспериментов, показавших повышение производительности базового алгоритма с использованием Backfill. При этом условие принятия решения о запуске задания заменяется условием «вероятность задержки момента запуска первого задания из очереди (вероятность ошибки) меньше заданного порога τ». Вероятность ошибки вычисляется при помощи методов динамического программирования каждый раз, когда необходимо принять решение о внеочередном запуске задания.

В диссертационной работе представлена математическая модель задачи и аналитическое выражение для оценки вероятности ошибки, предложенной в работе [18], в предположении, что распределение моментов завершения заданий имеет экспоненциальный вид, а количество освобождаемых в эти моменты процессоров подчиняется закону усеченного экспоненциального распределения или дискретного распределения Зипфа.

В третьей главе предлагается теоретико-игровая модель процесса управления потоком заданий в системе распределенных вычислений типа Desktop Grid. В системах такого рода в вычислительном процессе участвует множество лиц с различными интересами: администратор системы, пользователи, владельцы вычислительных узлов и др. Поэтому для решения задачи управления вычислительными ресурсами в распределенных системах находят применение ме-

тоды теории игр. При моделировании распределенных грид-систем в качестве игроков могут выступать пользователи [19–21], отдельные вычислительные узлы или их группы [22–25], вычислительные проекты [26].

Двухуровневая иерархическая игра [27, 28] позволяет смоделировать ситуацию, в которой участники делают свой выбор последовательно. Первым делает выбор игрок I уровня иерархии («центр»); затем игроки II уровня («подразделения») делают выбор, максимизируя свой выигрыш в условиях, ограниченных решением «центра». Зная правила, по которым «подразделения» будут отвечать на каждое решение, «центр» делает окончательный выбор, максимизируя свой выигрыш.

В системе Desktop Grid узлы имеют различные функции: сервер создает рабочие задания, распределяет их между клиентами и обрабатывает полученные от клиентов результаты. Клиенты, получив задания, занимаются непосредственно их выполнением. Таким образом, имеется иерархия узлов с двумя уровнями.

Интересы клиента и сервера являются не противоположными друг другу. Клиент стремится минимизировать свои расходы на проведение вычислений — например, плату за электричество. В зависимости от способа организации Desktop Grid, клиент может также быть заинтересован в минимизации или максимизации общего количества выполненной им работы. Сервер не занимается непосредственно вычислениями, и его выигрыш выражает некоторую характеристику работы, выполненной в совокупности всеми клиентами — например, общее время ожидания результатов.

В четвертой главе рассматривается задача обнаружения момента атаки типа «отказ в обслуживании» на вычислительную систему и последующего изменения протокола обслуживания входного потока заданий. Для обнаружения момента атаки используется метод кумулятивных сумм. Приводятся аналитические выражения характеристик метода для случая распределения Бернулли наблюдаемого параметра потока заданий. Предлагается протокол обслуживания, позволяющий отделять от общего входного потока задачи, созданные ата-

кующим.

Алгоритм обнаружения DoS-атаки с применением метода кумулятивных сумм чувствителен к небольшим изменениям в наблюдаемом процессе, а значит, потенциально эффективен для быстрого выявления DoS-атак с переменной интенсивностью. Метод кумулятивных сумм впервые был предложен в работе Е. Пейджа[29] для обнаружения момента разладки, то есть изменения среднего значения неотрицательных, независимых, одинаково распределенных случайных величин $\{x_n\}$, $n = 1, 2, ...$ Предполагается, что до момента разладки случайные величины подчиняются распределению $F(x, \alpha_0)$, а начиная с момента разладки $\theta \geq 0$ — распределению того же вида, но с измененным параметром $x_n \sim F(x, \alpha)$, где $\alpha \neq \alpha_0$.

Предположим, что каждое задание, поступающее в вычислительную систему, характеризуется численной величиной (количество запрашиваемых процессоров, ожидаемое время выполнения, идентификатор пользователя и др.). Если предположить, что данная характеристика задания — случайная величина, то регулярный и искусственный потоки будут иметь разные параметры распределений. Можно сделать вывод о наличии атаки, если зафиксировано изменение параметра распределения потока заданий. Если сигнал о наличии атаки подан в ее фактическое отсутствие, имеет место ложная тревога; характеристика процесса обнаружения ARL (от англ. Average Run Length) выражает среднее количество наблюдений до таковой. Характеристика алгоритма AD (от англ. Average Delay) выражает среднее количество наблюдений до подачи сигнала о фактически произошедшей атаке.

В диссертационной работе исследуется случай распределения Бернулли с параметром α_0, $0 < \alpha_0 < 1$. Выводятся аналитические выражения для характеристик процесса обнаружения разладки ARL и AD, полученные решением разностного уравнения (4.5) на стр. 71, описывающего динамику изменения среднего количества наблюдений до подачи сигнала о наличии атаки. Приводятся результаты моделирования входного потока заданий в вычислительной

системе, в котором наблюдается разладка, и предлагается новый протокол обслуживания после момента обнаружения, при котором из смешанного входного потока выделяются регулярные задания.

В заключении подведены итоги исследования и сформулированы основные выводы.

Общий объем диссертации составляет 112 страниц, включая 19 рисунков. Список литературы включает 109 наименований.

Глава 1

Обзор методов управления заданиями в высокопроизводительных вычислительных системах

Начиная с середины XX века, в ряде отраслей фундаментальной и прикладной науки стали возникать задачи, требующие для своего решения выполнения значительных объемов вычислений, равно как обработки, хранения, передачи и визуализации больших объемов данных. Подобные задачи связаны с моделированием в реальном времени процессов интенсивных физико-химических и ядерных реакций, экономического и промышленного развития регионов, глобальных атмосферных процессов и т.д., а также вычислениями и численным моделированием в области атомной энергетики, авиастроения, ракетно-космических технологий и прочих отраслей науки и техники. В монографии М. Гэри и Д. Джонсона 1982 г. [30] приведено множество примеров постановок задач, точные или приближенные решения которых накладывают высокие требования на производительность вычислительных ресурсов. Обзор подобных задач, решаемых в настоящее время, приводится в работах [31, 32]. В условиях недостаточной производительности обычных средств вычислительной техники, для решения таких задач используются значительно превосходящие их по техническим характеристикам высокопроизводительные вычислительные системы, которые можно условно разделить на три класса:

1. Суперкомпьютеры, архитектура и программное обеспечение которых разработаны для решения частного класса научных задач (например, вычислительно интенсивного прогнозирования погодно-климатических условий). В настоящей работе управление заданиями в этом классе устройств не рассматривается.

2. Вычислительные кластеры, представляющие собой множество стандартных многопроцессорных вычислительных узлов, объединенных высокоскоростными коммуникационными каналами в единую программно-аппаратную систему. Примером вычислительного кластера, на котором, в частности, апробировались результаты настоящей работы, является кластер КарНЦ РАН [33]: 10 вычислительных узлов, теоретическая пиковая производительность — 851 Гфлопс.

3. Системы распределенных вычислений. Системы этого класса объединяют территориально распределенные суперкомпьютеры, кластеры и другие вычислительные ресурсы, в том числе неспециализированные (персональные компьютеры, мобильные устройства, игровые приставки и др.). В частности, в настоящей работе рассмотрена модель управления заданиями в Desktop Grid — распределенной вычислительной системе с неспециализированными узлами.

Подробный обзор современных высокопроизводительных вычислительных систем приведен в работе Х. Хусейна и др. [34].

В диссертационной работе исследуются алгоритмы управления заданиями, оптимизирующие работу вычислительной системы по критериям, сформулированным с точки зрения ее владельца. В общем случае выбор принципов управления вычислительным процессом и критериев их эффективности во многом зависит не только от архитектуры вычислительной системы, но и от особенностей предоставления пользователям доступа к ней. В настоящее время активно развивается технология облачных вычислений — обеспечения сетевого доступа по требованию к высокопроизводительным вычислительным ресурсам общего пользования. Ресурсы для высокопроизводительной обработки, хранения и передачи информации аналогичным образом предоставляются в специализированных Центрах хранения и обработки данных. При управлении заданиями в таких системах наиболее актуальны задачи обеспечения пользователям безотказного

(с высокой вероятностью) доступа к ресурсам, «справедливого» распределения ресурсов между заданиями различных пользователей, возможности динамического изменения количества потребляемых ресурсов и др. При этом алгоритм управления заданиями, оптимальный с точки зрения пользователей, может не отвечать критериям оптимальности с точки зрения владельца вычислительной системы, и наоборот. Так, в работах [35, 36] показано, что стратегия управления заданиями, наиболее справедливая с точки зрения пользователей, может являться не оптимальной с точки зрения владельца системы по критерию средней загрузки вычислительных ресурсов.

1.1. Управление заданиями на вычислительном кластере

В Главе 2 рассматривается модель управления заданиями на вычислительном кластере, который представляет собой множество тесно связанных вычислительных узлов, имеющих единый центр управления и планирования заданий. Обобщенная структура системы изображена на Рис. 1.1. Вычислительные задания поступают на кластер из узлов-источников, а узел-исполнитель (логическое объединение множества вычислительных узлов) занимается непосредственно их выполнением.

Рис. 1.1. Структура вычислительного кластера.

С точки зрения пользователя кластер представляет собой единый вычислительный ресурс [37, 38]. Существует множество алгоритмов планирования,

которые направлены на оптимизацию задействования вычислительных ресурсов кластера по различным параметрам, а также учитывают различные характеристики заданий, которые могут быть как предопределенными, так и недетерминированными. Вопросам повышения производительности работы существующих алгоритмов планирования, сравнения и создания новых алгоритмов посвящен ряд исследований (например, [39–42] и др.). В работе А. Б. Новикова [40] приводится описание и сравнение алгоритмов планирования, учитывающих способность заданий к изменению объема требуемых ресурсов и времени выполнения заданий. С. Н. Мамойленко и А. В. Ефимов [43] сформулировали задачу распределения ресурсов распределенной многопроцессорной вычислительной системы как задачу многокритериальной оптимизации и предложили ряд алгоритмов планирования.

В целом алгоритмы планирования заданий на вычислительном кластере можно разделить на две группы в зависимости от принципа, лежащего в их основе [41, 42]. Указанные принципы не являются взаимоисключающими, и в ряде работ рассматриваются их комбинации [42, 44, 45]:

- Разделение времени (*time-sharing*): заданиям выделяются дискретные интервалы процессорного времени — слоты; по истечении очередного интервала времени выполнение задания приостанавливается до наступления следующего выделенного ему интервала. Оперативная память при этом может освобождаться, а ее содержимое записываться на жесткий диск. Такой тип планирования, как правило, целесообразно использовать для интерактивных заданий, требующих низкого времени ответа.

- Разделение пространства ресурсов (*space-sharing*): производится планирование одновременной работы нескольких заданий на нескольких центральных процессорах. Ресурсы, выделенные заданию для выполнения, находятся в его распоряжении до момента завершения, не участвуя при этом в выполнении других заданий. Как правило, такой тип планирования ис-

пользуется при обработке заданий в пакетном режиме, т.е. в большинстве современных кластерных систем.

- Комбинация принципов разделения времени и разделения пространства ресурсов используется в распространенном алгоритме управления группами заданий (*gang scheduling*).

Наиболее распространенными базовыми алгоритмами планирования заданий являются FCFS (от англ. First Come First Served — «первый поступивший обслуживается первым»), RR (Round Robin — назначение заданий по кругу), SJF (Shortest Job First — наиболее короткое задание обслуживается первым), LJF (Longest Job First — наиболее длинное задание обслуживается первым) и др. На их основе разрабатываются более сложные дисциплины обслуживания — в частности, с использованием системы приоритетов заданий, многоуровневых очередей и т.д.

Широко распространенной на практике и одной из наиболее эффективных по ряду критериев является комбинация базового алгоритма с процедурой Backfill. В зависимости от цели, преследуемой при составлении расписания, выделяются два варианта алгоритма [46]:

- *Aggressive Backfill* применяется, если требуется максимизировать задействование вычислительных ресурсов при невысоком среднем времени ожидания. Если для запуска очередного задания в настоящий момент недостаточно процессоров, то «резервируется» момент времени, в который должно освободиться необходимое количество процессоров для данного задания. Далее последовательно запускаются те задания в очереди, для которых свободных процессоров имеется достаточно, и которые, согласно оценке времени выполнения, не задержат запуск первого задания в «зарезервированный» момент времени.

- *Conservative Backfill* применяется, если требуется определять предел ожи-

дания в очереди для всех заданий, так чтобы большим заданиям не приходилось ждать неизвестное количество времени, «пропуская вперед» маленькие. В данном варианте алгоритма «резервируются» моменты времени для запуска всех заданий, ожидающих в очереди. Последующие задания запускаются вне очереди при условии, что они не задержат запуск ни одного предыдущего задания.

На практике эффективность того или иного варианта алгоритма зависит от свойств конкретного потока заданий. Однако в работе [47] показано, что, как правило, для многопроцессорных систем целесообразно использовать Aggressive Backfill, впервые реализованный в системе планирования заданий EASY (Extensible Argonne Scheduling SYstem) [48].

В обоих случаях для реализации алгоритма требуются предварительные оценки количества требуемых заданием ресурсов и времени выполнения, которые либо напрямую задаются пользователями при регистрации задания в системе, либо оцениваются на основе истории выполнения заданий. А. Ниссимов и Д. Фейтельсон в работе [18] впервые рассмотрели возможность предсказания распределения времени выполнения заданий вместо точечных оценок для алгоритма планирования заданий Backfill. В Главе 2 предлагается использовать параметры распределения, полученные анализом истории выполнения заданий, а принятие решения о применении процедуры Backfill осуществлять с использованием аналитического выражения вероятности ошибки.

1.2. Управление заданиями в системе распределенных вычислений

Системы распределенных вычислений применяются для решения крупных вычислительноемких задач, которые могут выполняться на большом количестве слабосвязанных (по сравнению с суперкомпьютерами и кластерами) вы-

числительных узлов. В обзорной статье Ванга и Морриса [49] систематизированы алгоритмы управления заданиями в системах распределенных вычислений и критерии их эффективности. В работе Касаванта и Кулля [50] классификация подходов к управлению заданиями в распределенной системе существенно расширена. Обзор алгоритмов планирования заданий в современных распределенных вычислительных системах приводится в работах [34, 51].

Обобщенная структура системы распределенных вычислений изображена на Рис. 1.2. Каждый узел-исполнитель может обладать собственной локальной подсистемой планирования заданий. Поскольку два множества узлов выполняют различные функции, критерии эффективности алгоритмов управления заданиями различаются в зависимости от того, интересами каких узлов они определяются. В частности, узлы-источники могут быть заинтересованы в минимизации времени выполнения своих заданий, а узлы-исполнители могут стремиться максимизировать свою среднюю загрузку, то есть по возможности избежать простаивания ресурсов.

Рис. 1.2. Структура распределенной вычислительной системы.

Выделяют три типа архитектуры подсистем планирования заданий в распределенных вычислительных системах:

- Централизованная: информация о доступных ресурсах и полномочия для их распределения между заданиями сосредоточены в единой подсистеме;

- Децентрализованная: имеется множество скоординированных равноправных подсистем планирования заданий;

- Иерархическая: имеется подсистема планирования первого уровня, координирующая работу множества подсистем второго уровня.

Первый тип архитектуры относительно прост, но тем не менее широко применяется на практике в настоящее время. В частности, на его основе создаются так называемые системы добровольных вычислений, задействующие в качестве узлов-исполнителей глобально распределенные компьютеры, добровольно предоставленные их владельцами. Вычислительные узлы в такой системе выполняют задания по мере наличия свободных ресурсов, не занятых выполнением задач владельца, а также могут выходить из состава системы и входить обратно в произвольные моменты времени. Отличительной особенностью систем добровольных вычислений является потенциальная возможность преднамеренного искажения узлами результатов вычислений. В связи с этим ряд работ [52–56] посвящен разработке и оценке алгоритмов планирования заданий, обеспечивающих достоверность получаемых результатов.

Второй и третий тип архитектуры применяются в тех случаях, когда узлы-исполнители являются автономными сложными вычислительными системами или иерархиями таких систем. Примером системы с иерархической архитектурой является Грид — объединение нескольких вычислительных кластеров или суперкомпьютеров, принадлежащих разным научно-исследовательским организациям. Тогда на уровне отдельного узла функционирует локальная подсистема управления заданиями, а на уровне групп узлов — подсистемы более высокого уровня, работа которых, в свою очередь, координируется глобально. Примером системы меньшего масштаба с децентрализованной архитектурой является одноранговая (peer-to-peer) вычислительная сеть из персональных компьютеров, которые обмениваются заданиями для минимизации среднеквадратичного отклонения загрузки узлов ([57, 58] и др.). Широкое распространение получили

также одноранговые сети хранения и обработки данных.

Подходы к управлению заданиями могут быть разделены на два класса в зависимости от типа узла, который является инициатором запуска заданий:

- Инициатором является узел-источник, если он принимает решение, на каком исполнителе запустить очередное задание. При этом на каждом из узлов-исполнителей формируется очередь заданий, поступивших от узлов-источников.

- Инициатором является узел-исполнитель, если он выбирает для запуска задания из доступных в системе. При этом на каждом из узлов-источников формируется очередь принадлежащих ему заданий, которые в дальнейшем будут обслуживаться узлами-исполнителями.

В Главе 3 рассматривается централизованная модель управления заданиями в разновидности системы добровольных вычислений, называемой Desktop Grid. Под Desktop Grid в данной работе понимается система, использующая в качестве узлов-исполнителей разнородные вычислительные ресурсы, принадлежащие организации или группе организаций, в том числе персональные настольные компьютеры, веб-серверы, узлы вычислительного кластера и т.д. Вычислительные узлы при этом связаны с управляющим центром сетью Интернет или локальной сетью передачи данных. Множество узлов-источников логически объединено в единый управляющий узел, который является инициатором запуска заданий. Узлы-исполнители ограниченны во взаимодействии или совсем не взаимодействуют друг с другом, в связи с чем вычислительные задания также должны быть слабосвязаны или полностью независимы. Обобщенная структура такой системы изображена на Рис. 1.3. Пример модели системы распределенных вычислений с централизованной архитектурой, но типом организации, отличным от Desktop Grid, рассматривается в работе [59], где узлы-исполнители полагаются однородными, а вычислительные задания сильносвязанными.

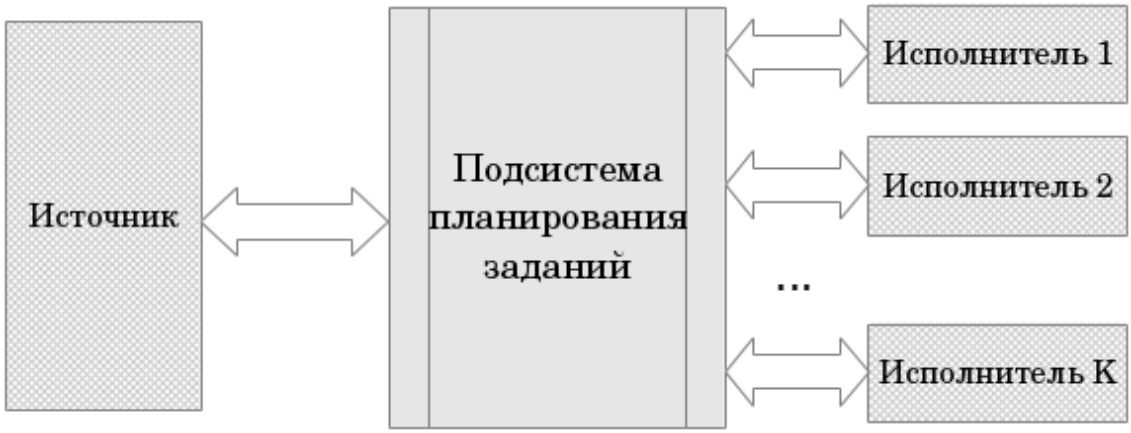

Рис. 1.3. Структура системы Desktop Grid.

При управлении заданиями в системе Desktop Grid необходимо учитывать ряд особенностей, связанных с принципами ее построения. Во-первых, вычислительные узлы не являются абсолютно надежными — выполнение задания может завершиться ошибкой в связи с программно-аппаратными особенностями узла или его временной недоступностью. В ряде работ [60–62] рассматривается эвристический алгоритм назначения множества независимых заданий на идентичные вычислительные узлы, выходящие из состава системы в случайные моменты времени, и приводятся точные аналитические оценки производительности алгоритма. В работе А. С. Румянцева [63] аналитически найдены условия целесообразности репликации идентичных заданий, выполняющихся на ненадежных узлах.

Во-вторых, вычислительные узлы Desktop Grid различны по программно-аппаратным характеристикам, скорости и пропускной способности сетевого соединения. В работах О. Ибарра [64] и Ч. Кима [65] исследуются оценки эффективности ряда эвристических алгоритмов для случая назначения независимых заданий на разнородные вычислительные узлы.

Как правило, эффективность конфигурации Desktop Grid оценивается по некоторой совокупности характеристик, в число которых зачастую входит общая длительность выполнения заданий [59, 60, 63, 65–67] и средняя пропускная способность системы [26, 68]. Однако на практике может оказаться целесооб-

разным учесть и другие критерии оптимизации работы системы, следующие из особенностей организации Desktop Grid или свойств вычислительного проекта. Например, для проектов добровольных вычислений может оказаться целесообразным максимизировать количество полезных вычислений, обеспечив при этом необходимую точность и достоверность результатов.

1.3. Обнаружение DoS-атаки на вычислительную систему

В Главе 4 рассматривается еще одна из задач, возникающих при управлении высокопроизводительной вычислительной системой, в частности, при обеспечении безопасности вычислительного процесса — обнаружение момента вторжения в систему и противодействие обнаруженной атаке.

Проведение DOS-атаки (от англ. Denial of Service), или более распространенной ее разновидности — распределенной DDOS-атаки (от англ. Distributed Denial of Service) на вычислительную систему заключается в том, что ко входному потоку заданий, генерируемому обычными пользователями (далее будем называть этот поток регулярным), добавляется искусственный поток заданий, поступающих с высокой интенсивностью и, возможно, требующих большого количества ресурсов. При отсутствии атак ресурсы системы, как правило, позволяют своевременно обслуживать все регулярные задания. Но при наличии большого количества искусственных заданий во входном потоке снижается эффективность обслуживания, поскольку помимо регулярных заданий приходится обслуживать и искусственные, поступающие с высокой интенсивностью и/или требующие значительного количества ресурсов. Вычислительная система не способна обрабатывать все поступающие задачи. В результате возрастает количество отказов в обслуживании, что и является целью атаки. Таким образом, возникает две задачи: зарегистрировать момент вторжения и изменить протокол обслуживания заданий так, чтобы продолжать обслуживать регулярные задачи в присутствии искусственного потока.

Подробный обзор существующих методов защиты от сетевых атак приводится в работе А. А. Кондратьева и др. [69], а также в работах А. В. Лукацкого [70], В. Зима [71] и других исследователей. Классификация DOS- и DDOS-атак и методов противодействия им приведена в обзорной статье [72]. В зависимости от интенсивности проведения, DDOS-атаки могут быть разделены на два класса:

1. При проведении атаки с *постоянной интенсивностью* злоумышленник имитирует поведение обычных пользователей, задействуя для этого тысячи территориально распределенных компьютеров. Такие атаки быстро достигают своей цели, но могут быть быстро обнаружены. Так, в работе [73] предлагается метод обнаружения DDOS-атаки как наличия большого количества источников запросов к системе, действующих одинаково (а значит, с большой вероятностью, автоматизированных для этого). Существуют и другие методы эффективного обнаружения атак данного класса (см., например, [74–79]).

2. При проведении атаки с *переменной интенсивностью* злоумышленник варьирует параметры искусственного потока заданий, достигая своей цели на протяжении более длительного интервала времени и затрудняя своевременное обнаружение атаки. Методы обнаружения атак данного класса представлены, например, в работах [80–86].

Для обнаружения DDOS-атак зачастую применяются методы математической статистики: например, спектральный анализ [87], ковариационный анализ [85], анализ временных рядов [88]. Метод кумулятивных сумм, впервые предложенный Е. С. Пейджем [29] чувствителен к небольшим изменениям в наблюдаемом случайном процессе, а значит, потенциально эффективен для быстрого выявления DDOS-атак с переменной интенсивностью. Примеры алгоритмов, использующих данный метод для обнаружения изменения характеристик сетевого трафика, приводятся в работах [87, 89, 90].

В работе А. Н. Ширяева [91] была показана оптимальность метода кумулятивных сумм в минимаксном смысле; в ряде работ были получены приближенные и аналитические выражения среднего количества наблюдений до ложной тревоги и до обнаружения разладки: в частности, численное решение для случая нормального распределения [92], численные и аналитические решения для случая экспоненциального распределения [93–95], для распределений с «тяжелыми хвостами» [96].

Глава 2

Оценка характеристик алгоритма Backfill при управлении потоком заданий на вычислительном кластере

2.1. Постановка задачи

Исходный вариант алгоритма Backfill был впервые описан в работе [48], в рамках системы планирования заданий на суперкомпьютере IBM SP1 Аргонской Национальной лаборатории (США). В ходе работы алгоритма обрабатывается очередь заданий, ожидающих запуска на выполнение. Задания в очереди упорядочены по времени их регистрации в системе. При каждой регистрации и каждом завершении задания происходит выполнение следующих шагов алгоритма:

1. пока имеется необходимое количество свободных процессоров для запуска первого задания в очереди, удалять задание из очереди и запускать на выполнение;

2. если для запуска первого задания в очереди недостаточно свободных процессоров, вычислить момент времени, когда процессоров станет достаточно, и зарезервировать их для данного задания;

3. продолжать двигаться по очереди, запуская на выполнение задания, если они не нарушают резервирование первого задания в очереди.

Таким образом, задания, требующие небольшое количество процессоров или процессорного времени, могут запускаться на выполнение вне очереди при наличии необходимого количества доступных ресурсов. При этом их запуск не должен задерживать выполнение предыдущих по очереди заданий (Рис. 2.1).

В результате удается задействовать вычислительные ресурсы, простаивающие во время выполнения первоочередных заданий.

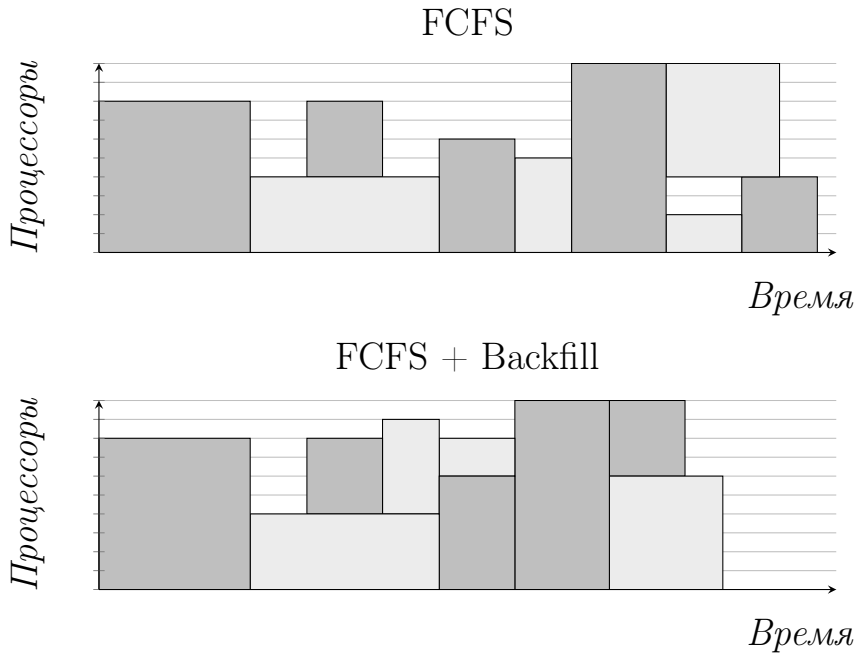

Рис. 2.1. Управление очередью заданий с использованием алгоритма Backfill.

В зависимости от политики регистрации заданий на кластере и от поведения пользователей, оценки длительности могут отсутствовать или быть неточными, вследствие чего при применении процедуры Backfill неизбежно возникают ошибки. Это приводит к задержкам запуска заданий или их принудительным прерываниям. Согласно модификации алгоритма, предложенной в работе А. Ниссимова [18], условие принятия решения о запуске задания, осуществляемого на шаге 2 и 3, заменяется условием «вероятность задержки момента запуска первого задания из очереди (вероятность ошибки) меньше заданного порога τ». Выразим вероятность ошибки аналитически и исследуем характеристики модифицированного алгоритма.

Рассмотрим математическую модель вычислительного кластера с M процессорами. Пусть в настоящий момент времени свободно $0 \leq c_0 \leq M$ процессоров, и пусть в очереди находится N заданий J_1, J_2, ..., J_N, упорядоченных по времени их регистрации в системе. Предположим, что для запуска первого задания в очереди J_1 требуется c_q свободных процессоров и $c_q > c_0$, то есть

свободных процессоров для ее запуска в настоящий момент недостаточно. Согласно алгоритму Backfill, необходимо принять решение о запуске некоторого задания в очереди $J_i, 1 < i \leqslant N$, для которого имеется достаточно свободных процессоров: $c_i \leqslant c_0$. Пусть t_e — ожидаемое время выполнения задания J_i. Если в течение интервала времени длиной t_e освободится число процессоров, достаточное для запуска задания J_1, то немедленный запуск J_i может задержать запуск J_1. Решение запустить задание J_i принимается, если вероятность этого события меньше заданного порога τ.

2.2. Аналитическое выражение вероятности ошибки

Найдем аналитическое выражение вероятности задержки запуска задания J_1 в случае принятия решения о немедленном запуске задания J_i. Будем считать, что как только некоторое задание завершает работу, освобождаются занимаемые им ресурсы кластера. Будем называть освобождение ресурсов скачком. Число свободных процессоров в дискретный момент времени $t_i, i \geqslant 0$, описывается случайным процессом

$$S(t_i) = S(t_{i-1}) + \xi_i,$$

где ξ_i — число процессоров, освобожденных в момент i-го скачка, $S(0) = S_0 = 0$. Предположим, что величины $\xi_1, \xi_2, \ldots$ имеют усеченное экспоненциальное распределение с параметром μ, принимая значения от единицы до M, то есть плотность распределения имеет вид

$$f_{\xi_1}(x) = \frac{\mu e^{-\mu x}}{e^{-\mu L} - e^{-\mu H}}, \text{ где } \mu > 0,\ L = 1,\ H = M. \tag{2.1}$$

Предположим также, что моменты скачков (завершения заданий) распре-

делены экспоненциально с параметром λ,

$$f_{t_1}(x) = \lambda e^{-\lambda x},\ \lambda > 0.$$

Таким образом, необходимо найти аналитическое выражение

$$P(\exists t_i : c_q \leqslant S(t_i) < c,\ c = c_q + c_i), \tag{2.2}$$

вероятности существования момента скачка, в который число свободных процессоров окажется достаточным для запуска первого задания из очереди J_1, но недостаточным для одновременной работы J_1 и рассматриваемого задания J_i (Рис. 2.2).

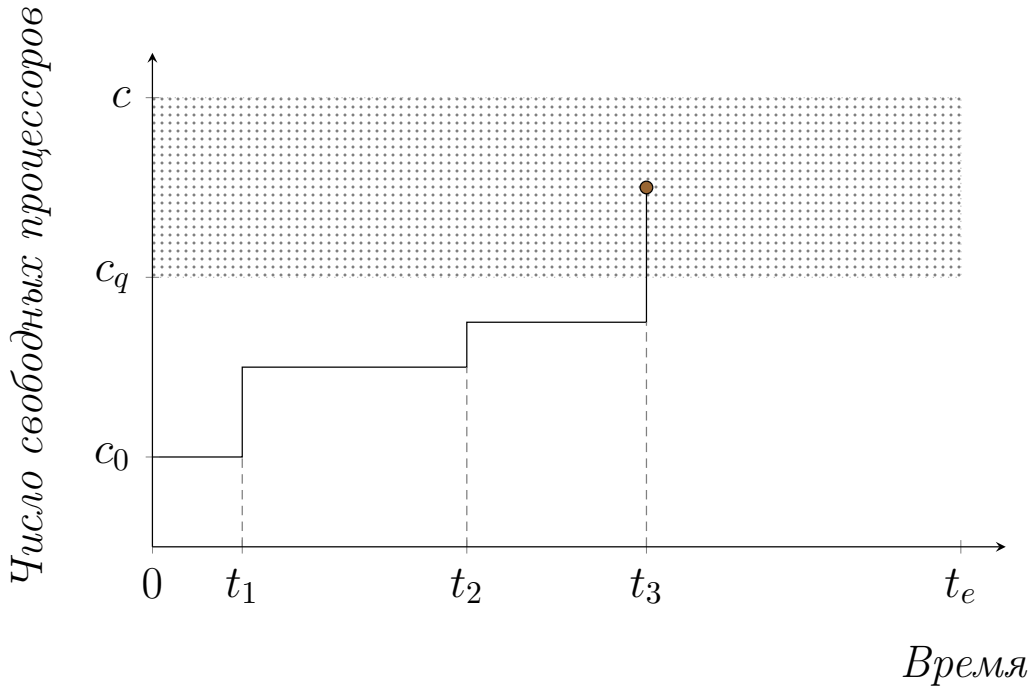

Рис. 2.2. Задержка запуска первого задания в очереди.

В предположении, что число скачков N на интервале $[0; t_e]$ имеет пуассоновское распределение с параметром λt_e, вероятность существования момента скачка, в который число свободных процессоров окажется достаточным для запуска J_1, но недостаточным для одновременной работы J_1 и J_i, может быть

выражена следующим образом:

$$P(\exists t_i : c_q \leqslant S(t_i) < c) =$$
$$= \sum_{k=1}^{\infty} P(N = k) \sum_{n=1}^{k} P(S_{n-1} < c_q, c_q \leqslant S_n < c) =$$
$$= \sum_{k=1}^{\infty} \frac{(\lambda t_e)^k}{k!} e^{-\lambda t_e} \sum_{n=1}^{k} P(S_{n-1} < c_q, c_q \leqslant S_n < c)$$

Вероятность $P(S_{n-1} < c_q, c_q \leqslant S_n < c)$, того, что после $(n-1)$-го скачка свободных процессоров недостаточно для запуска J_1, а после n-го скачка свободных процессоров станет достаточно для запуска J_1, но недостаточно для одновременной работы J_1 и J_i, может быть вычислена рекурсивно.

Вероятность того, что в начальный момент времени свободных процессоров недостаточно для запуска J_1, а после первого скачка свободных процессоров станет достаточно для запуска J_1, но недостаточно для одновременной работы J_1 и J_i:

$$P(S_0 < c_q, c_q \leqslant S_1 < c) = P(c_q \leqslant S_1 < c) = P(c_q \leqslant \xi_1 < c) =$$
$$= F_{\xi_1}(c) - F_{\xi_1}(c_q) = e^{-\mu} - e^{-\mu c} - e^{-\mu} + e^{-\mu c_q} = e^{-\mu c_q} - e^{-\mu c}$$

Вероятность того, что после первого скачка свободных процессоров недостаточно для запуска J_1, а после второго скачка свободных процессоров станет

достаточно для запуска J_1, но недостаточно для одновременной работы J_1 и J_i:

$$P(S_1 < c_q, c_q \leqslant S_2 < c) = P(\xi_1 < c_q, c_q \leqslant \xi_1 + \xi_2 < c) =$$
$$= \int_0^{c_q} f_{\xi_1}(x_1) \int_{c_q - x_1}^{c - x_1} f_{\xi_2}(x_2)\, dx_2\, dx_1 =$$
$$= \int_0^{c_q} \frac{\mu e^{-\mu x_1}}{e^{-\mu} - e^{-\mu M}} \int_{c_q - x_1}^{c - x_1} \frac{\mu e^{-\mu x_2}}{e^{-\mu} - e^{-\mu M}}\, dx_2\, dx_1 =$$
$$= \frac{\mu \left(e^{-\mu c_q} - e^{-\mu c}\right)}{(e^{-\mu} - e^{-\mu M})^2} \int_0^{c_q} dx_1 = \frac{\mu \left(e^{-\mu c_q} - e^{-\mu c}\right) c_q}{(e^{-\mu} - e^{-\mu M})^2}$$

Вероятность того, что после второго скачка свободных процессоров недостаточно для запуска J_1, а после третьего скачка свободных процессоров станет достаточно для запуска J_1, но недостаточно для одновременной работы J_1 и J_i:

$$P(S_2 < c_q, c_q \leqslant S_3 < c) = P(\xi_1 < c_q, \xi_1 + \xi_2 < c_q, c_q \leqslant \xi_1 + \xi_2 + \xi_3 < c) =$$
$$= \int_0^{c_q} f_{\xi_1}(x_1) \int_0^{c_q - x_1} f_{\xi_2}(x_2) \int_{c_q - x_1 - x_2}^{c - x_1 - x_2} f_{\xi_3}(x_3) dx_3 dx_2 dx_1 =$$
$$= \frac{\mu^2 c_q^2}{2(e^{-\mu} - e^{-\mu M})^3} \left(e^{-\mu c_q} - e^{-\mu c}\right)$$

Продолжая те же рассуждения, получим

$$P(S_{n-1} < c_q, c_q \leqslant S_n < c) = \frac{(\mu c_q)^{n-1}}{(e^{-\mu} - e^{-\mu M})^n (n-1)!} \left(e^{-\mu c_q} - e^{-\mu c}\right) \qquad (2.3)$$

Следовательно, аналитическое выражение вероятности (2.2) имеет следу-

ющий вид:

$$P(\exists t_i : c_q \leqslant S(t_i) < c) =$$

$$= \sum_{k=1}^{\infty} \frac{(\lambda t_e)^k}{k!} e^{-\lambda t_e} \sum_{n=1}^{k} \frac{(\mu c_q)^{n-1}}{(n-1)!} \frac{e^{-\mu c_q} - e^{-\mu c}}{(e^{-\mu} - e^{-\mu M})^n} =$$

$$= \sum_{k=1}^{\infty} \left(1 - e^{-\lambda t_e}\right) \sum_{n=1}^{k} \frac{(\mu c_q)^{n-1}}{(n-1)!} \frac{e^{-\mu c_q} - e^{-\mu c}}{(e^{-\mu} - e^{-\mu M})^n} =$$

$$= \left(1 - e^{-\lambda t_e}\right) \left(e^{-\mu c_q} - e^{-\mu c}\right) \sum_{n=1}^{\infty} \sum_{k=n}^{\infty} \frac{(\mu c_q)^{n-1}}{(n-1)!(e^{-\mu} - e^{-\mu M})^n} \quad (2.4)$$

Предположение экспоненциальности позволяет аналитически найти основные характеристики процесса вычислений для последующего применения алгоритма Backfill. Отметим, что такая модель не в полной мере отражает реальные характеристики вычислительного кластера, что проиллюстрировал статистический анализ истории выполнения заданий на вычислительном кластере Карельского научного центра РАН [33] в 2012 г. По критерию Колмогорова-Смирнова статистический уровень значимости каждой из гипотез об экспоненциальном распределении моделируемых случайных величин (количество требуемых процессоров, длительность выполнения и время между приходами заданий) не превысил $2{\times}10^{-16}$. Следовательно, каждую из трех гипотез следует отвергнуть при любом практически используемом уровне значимости.

В работе [97] показано, что реальное распределение времени выполнения заданий (на примере истории использования кластера [33] в 2009-2011 гг.) хорошо приближается усеченным распределением Парето с плотностью

$$f(x) = \frac{\alpha L^\alpha}{1 - (\frac{L}{H})^\alpha} x^{-\alpha-1}, \ \alpha > 0, \ 0 < L \leqslant x \leqslant H, \quad (2.5)$$

а количество требуемых процессоров N — дискретным распределением Зипфа

$$P(N = i) = \frac{i^{-\alpha}}{\sum_{k=1}^{H} k^{-\alpha}}, \ \alpha > 0, \ 1 \leqslant i \leqslant H. \quad (2.6)$$

Однако в этом случае сложность составляет определение параметров распределения уходов задач (скачков).

В работе [98] приводятся результаты исследования, показавшего, что даже весьма простые модели потоков заданий, как правило, позволяют получить практически значимые выводы об эффективности алгоритмов планирования. Относительно простая модель потока заданий, сформулированная в данной главе, позволяет оценить эффективность предлагаемого алгоритма благодаря возможности проведения множества экспериментов с различными значениями параметров модели, видами и значениями параметров распределений моделируемых случайных величин. В качестве примера модификации модели приведем аналитическое выражение вероятности (2.2) в предположении, что величины ξ_1, ξ_2, ... имеют дискретное распределение Зипфа (2.6) с параметром $\mu > 0$, принимая значения от единицы до M, то есть

$$P(\xi_1 = i) = \sigma i^{-\mu} \text{ , где } \sigma = \left(\sum_{k=1}^{M} k^{-\mu} \right)^{-1} .$$

Как и для случая экспоненциального распределения, вычислим вероятность $P(S_{n-1} < c_q, c_q \leqslant S_n < c)$ рекурсивно.

$$P(c_q \leqslant \xi_1 < c) = \sum_{i=c_q}^{c-1} P(\xi_1 = i) = \sigma \sum_{i=c_q}^{c-1} i^{-\mu}$$

$$P(\xi_1 < c_q, c_q \leqslant \xi_1 + \xi_2 < c) = \sum_{i=1}^{c_q-1} P(\xi_1 = i) \sum_{j=c_q-i}^{c-i-1} P(\xi_2 = j) = \sigma^2 \sum_{i=1}^{c_q-1} \sum_{j=c_q-i}^{c-i-1} (ij)^{-\mu}$$

$$P(\xi_1 < c_q, \xi_1 + \xi_2 < c_q, c_q \leqslant \xi_1 + \xi_2 + \xi_3 < c) = \sigma^3 \sum_{i=1}^{c_q-1} \sum_{j=1}^{c_q-i-1} \sum_{k=c_q-i-j}^{c-i-j-1} (ijk)^{-\mu}$$

Продолжая аналогично, получаем выражение

$$P(S_{n-1} < c_q, c_q \leqslant S_n < c) = \\ = \sigma^n \sum_{i_1=1}^{c_q-1} \sum_{i_2=1}^{c_q-i_1-1} \cdots \sum_{i_{n-1}=1}^{c_q-i_1-\ldots-i_{n-2}-1} \sum_{i_n=c_q-i_1-\ldots-i_{n-1}}^{c-i_1-\ldots-i_{n-1}-1} \left(\prod_{j=1}^{n} i_j \right)^{-\mu}$$

Аналитическое выражение вероятности (2.2) принимает следующий вид:

$$P(\exists t_i : c_q \leqslant S(t_i) < c) = \sum_{k=1}^{\infty} \frac{(\lambda t_e)^k}{k!} e^{-\lambda t_e} \times \\ \times \sum_{n=1}^{k} \left(\sigma^n \sum_{i_1=1}^{c_q-1} \sum_{i_2=1}^{c_q-i_1-1} \cdots \sum_{i_{n-1}=1}^{c_q-i_1-\ldots-i_{n-2}-1} \sum_{i_n=c_q-i_1-\ldots-i_{n-1}}^{c-i_1-\ldots-i_{n-1}-1} \left(\prod_{j=1}^{n} i_j \right)^{-\mu} \right) \quad (2.7)$$

2.3. Вычислительные эксперименты

2.3.1. Имитационная модель

Для исследования характеристик модифицированного алгоритма Backfill была реализована имитационная модель управления потоком заданий на вычислительном кластере с 80 процессорами. В качестве базовой дисциплины обслуживания была принята FCFS. Значения параметров имитационной модели задавались, исходя из значений следующих величин, полученных статистическим анализом истории выполнения заданий на вычислительном кластере Карельского научного центра РАН:

- $1/\mu = 9.53$ — среднее запрашиваемое количество процессоров,
- $1/\lambda_a = 105.93$ мин. — средний интервал времени между приходами,
- $1/z = 208.33$ мин. — среднее время выполнения заданий.

При моделировании параметры распределений устанавливались следующим образом при заданных средних значениях.

- Усеченное экспоненциальное распределение (2.1): математическое ожидание длительности выполнения задания

$$\frac{e^{-\alpha L}(1+\alpha L) - e^{-\alpha H}(1+\alpha H)}{\alpha} = \frac{1}{z} = 208.33$$

при $L = 1$ и $H = 4320$, отсюда приближенное значение параметра $\alpha = 0.00480002$. Аналогично, параметр распределения времени между приходами заданий $\alpha = 0.00943978$.

- Усеченное распределение Парето (2.5): математическое ожидание длительности выполнения задания

$$\frac{L^\alpha}{1-(\frac{L}{H})^\alpha}\frac{\alpha}{\alpha-1}\left(\frac{1}{L^{\alpha-1}} - \frac{1}{H^{\alpha-1}}\right) = \frac{1}{z} = 208.33$$

при $L = 1$ и $H = 4320$, отсюда приближенное значение параметра $\alpha = 0.24296$. Аналогично, параметр распределения времени между приходами заданий $\alpha = 0.396389$.

- Дискретное распределение Зипфа (2.6): математическое ожидание количества запрашиваемых процессоров

$$\sum_{i=1}^{M} i\frac{i^{-\alpha}}{\sum_{k=1}^{M} k^{-\alpha}} = \frac{1}{\mu} = 9.53$$

при $M = 64$, отсюда приближенное значение параметра $\alpha = 1.23295$.

Вычисления значений параметров производились с использованием прикладного пакета Wolfram Mathematica©. Имитационная модель потока заданий и предложенный в работе алгоритм управления заданиями были реализованы на языке C++. Для аппроксимации с высокой точностью значений бесконечных сумм в выражении для P была использована библиотека GSL (GNU Scientific Library). Погрешность при вычислении приближенного значения внешней сум-

мы с использованием функций GSL не превышала 10^{-15}.

В качестве порогового значения вероятности ошибки было принято значение $\tau = 0.05$. При проведении вычислительных экспериментов мы полагали, что такое значение соответствует приемлемому на практике количеству задержек первоочередных заданий. Предполагалось также, что при регистрации задания в системе была известна точная оценка его времени выполнения.

2.3.2. Производительность алгоритма при различных видах распределений моделируемых величин

Одно и то же множество заданий, смоделированных с указанными входными параметрами, обрабатывалось сначала с использованием базовой дисциплины обслуживания, в данном случае FCFS (первое пришедшее задание обслуживается первым). По результатам моделирования вычислялась средняя длительность интервалов времени между уходами заданий для последующего использования при вычислении вероятности ошибки. Затем множество заданий обрабатывалось вновь с использованием процедуры Backfill. Было вычислено среднее время ожидания задания в очереди, количество применений процедуры Backfill и количество ошибок (задержек первоочередных заданий) при ее применении.

В Таблице 2.1 представлены результаты экспериментов, проведенных на множествах размером 2 000 заданий, смоделированных с различными распределениями при одинаковых средних значениях соответствующих величин. В целом модифицированная процедура Backfill позволила сократить среднее время ожидания задания в очереди, доля ошибок при этом не превышала приемлемых на практике значений. Эксперименты показали, что вид распределений моделируемых случайных величин даже при их равных средних значениях существенно влияет на эффективность работы алгоритма. Отметим, что на модели потока заданий при распределениях случайных величин, предложенных в [97], алгоритм показывает наилучшую эффективность как по критерию снижения

времени ожидания, так и по количеству ошибок.

В Таблице 2.2 представлены результаты экспериментов, проведенных с неточными (случайными) оценками времени выполнения. При моделировании каждого задания время его выполнения описывалось парой независимых одинаково распределенных случайных величин: первая из них определяла истинное время выполнения, а вторая — время, указанное пользователем. Результаты экспериментов, представленных в Таблицах 2.1 и 2.2, позволяют сделать выводы о работоспособности алгоритма в двух «экстремальных» ситуациях: наличие точной информации о времени выполнения задания и наличие информации лишь о его распределении. На практике, как правило, ни одна из данных ситуаций идеальным образом не реализуется, и имеет место некоторая их комбинация.

2.3.3. Производительность алгоритма при различных значениях параметров модели

На втором этапе экспериментов изменялись значения входных параметров, и описанные выше эксперименты повторялись для исследования влияния входных параметров на производительность алгоритма.

На Рис. 2.3—2.6 приведены результаты серии экспериментов. Использованы следующие обозначения:

T_{wait}^{FCFS} — среднее время ожидания задания в очереди при дисциплине обслуживания FCFS,

$T_{wait}^{Backfill}$ — среднее время ожидания задания в очереди с применением процедуры Backfill,

$\Delta T_{wait} = \dfrac{T_{wait}^{FCFS} - T_{wait}^{Backfill}}{T_{wait}^{FCFS}}$ — относительное изменение среднего времени ожидания задания в очереди,

$\bar{t}_e$ — среднее время выполнения задания,

$\tilde{p}$ — средняя доля процессоров кластера, занимаемых заданием.

Эксперименты показали, что в целом использование алгоритма планиро-

вания Backfill с принятием решения на основе вероятности ошибки позволяет снизить среднее время ожидания заданий в очереди по сравнению с алгоритмом планирования FCFS (Рис. 2.3, 2.4). Выигрыш в среднем времени ожидания имеет прямую зависимость как от среднего количества процессоров, требуемых заданием $\tilde{p}$, так и от среднего времени выполнения задания $\overline{t}_e$.

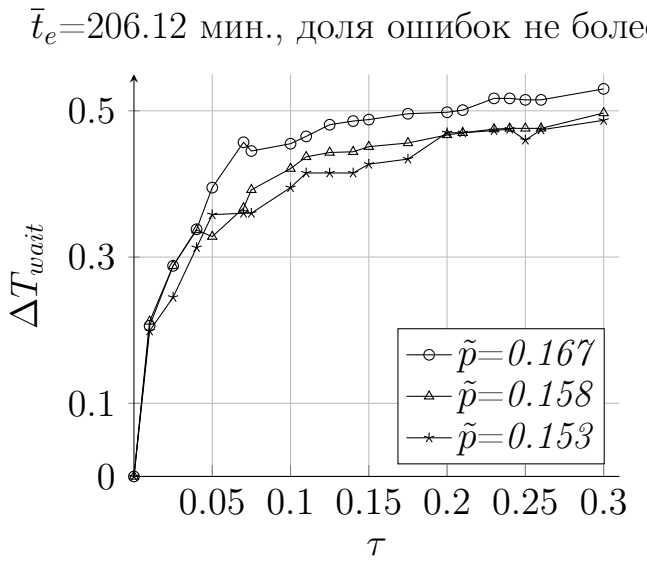

Рис. 2.3. Изменение среднего времени ожидания в зависимости от порогового значения τ при фиксированной средней длительности выполнения задания.

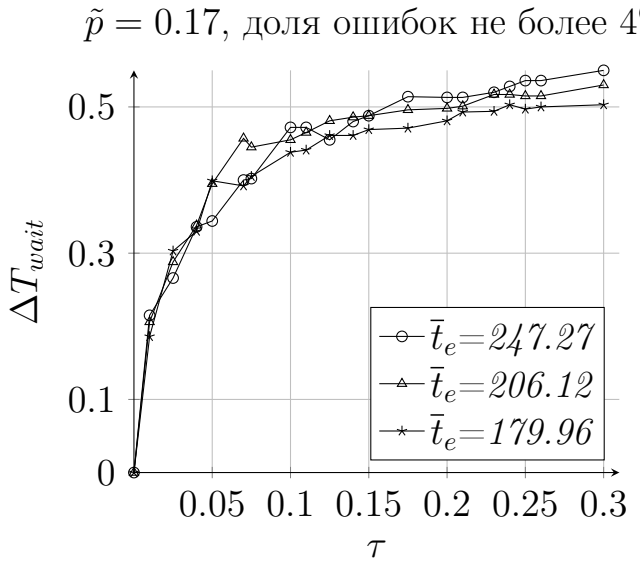

Рис. 2.4. Изменение среднего времени ожидания в зависимости от порогового значения τ при фиксированном среднем количестве занятых процессоров.

Таблица 2.1. Результаты работы алгоритма на множествах заданий, смоделированных с различными распределениями (пороговое значение вероятности ошибки при принятии решения $\tau = 0.05$).

Вид распределения случайной величины			ΔT_w	Доля применений Backfill	Доля ошибок
Интервал между приходами заданий	Продолжительность выполнения задания	Количество процессоров			
Экспоненциальное	Экспоненциальное	Усеченное экспоненциальное	0.285	0.027	0.009
Усеченное экспоненциальное	Усеченное экспоненциальное	Усеченное экспоненциальное	0.282	0.027	0.009
Усеченное Парето	Усеченное Парето	Усеченное экспоненциальное	0.643	0.195	0.021
Экспоненциальное	Экспоненциальное	Равномерное дискретное	0.049	0.002	0.001
Усеченное экспоненциальное	Усеченное экспоненциальное	Равномерное дискретное	0.046	0.002	0.001
Усеченное Парето	Усеченное Парето	Равномерное дискретное	0.459	0.063	0.015
Экспоненциальное	Экспоненциальное	Распределение Зипфа	0.521	0.099	0.025
Усеченное экспоненциальное	Усеченное экспоненциальное	Распределение Зипфа	0.518	0.095	0.021
Усеченное Парето	Усеченное Парето	Распределение Зипфа	0.753	0.264	0.016

Таблица 2.2. Результаты работы алгоритма на множествах заданий, смоделированных с различными распределениями (пороговое значение вероятности ошибки при принятии решения $\tau = 0.05$) со случайными оценками времени выполнения.

Вид распределения случайной величины			ΔT_w	Доля применений Backfill	Доля ошибок
Интервал между приходами заданий	Продолжительность выполнения задания	Количество процессоров			
Экспоненциальное	Экспоненциальное	Усеченное экспоненциальное	0.125	0.032	0.017
Усеченное экспоненциальное	Усеченное экспоненциальное	Усеченное экспоненциальное	0.110	0.031	0.020
Усеченное Парето	Усеченное Парето	Усеченное экспоненциальное	0.602	0.290	0.065
Экспоненциальное	Экспоненциальное	Равномерное дискретное	0.049	0.002	0.003
Усеченное экспоненциальное	Усеченное экспоненциальное	Равномерное дискретное	0.048	0.002	0.003
Усеченное Парето	Усеченное Парето	Равномерное дискретное	0.385	0.110	0.050
Экспоненциальное	Экспоненциальное	Распределение Зипфа	0.370	0.106	0.043
Усеченное экспоненциальное	Усеченное экспоненциальное	Распределение Зипфа	0.368	0.104	0.043
Усеченное Парето	Усеченное Парето	Распределение Зипфа	0.726	0.365	0.048

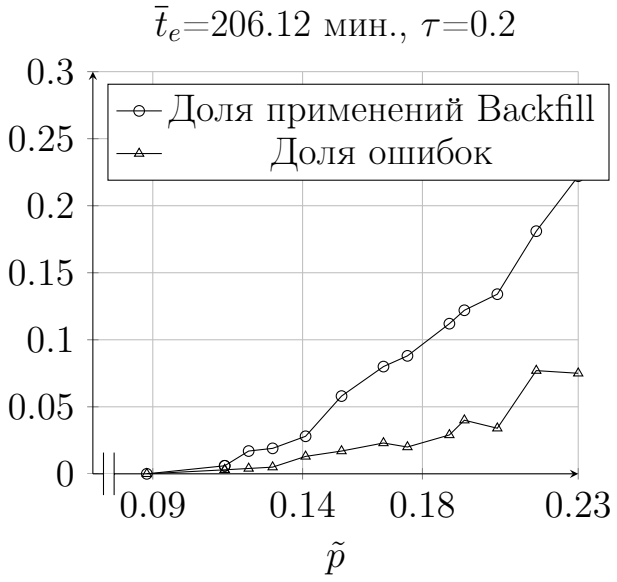

Рис. 2.5. Доля применений Backfill и доля ошибок в зависимости от среднего числа процессоров при фиксированной средней длительности выполнения задания.

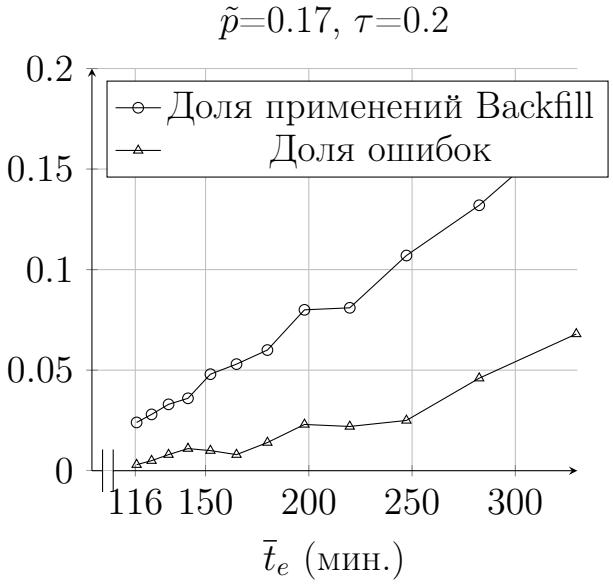

Рис. 2.6. Доля применений Backfill и доля ошибок в зависимости от среднего времени выполнения задания при фиксированном среднем количестве занятых процессоров.

Нужно отметить, что с увеличением среднего количества процессоров или среднего времени выполнения задания (а следовательно, с увеличением доли относительно «больших» заданий) возрастает и количество ошибок: на Рис. 2.5, 2.6 представлены зависимости долей успешных и ошибочных применений Backfill от соответствующих характеристик вычислительного процесса. Зависимость до-

ли ошибок от размера заданий ограничивает возможность применения алгоритма, поскольку при улучшении в среднем времени ожидания неизбежно увеличивается количество ошибок.

2.3.4. Производительность алгоритма на моделях потоков заданий, восстановленных из истории функционирования вычислительных систем

На следующем этапе экспериментов эффективность алгоритма исследовалась на имитационной модели, по задействованным характеристикам полностью повторяющей входной поток заданий на вычислительном кластере. Поток заданий был восстановлен из регистрационных файлов в формате SWF [99].

Согласно сформулированной математической модели, интерес представляли входные потоки заданий с известным временем регистрации в системе, выполняемых как в однопроцессорном, так и в многопроцессорном режимах, с единой очередью заданий. Обработанные регистрационные файлы были получены из ВК ЦКП КарНЦ РАН и открытого Интернет-архива[1]. Для моделирования были использованы регистрационные файлы следующих систем (Рис. 2.7):

1. **KRC** — 80-процессорный вычислительный кластер КарНЦ РАН с использованием системы управления заданиями Cleo 5.22[2], которая эксплуатировалась в период с 03.06.2009 г. по 04.02.2011 г. В течение рассматриваемого периода на кластере были произведены расчеты 8 282 заданий с длительностью выполнения, ограниченной 3 сутками согласно политике использования кластера. Среднее время выполнения задания составило 208.4 минуты, среднее количество использованных процессоров — 9.57. Поскольку регистрационный файл не содержал информации о пользовательских оценках времени выполнения заданий, в имитационной модели

[1] Parallel Workload Archive. URL: http://www.cs.huji.ac.il/labs/parallel/workload

[2] Система управления заданиями Cleo // Лаборатория параллельных информационных технологий НИВЦ МГУ. URL: http://parcon.parallel.ru/cleo.html

были использованы точные апостериорные оценки длительности.

2. **DAS2** — 64-процессорный вычислительный кластер университета г. Амстердам в составе суперкомпьютера «Distributed ASCI Supercomputer-2»[3] с использованием системы управления Maui [100] в период с 01.01.2003 г. по 31.12.2003 г. В течение рассматриваемого периода на кластере были произведены расчеты 66 429 заданий с максимальной длительностью выполнения 37.5 суток. Среднее время выполнения задания составило 20.07 минут, среднее количество использованных процессоров — 9.54. В имитационной модели были использованы оценки длительности выполнения, указанные пользователями при регистрации заданий.

3. **DAS4** — 64-процессорный вычислительный кластер университета г. Утрехт в составе суперкомпьютера «Distributed ASCI Supercomputer-2»[3] с использованием системы управления Maui в период с 01.01.2003 г. по 31.12.2003 г. В течение рассматриваемого периода на кластере были произведены расчеты 33 695 заданий с максимальной длительностью выполнения 25.1 суток. Среднее время выполнения задания составило 45.94 минуты, среднее количество использованных процессоров — 3.55. В имитационной модели были использованы оценки длительности выполнения, указанные пользователями при регистрации заданий.

4. **KTH** — 100-процессорный вычислительный кластер IBM SP2 шведского королевского технологического института[4] в период с 01.10.1996 г. по 01.08.1997 г. В течение рассматриваемого периода на кластере были произведены расчеты 28 490 заданий с максимальной длительностью выполнения 2.6 суток. Среднее время выполнения задания составило 147.54 минут, среднее количество использованных процессоров — 7.67. В имитационной

[3] The Distributed ASCI Supercomputer 2 (DAS-2) // Department of Computer Science, Faculty of Sciences, Vrije Universiteit Amsterdam. URL: http://www.cs.vu.nl/das2

[4] PDC Center for High Performance Computing. URL: https://www.pdc.kth.se

модели были использованы оценки длительности выполнения, указанные пользователями при регистрации заданий.

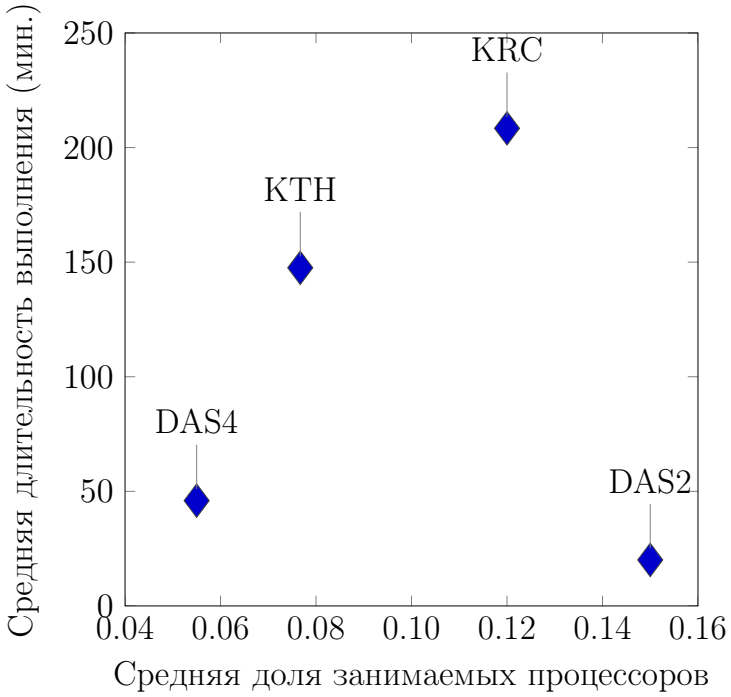

Рис. 2.7. Средние размеры вычислительных заданий в четырех рассматриваемых вычислительных системах.

Результаты моделирования при заданном пороговом значении $\tau = 0.05$ приведены в Таблице 2.3. Количество успешных применений процедуры Backfill и количество ошибок соответствуют зависимостям от параметров потока заданий, проиллюстрированным в предыдущем разделе (Рис. 2.5).

Система	ΔT_w	Доля применений Backfill	Доля ошибок
KRC	0.204	0.013	0.001
DAS2	0.540	0.004	0.0004
DAS4	0.525	0.017	0.001
KTH	0.678	0.534	0.144

Таблица 2.3. Результаты работы алгоритма на множествах заданий, смоделированных по истории функционирования четырех вычислительных систем.

В качестве критерия эффективности алгоритма также может быть рассмотрена эмпирическая функция распределения времени ожидания, которая

позволяет сделать вывод об изменении производительности вычислительной системы в целом. На Рис. 2.8 изображены эмпирические функции распределения времени ожидания, вычисленные на одном и том же входном потоке заданий для трех вариантов алгоритмов планирования: базового FCFS, модифицированного Backfill и использованного на практике в вычислительной системе. Данный сравнительный анализ произведен по регистрационному файлу системы DAS2, в которой на практике использовался вариант алгоритма EASY Backfill при базовой дисциплине обслуживания FCFS.

Полученные зависимости иллюстрируют, что при использовании модифицированного алгоритма Backfill доля задач со временем ожидания не более 8.3 минут на 2% больше, со временем ожидания не более 52 минут — на 1% больше, а со временем ожидания не более 1.5 часов — на 0.4% меньше, чем при практически реализованной дисциплине обслуживания. Этот результат можно интерпретировать следующим образом. Алгоритм, реализованный в системе управления заданиями на практике, при принятии решения о применении процедуры Backfill использует некоторую комбинацию параметров (например, идентификатор пользователя или группы пользователей и др.), которая отражает желаемое качество обслуживания — в частности, «справедливость». В предложенной в данной главе модели подобные параметры не учитываются, поэтому вместе с увеличением доли задач с очень малым временем ожидания увеличилась и доля задач с очень большим временем ожидания. В дальнейшем целесообразно модифицировать алгоритм так, чтобы он позволял оптимизировать вычислительный процесс не только по критерию улучшения среднего времени ожидания, но и по ряду других критериев, актуальных для многопользовательских вычислительных систем.

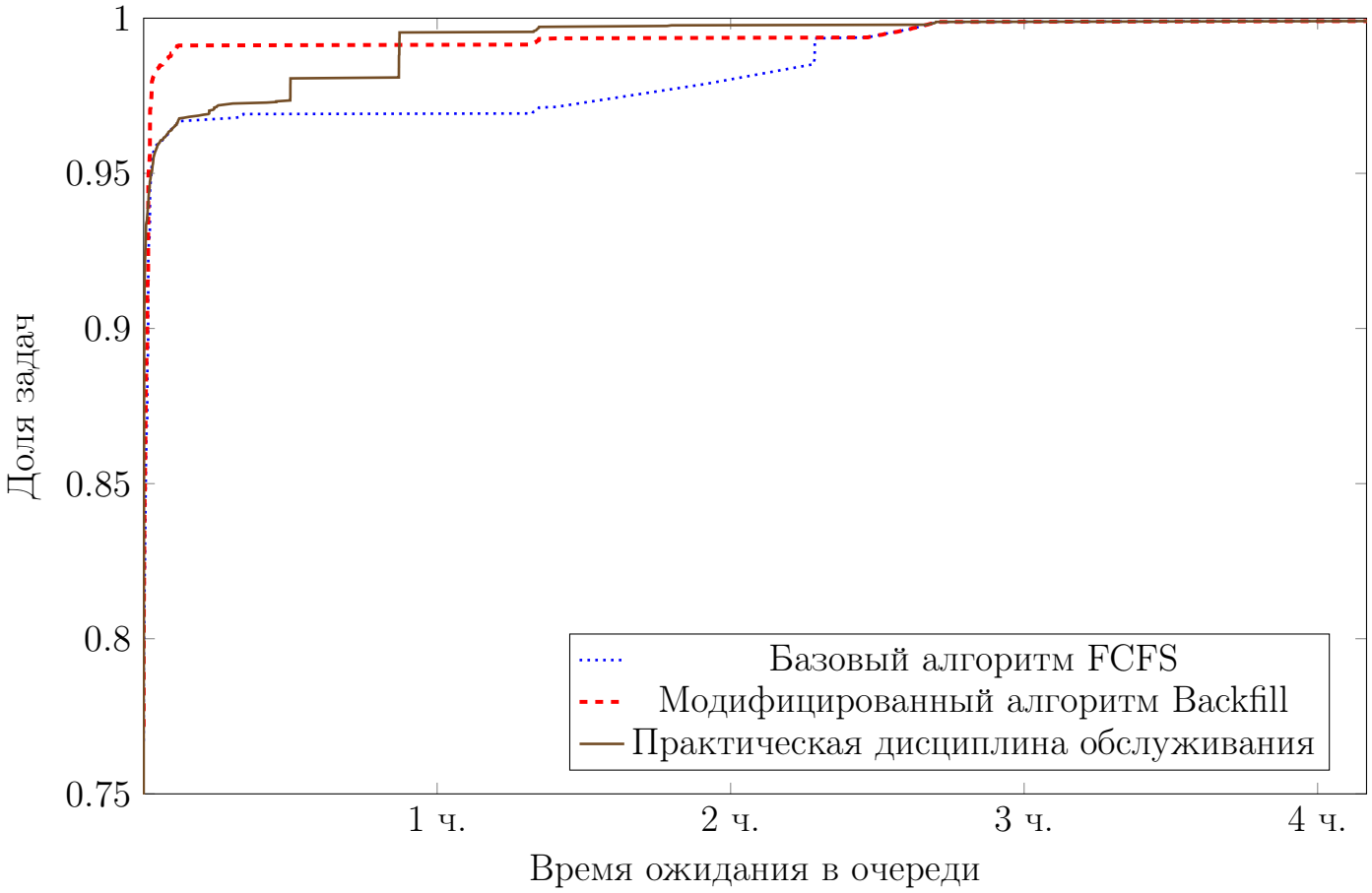

Рис. 2.8. Эмпирические функции распределения времени ожидания для различных алгоритмов планирования.

Глава 3

Теоретико-игровая модель управления заданиями в системе Desktop Grid

3.1. Виртуальный скрининг

Разработка новых лекарственных средств является трудоемкой и времяемкой задачей. От момента определения множества химических соединений, среди которых будет происходить поиск, до поступления готового лекарственного средства в продажу проходит в среднем от 7 до 15 лет. Цель процесса разработки лекарства [101] заключается в нахождении или синтезе низкомолекулярного (то есть обладающего относительно малой молекулярной массой) химического соединения, называемого также *лигандом*, которое способно взаимодействовать с белковой макромолекулой-*мишенью*, особым образом влияя на активность последней и тем самым изменяя ход течения заболевания. На различных этапах процесса из множества потенциальных лекарств выделяются лишь соединения, обладающие желаемыми свойствами. При этом исключаются вещества, не удовлетворяющие тому или иному критерию отбора — например, проявившие токсичное воздействие на организм мыши в серии экспериментов и т. д. Все этапы процесса разработки лекарства успешно проходят лишь доли процента исходного множества лигандов.

Начальный этап процесса, определение множества лигандов-кандидатов, которые затем предстоит тестировать в лабораторных условиях, сам по себе является весьма ресурсоемким. Выделяют различные методы поиска кандидатов в библиотеках химических соединений, которые содержат, как правило, миллионы записей. Так, при наличии информации о структуре лиганда с желаемыми биохимическими свойствами, возможен поиск сходных с ним по структуре молекул для последующего теоретического и практического анализа. Если струк-

тура и функции молекулы-мишени хорошо изучены, зачастую целесообразен прямой перебор всей библиотеки лигандов или ее части.

Метод автоматизированного перебора химических соединений в лабораторных условиях называется высокопроизводительным скринингом [102] и на протяжении нескольких десятков лет успешно применяется для разработки лекарств. Однако данный метод имеет ряд физических ограничений, которые влияют как на множество исходных данных для экспериментов (сокращая область применимости метода), так и на стоимость экспериментов, в ряде случаев делая их нецелесообразными.

С развитием методов высокоточного компьютерного моделирования процесса связывания лиганда с белком-мишенью, а также с разработкой хорошо приближающих реальность моделей макромолекул на основе результатов физических экспериментов, альтернативой высокопроизводительному скринингу стал его виртуальный аналог [101–103]. При этом могут использоваться как библиотеки молекул существующих химических веществ, так и модели молекул, не синтезированных ранее на практике, а также их фрагментов. Понятие виртуального скрининга включает в себя целый ряд вычислительных технологий, программных и аппаратных решений, как правило, требующих задействования высокопроизводительных компьютеров или систем распределенных вычислений и обработки данных. Далее в диссертационной работе под виртуальным скринингом будет подразумеваться один из этапов итерационного цикла разработки лекарства — процедура отбора моделей лигандов по результатам молекулярного докинга, то есть серии вычислительных экспериментов, моделирующих взаимодействие лиганда и мишени в трехмерном пространстве.

В процессе виртуального скрининга для каждого предсказанного взаимного расположения лиганда и мишени, полученного в результате молекулярного докинга, вычисляется значение специальной оценочной функции, характеризующее силу их связывания. В идеальном случае для этого были бы исследованы все доступные степени свободы многоатомной системы и найден глобальный

минимум потенциальной энергии взаимодействия между лигандом и мишенью. Однако в условиях большой поверхности поиска и множества локальных минимумов, полный перебор за приемлемое время невозможен. Поэтому в программном обеспечении для докинга реализованы методы поиска приближенного решения, в частности, генетический алгоритм, а также методы имитационного моделирования.

Вычисленные значения оценок силы связывания используются для выбора наиболее согласующегося с эмпирическими данными расположения молекул, ранжирования лигандов и сокращения пространства поиска. Результатом виртуального скрининга является множество лигандов с высокой предсказанной силой связывания. Мощность этого множества существенно меньше размера исходной библиотеки лигандов и позволяет синтезировать и проверить результаты в лабораторных условиях.

В качестве входных данных для проведения виртуального скрининга используются подготовленные к докингу библиотеки моделей лигандов и белков. Активное развитие подобных библиотек с открытым доступом послужило важным шагом к созданию ряда научно-исследовательских проектов по поиску лекарственных средств. Примеры крупных проектов добровольных вычислений, осуществляющих поиск лекарств, приведены в Таблице 3.1. Данные, приведенные в таблице, иллюстрируют тот факт, что высокий общественный интерес к подобным задачам позволяет задействовать для их решения значительные вычислительные мощности.

Однако для проведения виртуального скрининга при исследовании редких или локально распространенных заболеваний, а также на начальных этапах исследований использование систем добровольных вычислений затруднительно. В таких случаях целесообразнее задействовать вычислительные средства, которыми располагает научно-исследовательский коллектив, организовав систему Desktop Grid.

Таблица 3.1. Проекты добровольных вычислений, посвященные поиску лекарств.

Название проекта	**Исследуемые заболевания**	**Количество зарегистрированных вычислительных узлов**
Discovering Dengue Drugs — Together	Тропическая лихорадка, энцефалит Западного Нила, гепатит С, желтая лихорадка	Глобальная система World Community Grid, более 2,7 млн
Drug Search for Leishmaniasis	Лейшманиоз	
Influenza Antiviral Drug Search	Грипп	
Help Conquer Cancer	Рак	
Help Fight Childhood Cancer	Нейробластома	
Help Cure Muscular Dystrophy	Мышечная дистрофия	
FightAIDS@Home	ВИЧ	
Mapping Cancer Markers	Рак	
Say No to Schistosoma	Шистосомоз	
GO Fight Against Malaria	Малярия	
Docking@Home	ВИЧ, болезнь Паркинсона, артрит, рак	более 120 тысяч
FightMalaria@Home	Малярия	более 22 тысяч
Rosetta@Home	ВИЧ, малярия, болезнь Альцгеймера, рак, герпес, сибирская язва	более 1,2 миллиона
Folding@Home	Болезнь Альцгеймера, коровье бешенство, болезнь Паркинсона, диабет II типа, склероз, рак	более 170 тысяч

3.2. Постановка задачи

С октября 2013 г. по июль 2014 г. был реализован совместный научно-исследовательский проект между Институтом прикладных математических исследований КарНЦ РАН и Любекским Институтом экспериментальной дерматологии (до мая 2014 г. — Клиникой дерматологии, аллергологии и венерологии) при университете г. Любек, Германия. В ходе выполнения проекта была реализована вычислительная инфраструктура типа Desktop Grid с применением компьютеров, имеющихся в распоряжении сотрудников институтов, и проведен виртуальный скрининг лигандов для одного из белков. Молекулярный докинг проводился с применением открытого программного обеспечения AutoDock Vina [104].

На Рис. 3.1 представлены усредненные (в процентах от общей суммы) характеристики вычислительных узлов, вошедших в состав Desktop Grid на время выполнения проекта.

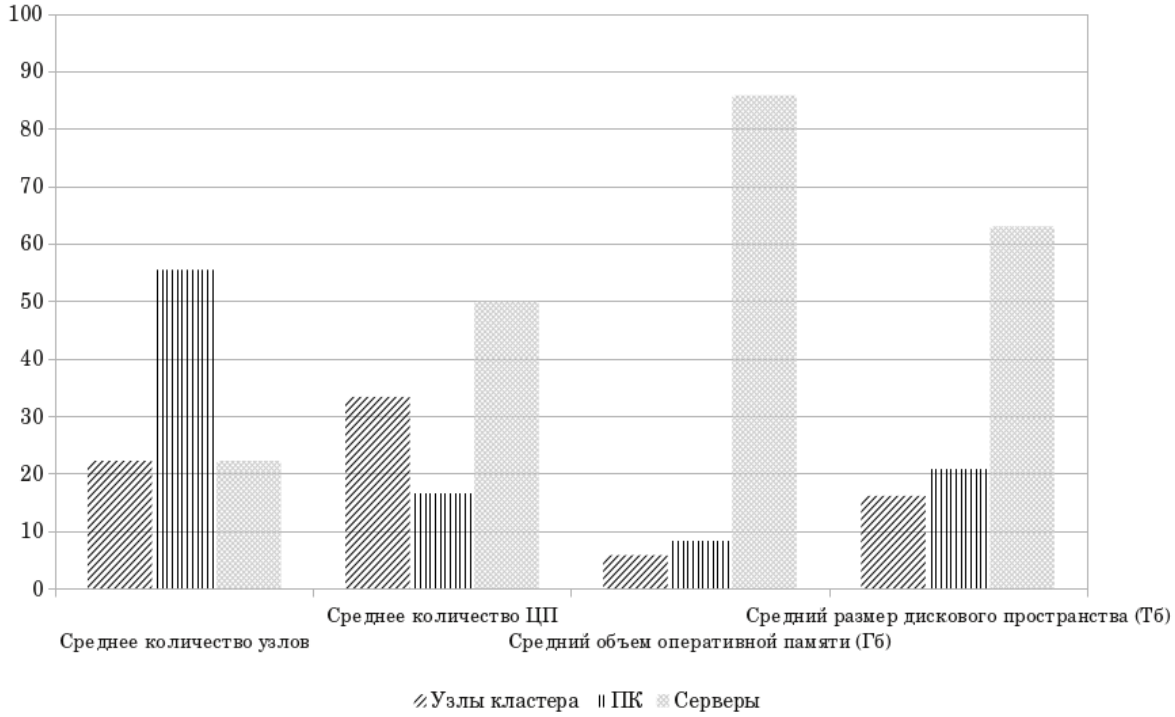

Рис. 3.1. Средние характеристики узлов Desktop Grid.

В качестве промежуточного программного обеспечения была использована платформа BOINC (Berkeley Open Infrastructure for Desktop Computing) [105].

Модель управления заданиями в BOINC является централизованной, типа «клиент-сервер». Инициатором любого взаимодействия с сервером является клиент. В состав серверного компонента входит ряд независимых подсистем, отвечающих за управление базой данных, создание экземпляров рабочих заданий, пересылку ассоциированных с ними файлов по сети, получение и обработку результатов и т.д. Клиентская часть выполняет функции, которые можно разделить на два класса:

1. Коммуникация с сервером или группой независимых серверов BOINC (уведомление о готовности принимать задания; уведомление о наличии результатов выполнения заданий, готовых к отправлению; получение по сети исполняемых файлов, сценариев командной оболочки, программных библиотек и других файлов, необходимых для выполнения вычислений и др.).

2. Управление вычислительным процессом на клиенте (графический интерфейс пользователя; запуск, временная и полная остановка выполнения задания; отслеживание процессорного времени и прочих ресурсов, занимаемых заданием; управление специальной файловой подсистемой и др.).

Адаптация научного приложения для выполнения на клиентской части может производиться как внесением изменений в его исходный код, так и при помощи вспомогательных приложений-«оберток». Являясь программным продуктом с открытым исходным кодом (распространяемым по лицензии Lesser General Public License), BOINC допускает модификацию компонентов как серверной, так и клиентской части в целях оптимизации работы конкретного вычислительного проекта на основе данной платформы.

В ходе выполнения проекта основными причинами ошибок при выполнении вычислительных заданий являлись нехватка оперативной памяти и отказы в доступе к жесткому диску в моменты интенсивного использования компьютеров их хозяевами, а также ошибками в реализации клиентской части программ-

ного обеспечения (в частности, отсутствием на узлах динамически подключаемых библиотек нужной версии).

В процессе виртуального скрининга на Desktop Grid задания, выполняющие моделирование связывания отдельных лигандов с белком, выполняются на клиентах за относительно малое время — от нескольких секунд до нескольких минут. Быстро выполняя задания, отсылая их результаты серверу и запрашивая новые задания, клиенты генерируют интенсивный поток запросов, что накладывает высокие требования на программно-аппаратные характеристики сервера и сети передачи данных. При достаточно большом количестве клиентов такой процесс может иметь эффект DDoS-атаки на сервер.

Для снижения нагрузки на сервер в диссертационной работе предлагается группировать вычислительные задания в «пакеты». Узлы Desktop Grid являются ненадежными, и если хотя бы одно задание из пакета завершилась с ошибкой, то клиенту придется повторно запрашивать пакет у сервера и запускать его на выполнение. Метод группировки заданий для повышения эффективности работы Desktop Grid рассмотрен, например, в [63], где исследуется возможность снижения общего времени выполнения множества заданий при наличии репликации. В работе [106] для снижения нагрузки на сервер предлагается объединять территориально близкие узлы Desktop Grid в группы и выделять в каждой группе узел, исполняющий функции промежуточной подсистемы распределения заданий.

В данной главе приводится математическая модель Desktop Grid в форме двухуровневой иерархической игры, общая постановка которой приведена, например, в [27]. Выбор стратегий игроков заключается в определении оптимального размера пакета и количества назначаемых заданий, и в равновесной ситуации достигается баланс между объемом вычислений на клиентах и нагрузкой на сервер.

3.3. Математическая модель

Запишем задачу в форме двухуровневой иерархической игры $m+1$ игроков — одного сервера и m клиентов Desktop Grid.

Сервер M_0 имеет N вычислительных заданий и может задействовать для их выполнения множество клиентов $M_1, \ldots, M_m$. Сервер выбирает способ распределения заданий между клиентами $u = (N_1, \ldots, N_m)$, где N_i — количество заданий, назначенное клиенту M_i. Множество возможных способов распределения заданий u составляет множество стратегий сервера,

$$U = \{u = (u_1, \ldots, u_m) : u_i \in \mathbb{Z},\ 0 \leq u_i \leq N,\ \sum_{i=1}^{m} u_i = N\} \tag{3.1}$$

Зная назначенное сервером N_i, клиент M_i выбирает размер пакета n_i, минимизирующий его расходы. Множество возможных размеров пакета n_i для данного N_i составляет множество стратегий i-го клиента,

$$V_i(N_i) = \left\{ v_i \in \mathbb{Z} : \begin{cases} 1 \leq v_i \leq N_i & \text{при } N_i \geq 1, \\ v_i = 0 & \text{при } N_i = 0 \end{cases} \right\} \tag{3.2}$$

Таким образом, в сумме клиент должен выполнить $\lceil \frac{N_i}{n_i} \rceil$ пакетов заданий. Функция выигрыша для клиента M_i выражает его расходы на вычисления:

$$C_i(N_i, n_i) = -K(n_i) \left\lceil \frac{N_i}{n_i} \right\rceil, \tag{3.3}$$

где $K(n_i)$ — стоимость расчета пакета из n_i заданий.

В данной модели мы полагаем, что выигрыш клиента не зависит от стратегий, выбранных другими клиентами, что согласуется с общими принципами функционирования Desktop Grid. Однако в дальнейшем модель может быть расширена для соответствия ситуациям, когда влиянием игроков второго уровня друг на друга не следует пренебрегать. Например, клиенты могут разделять

между собой единый канал связи с сервером, передавать друг другу на выполнение завершившиеся с ошибкой пакеты, соревноваться за количество полезных вычислений при наличии репликации и др. Для систем распределенных вычислений, в которых не выполняются принятые для рассматриваемой модели Desktop Grid предположения (в частности, полная информированность сервера о выборе клиента), для определения оптимальных стратегий потребуется определить другие функции выигрыша и методы согласования действий сервера и клиентов.

Оптимальный размер пакета n_i^* минимизирует расходы клиента, то есть максимизирует его выигрыш:

$$C_i^* = C_i(N_i, n_i^*) = \max_{n_i \in V_i(N_i)} C_i(N_i, n_i),\ 1 \le i \le m$$

Зная размеры пакетов, которые выберут клиенты для заданного количества заданий, сервер стремится минимизировать свои расходы. Функция выигрыша для сервера выражает его расходы на подготовку пакетов, ожидание и обработку результатов:

$$C_0 = C_0(N_1, \ldots, N_m, n_1, \ldots, n_m)$$

Оптимальное распределение заданий $u^* = (N_1^*, \ldots, N_m^*)$ минимизирует расходы сервера, то есть максимизирует его выигрыш:

$$C_0^* = C_0(N_1^*, \ldots, N_m^*, n_1^*, \ldots, n_m^*) = \max_{u \in U} C_0(N_1, \ldots, N_m, n_1^*, \ldots, n_m^*)$$

3.3.1. Функция выигрыша клиента

Если сервер назначил клиенту N заданий, общая стоимость расчетов для клиента составит $C(n) = K(n) \left\lceil \frac{N}{n} \right\rceil, n \ge 1$. Расходы клиента $K(n)$ на расчет одного пакета размером n заданий можно выразить следующим образом:

$$K(n) = G_1 S + G_2 Z \tag{3.4}$$

Здесь первое слагаемое $G_1 S$ — плата за коммуникацию с сервером; G_1 — стоимость одного обращения к серверу, S — математическое ожидание количества обращений к серверу для расчета одного пакета. Второе слагаемое $G_2 Z$ — плата непосредственно за расчеты (например, за электричество); G_2 — стоимость расчета одного задания, $Z = n \sum_{i=0}^{\infty} (1-\bar{p})^i = \frac{n}{\bar{p}}$ — математическое ожидание числа посчитанных заданий с учетом перезапусков в результате ошибок. Величина $\bar{p}$ — вероятность успешного выполнения пакета размером n заданий за одну попытку, $\bar{p} = p^n$, где p — вероятность успешного выполнения одного задания за одну попытку.

Каждый расчет пакета заданий требует одного запроса к серверу и одного отчета с результатом на сервер; если возникла ошибка, то вместо результата посылается отчет об ошибке, и требуется пересчет всего пакета, то есть новый запрос к серверу. Расходы на коммуникацию узла с сервером определяются стоимостью выполнения операции сервером:

- $A_{\text{req}}(n) \geq 0$ — стоимость запроса пакета заданий,
- $a_{\text{req}} = A_{\text{req}}(1)$ — стоимость запроса одного задания,
- $A_{\text{rep}}(n) \geq 0$ — стоимость отправки серверу результата пакета заданий,
- $a_{\text{rep}} = A_{\text{rep}}(1)$ — стоимость отправки серверу результата одного задания,
- $A_{\text{err}} = a_{\text{err}} \geq 0$ — стоимость отправки серверу уведомления об ошибке.

Таким образом, математическое ожидание количества обращений к серверу для расчета одного пакета принимает вид (3.5), а функция выигрыша клиента M_i — вид (3.6).

$$S(n,p) = \sum_{i=0}^{\infty} (1-\bar{p})^i \bar{p} \left(i(A_{\text{req}}(n) + A_{\text{err}}(n)) + A_{\text{req}}(n) + A_{\text{rep}}(n)\right) = \\ = \frac{A_{\text{req}}(n) + A_{\text{err}}(n)}{\bar{p}} + A_{\text{rep}}(n) - A_{\text{err}}(n) \quad (3.5)$$

$$C_i(n_i) = -\left\lceil \frac{N_i}{n_i} \right\rceil \left(G_1 \left(\frac{A_{\text{req}}(n_i) + A_{\text{err}}(n_i)}{p_i^n} + A_{\text{rep}}(n_i) - A_{\text{err}}(n_i) \right) + G_2 \frac{n_i}{p_i^n} \right) \quad (3.6)$$

3.3.2. Функция выигрыша сервера

Расходы сервера на подготовку отдельного пакета и обработку его результата не зависят от общего количества заданий, поэтому функцию выигрыша можно записать в виде суммы двух слагаемых:

$$C_0(N_1, \ldots, N_m, n_1, \ldots, n_m) = -T(N_1, \ldots, N_m, n_1, \ldots, n_m) - \\ - F(N_1, \ldots, N_m, n_1, \ldots, n_m) = -\sum_{i=1}^{m} \left\lceil \frac{N_i}{n_i} \right\rceil (B(n_i) + S(n_i, p_i)) - \\ - F(N_1, \ldots, N_m, n_1, \ldots, n_m), \quad (3.7)$$

где $T(N_1, \ldots, N_m, n_1, \ldots, n_m)$ — функция стоимости обработки сервером всех заданий, $B(n_i)$ — стоимость создания одного пакета из n_i заданий и обработки результата его выполнения, $S(n_i, p_i)$ — математическое ожидание количества обращений к серверу для расчета пакета (3.5), $F(N_1, \ldots, N_m, n_1, \ldots, n_m)$ — функция штрафа за ожидание результатов.

Таким образом, имеется игра в нормальной форме:

$$\Gamma = \langle \{M_0, M_1, \ldots, M_m\}, \{U, V_1, \ldots, V_m\}, \{C_0, C_1, \ldots, C_m\} \rangle \quad (3.8)$$

Множество стратегий игрока M_0 задано выражением (3.1), множества стратегий игроков $M_1, \ldots, M_m$ — выражением (3.2). Функции выигрыша игроков имеют вид (3.6), (3.7).

3.4. Аналитическое выражение функций выигрыша

Предположим, что стоимость запроса пакета заданий имеет линейную зависимость от размера пакета ($A_{\text{req}}(n) = kn + b$, $b \neq 0$, $k > 0$), стоимость ответа

прямо пропорциональна размеру пакета ($A_{\text{rep}}(n) = \alpha n$, $\alpha > 0$), а стоимостью отправки серверу сообщения об ошибке можно пренебречь ($A_{\text{err}}(n) \equiv 0$). Пусть стоимость создания и обработки пакета также линейна ($B(n) = xn + y$, $x > 0, y \neq 0$), а функция штрафа за ожидание имеет вид

$$F(N_1, \ldots, N_m) = \frac{1}{m} \sum_{i=1}^{m} \left(\frac{N_i}{p_i^{n_i^*}} \right)^2 .$$

Такие предположения о виде функций согласуются с практикой и частично упрощены для последующих аналитических выкладок. Будем также считать, что количество заданий N достаточно велико, чтобы округление до ближайшего целого в (3.3) можно было опустить.

Найдем оптимальный размер пакета n^* для клиента при $n \geq 1$, полагая, что клиент получил $N > 0$ заданий.

$$C(n) = -\frac{N}{n} \left(G_1 \left(\frac{kn + b}{p^n} + \alpha n \right) + G_2 \frac{n}{p^n} \right) = -N \left(\frac{G_1 k + G_2}{p^n} + \frac{G_1 b}{n p^n} + G_1 \alpha \right)$$

$$C'(n) = -N \left(\frac{-\ln \mathrm{p}(G_1 k + G_2)}{p^n} - \frac{G_1 b (p^n + n p^n \ln \mathrm{p})}{n^2 p^{2n}} \right) = \\ = N \frac{G_1 b + n G_1 b \ln \mathrm{p} + n^2 \ln \mathrm{p}(G_1 k + G_2)}{n^2 p^n}$$

Знаменатель дроби в правой части уравнения положителен при любых значениях n. В числителе — квадратичная функция с отрицательным коэффициентом при n^2 ($\ln \mathrm{p}(G_1 k + G_2) < 0$, т.к. $0 < p < 1$). Найдем нули данной функции.

$$C'(n) = 0 \iff n^2 \ln \mathrm{p}(G_1 k + G_2) + n G_1 b \ln \mathrm{p} + G_1 b = 0$$

$$D = G_1^2 b^2 \ln \mathrm{p}^2 - 4 \ln \mathrm{p}(G_1 k + G_2) G_1 b$$

а) $D \leq 0 \Rightarrow C'(n) \leq 0 \ \ \forall n \ \Rightarrow C(n)$ монотонно убывает, и оптимальный

размер пакета $n^* = 1$. Это соотношение верно при выполнении условий $b < 0$ и $\ln p \geq 4(G_1 k + G_2)/G_1 b$.

b) $D > 0 \Rightarrow C'(n) = 0$ при

$$n_{1,2} = \frac{-G_1 b \ln \mathrm{p} \pm \sqrt{G_1^2 b^2 \ln \mathrm{p}^2 - 4 \ln \mathrm{p}(G_1 k + G_2) G_1 b}}{2 \ln \mathrm{p}(G_1 k + G_2)} \tag{3.9}$$

$2 \ln \mathrm{p}(G_1 k + G_2) < 0 \Rightarrow n_1 < n_2$, поэтому $C(n)$ имеет минимум в т. n_1 и максимум в т. n_2. Клиент выбирает оптимальный размер пакета n^* в зависимости от значения n_2, определенного (3.9):

$$n^* = \begin{cases} 1, & \text{если } n_2 \leq 1, \\ \lfloor n_2 \rfloor, & \text{если } 1 < n_2 < N \text{ и } C(\lfloor n_2 \rfloor) \geq C(\lceil n_2 \rceil), \\ \lceil n_2 \rceil, & \text{если } 1 < n_2 < N \text{ и } C(\lfloor n_2 \rfloor) < C(\lceil n_2 \rceil), \\ N, & \text{иначе.} \end{cases} \tag{3.10}$$

Пусть в состав Desktop Grid входят два клиента I и II с надежностями $0 < p < 1$ и $0 < q < 1$. Сервер выбирает способ разделения N заданий на две части: N_1 заданий будет отправлено первому клиенту, $N - N_1$ — второму. Найдем точку максимума N_1 функции выигрыша сервера при фиксированных

размерах пакетов n_1 и n_2, выбранных клиентами I и II соответственно.

$$C_0(N_1) = -\frac{N_1}{n_1}(B(n_1)+S(n_1,p)) - \frac{N-N_1}{n_2}(B(n_2)+S(n_2,q)) - \\ - \frac{\left(\frac{N_1}{p^{n_1}}\right)^2 + \left(\frac{N-N_1}{q^{n_2}}\right)^2}{2} = -\frac{N_1}{n_1}\left(xn_1 + y + \frac{kn_1+b}{p^{n_1}} + \alpha n_1\right) - \\ - \frac{N-N_1}{n_2}\left(xn_2 + y + \frac{kn_2+b}{q^{n_2}} + \alpha n_2\right) - \\ - \frac{N_1^2}{2p^{2n_1}} - \frac{(N-N_1)^2}{2q^{2n_2}} = \\ = -N_1\left(\left(\frac{kn_1+b+yp^{n_1}}{n_1p^{n_1}} - \frac{kn_2+b+yq^{n_2}}{n_2q^{n_2}}\right) - \frac{N}{q^{2n_2}}\right) - \\ - N_1^2\left(\frac{1}{2p^{2n_1}} + \frac{1}{2q^{2n_2}}\right) - \frac{N^2}{2q^{2n_2}} - N\left(x+\alpha+\frac{kn_2+b+yq^{n_2}}{n_2q^{n_2}}\right) \longrightarrow \max$$

$C_0(N_1)$ — квадратичная функция с отрицательным коэффициентом при N_1^2, следовательно, она имеет единственную точку максимума на числовой прямой.

$$C_0'(N_1) = -N_1\left(\frac{1}{p^{2n_1}} + \frac{1}{q^{2n_2}}\right) - \left(\frac{kn_1+b+yp^{n_1}}{n_1p^{n_1}} - \frac{kn_2+b+yq^{n_2}}{n_2q^{n_2}} - \frac{N}{q^{2n_2}}\right)$$

$$C_0'(N_1^0) = 0 \iff N_1^0 = \left(\frac{kn_1+b+yp^{n_1}}{n_1p^{n_1}} - \frac{kn_2+b+yq^{n_2}}{n_2q^{n_2}} - \frac{N}{q^{2n_2}}\right) \Big/ \left(-\frac{p^{2n_1}+q^{2n_2}}{p^{2n_1}q^{2n_2}}\right)$$

$$N_1^0 = \frac{p^{2n_1}q^{2n_2}}{p^{2n_1}+q^{2n_2}}\left(k\left(\frac{1}{q^{n_2}} - \frac{1}{p^{n_1}}\right) + b\left(\frac{1}{n_2q^{n_2}} - \frac{1}{n_1p^{n_1}}\right) + \right. \\ \left. + y\left(\frac{1}{n_2} - \frac{1}{n_1}\right) + \frac{N}{q^{2n_2}}\right) \quad (3.11)$$

Сервер выбирает оптимальное количество заданий N_1^* в зависимости от значе-

ния N_1^0, определенного (3.11):

$$N_1^* = \begin{cases} 0, & \text{если } N_1^0 \le 0, \\ [N_1^0], & \text{если } 0 < N_1^0 < N, \\ N, & \text{иначе.} \end{cases} \tag{3.12}$$

Предположим теперь, что в состав Desktop Grid входит один клиент типа I и $m > 0$ клиентов типа II. Поскольку параметры m клиентов второго типа идентичны, в оптимальном для сервера распределении клиент типа I получит N_1^*, а каждый из клиентов типа II — по $\frac{N-N_1^*}{m}$ заданий. Выражение для оптимального количества заданий N_1^* остается прежним (3.12) при N_1^0, заданном выражением (3.11′).

$$N_1^0 = \frac{m(m+1)p^{2n_1}q^{2n_2}}{2(p^{2n_1}+mq^{2n_2})}\left(k\left(\frac{1}{q^{n_2}}-\frac{1}{p^{n_1}}\right)+b\left(\frac{1}{n_2q^{n_2}}-\frac{1}{n_1p^{n_1}}\right)+\right.$$
$$\left.+\,y\left(\frac{1}{n_2}-\frac{1}{n_1}\right)+\frac{2N}{m(m+1)q^{2n_2}}\right) \tag{3.11′}$$

Пример 1. Пусть в состав Desktop Grid входят два клиента с надежностями $p = 0.999$ и $q = 0.97$; $k = 0.0025$, $b = 0.003$, $\alpha = 0.005$, $x = 0.025$, $y = 0.5$, $G_1 = 1$, $G_2 = 0.01$ (расчет задания на клиенте стоит в 100 раз дешевле, чем обращение к серверу), $N = 1\,000\,000$. Тогда функция выигрыша клиента I, $C_I(n)$, достигает максимума при

$$n_1 = \frac{0.003\sqrt{|\ln 0.999|} - \sqrt{0.003(4\cdot 0.0025 + 0.04 - 0.003\cdot\ln 0.999)}}{-2\sqrt{|\ln 0.999|}(0.0025+0.01)} \approx 15.4$$

и, поскольку $C_I(15) \approx -N_1 \times 0.017892 > C_I(16) \approx -N_1 \times 0.0178922$, оптимальный размер пакета для клиента I равен $n_1^* = 15$. Аналогично,

$$n_2 = \frac{0.003\sqrt{|\ln 0.97|} - \sqrt{0.003(4\cdot 0.0025 + 0.04 - 0.003\cdot\ln 0.97)}}{-2\sqrt{|\ln 0.97|}(0.0025+0.01)} \approx 2.7$$

и, поскольку $C_{II}(2) \approx -N \times 0.0198794 < C_{II}(3) \approx -N \times 0.0197917$, оптимальный размер пакета для клиента II равен $n_2^* = 3$.

Подставляя значения n_1^* и n_2^* в (3.11) и используя условие (3.12), получаем оптимальное значение N_1^*:

$$N_1^* = [N_1^0] = [N \times 0.538111 + 0.0602546] = 538\,111.$$

Выигрыш клиента I составит $C_I(n_1^*) = -9\,627.9$, выигрыш клиента II $C_{II}(n_2^*) = -9\,013.7$. Выигрыш сервера составит $C_0(N_1^*) = -2.77254 \cdot 10^{11}$, а стоимость обработки сервером всех заданий $T(N_1^*, n_1^*) = -128\,165$.

Рассмотрим случай, когда группировка заданий в пакеты отсутствует, то есть $n_1^* = n_2^* = 1$. Тогда оптимальное количество заданий для первого клиента $N_1^* = 514\,725$. Выигрыш каждого из клиентов меньше, чем в случае группировки в пакеты: $C_I(1) = -10\,559.8$, $C_{II}(1) = -10\,180.8$. Выигрыш сервера в отсутствие затрат на обработку пакетов становится на 7.5% больше: $C_0 = -2.57879 \cdot 10^{11}$. Однако и стоимость обработки всех заданий становится в 4.18 раза больше: $T = -535\,585$. Таким образом, группировка заданий в пакеты, снизив общий выигрыш сервера, позволила тем не менее увеличить выигрыши клиентов и в 4.18 раза снизить общую нагрузку на сервер.

Пример 2. Предположим, что программная часть клиента I установлена на сервере, то есть можно считать, что затраты на коммуникацию по сети у него отсутствуют: $G_1^I = G_2^I = 0.01$. Для клиента II, как и в Примере 1, $G_1^{II} = 1$, $G_2^{II} = 0.01$. Значения остальных параметров будем считать прежними.

Тогда оптимальные размеры пакетов $n_1^* = 2$, $n_2^* = 3$. Подставляя данные значения в (3.11) и используя условие (3.12), получаем оптимальное значение $N_1^* = 544\,570$. Выигрыш клиента I составит $C_I(n_1^*) = -5\,505.7$, выигрыш клиента II $C_{II}(n_2^*) = -9\,013.7$, выигрыш сервера $C_0(N_1^*) = -2.73377 \cdot 10^{11}$; стоимость обработки сервером всех заданий $T(N_1^*, n_1^*) = -245\,977$.

Группировка заданий в пакеты снизила выигрыш сервера на 6%, но позво-

лила снизить стоимость обработки всех заданий в 2.2 раза.

Пример 3. Предположим, что программная часть клиента I установлена на сервере, и имеется 16 клиентов второго типа. Значения параметров будем считать теми же, что в Примере 2. Тогда оптимальным для сервера будет назначить $N_1^* = 69\,536$ заданий клиенту I и по 58 154 задания каждому из клиентов типа II. Выигрыш сервера при этом составит $C_0 = -4.10697 \cdot 10^9$.

Группировка заданий в пакеты снизила выигрыш сервера на 12%, но позволила снизить стоимость обработки всех заданий в 2.5 раза.

3.5. Вычислительные эксперименты

3.5.1. Модель Desktop Grid с клиентами различных типов

Вычислительные узлы, вошедшие в состав Desktop Grid в ходе выполнения проекта, можно условно разделить на три типа: настольные компьютеры, узлы вычислительного кластера и веб-серверы. Узлы одного и того же типа обладают схожим временем доступности ресурсов (например, настольные компьютеры приостанавливают вычисления для Desktop Grid в рабочие часы, а узлы кластера — на протяжении нескольких суток в месяц, когда на кластере запущена параллельная программа), а также схожими программно-аппаратными характеристиками, а следовательно, и надежностью.

В ходе выполнения проекта была оценена надежность узлов различного типа. При выполнении 660 107 экземпляров заданий на настольных компьютерах 13 273 из них завершились ошибкой клиента, отсюда можем оценить $p \approx 1 - 13\,273/660\,107 = 0.98$. Аналогично, для узлов кластера $p \approx 1 - 22\,322/815\,502 = 0.973$, для веб-серверов $p \approx 1 - 2\,854/391\,672 = 0.993$.

Относительно низкую надежность узлов второго типа можно объяснить следующим образом. В Desktop Grid клиент получает задания «порциями» в расчете приблизительно на сутки непрерывной работы, чтобы снизить время простоя в случае длительного отсутствия сетевой связи. Если клиент времен-

но недоступен или возникают ошибки, интервал времени до отправления новой «порции» автоматически увеличивается. Количество заданий зависит от вычислительной мощности клиента, а следовательно, оно достаточно велико для узлов вычислительного кластера. В течение выполнения проекта на Desktop Grid администратор кластера несколько раз обновлял программное обеспечение на узлах, в результате чего все полученные на тот момент задания завершались ошибками, связанными с отсутствием необходимых библиотек. На настольных компьютерах, благодаря небольшим размерам «порций», ошибок подобного рода возникало меньше.

По статистике обработки запросов к серверу было получено приближение функции $A_{\text{req}}(n)$ при помощи полиномиальной регрессии, $A_{\text{rep}}(n)$, $A_{\text{err}}(n)$ и $B(n)$ — при помощи линейной регрессии:

$$A_{\text{req}}(n) = \begin{cases} -1.702 \cdot 10^{-7} n^3 + 1.54019 \cdot 10^{-5} n^2 - 7.43959 \cdot 10^{-5} n + 0.000170244, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad 1 \le n < 60, \\ 0.01176692\, n - 0.49019617,\ n \ge 60. \end{cases}$$

$$A_{\text{rep}}(n) = 0.00482994\, n + 0.001869563$$

$$A_{\text{err}}(n) \equiv A_{\text{err}}(1) = a_{\text{err}} = 0.006374616$$

$$B(0) = 0,\ B(n) = 0.000147776\, n + 0.59580745,\ n > 0$$

Пусть функция штрафа за ожидание F выражает длительность расчетов и имеет вид

$$F(N_1, \ldots, N_m, n_1, \ldots, n_m) = \sum_{i=1}^{m} S(n_i, p_i) \times \frac{N_i}{n_i} + \max_{1 \le i \le m} \left(t_e \times \frac{N_i}{p_i^{n_i}} \right),$$

где $t_e = 63.54$ секунды — среднее время выполнения задания на клиенте, первое слагаемое выражает общую длительность обработки запросов клиентов к серверу, второе — длительность расчетов на клиентах.

Запишем игровую модель системы Desktop Grid с сервером M_0 и тремя клиентами различных типов, используя полученные численные характеристики. Клиент M_1 (настольный компьютер) имеет надежность $p_1 = 0.98$, клиент M_2 (узел кластера) — $p_2 = 0.973$, клиент M_3 (веб-сервер) — $p_3 = 0.993$. Таким образом, имеется игра

$$\Gamma = \langle \{M_0, M_1, M_2, M_3\}, \{U, V_1, V_2, V_3\}, \{C_0, C_1, C_2, C_3\} \rangle \tag{3.13}$$

Построим ситуацию равновесия по Нэшу в данной игре.

Пусть $N = 10\,000$. Зная N_i, клиент M_i $(i = 1, 2, 3)$ выбирает оптимальный для себя размер пакета n_i^*: $\max\limits_{n \in V_i(N_i)} C_i(N_i, n) = C_i(N_i, n_i^*)$.

Функция выигрыша для клиента M_i имеет вид

$$C_i(N_i, n) = -\frac{N_i}{n}\left(G_1\left(\frac{A_{\text{req}}(n) + a_{\text{err}}}{p_i^n} + A_{\text{rep}}(n) - a_{\text{err}}\right) + G_2\frac{n}{p_i^n}\right),\ n \geq 1,$$

$$C_i(N_i, n) \equiv 0,\ n = 0. \tag{3.14}$$

Положим $G_2 = 0.01$, $G_1 = 1$. Тогда оптимальные размеры пакетов для M_1 и M_2 (Рис. 3.2):

$$n_1^* = \begin{cases} N_1, & 0 \leq N_1 \leq 3 \\ 3, & 4 \leq N_1 \leq N \end{cases} \qquad n_2^* = \begin{cases} N_2, & 0 \leq N_2 \leq 3 \\ 3, & 4 \leq N_2 \leq N \end{cases} \tag{3.15}$$

Оптимальный размер пакета для M_3 (Рис. 3.2):

$$n_3^* = \begin{cases} N_3, & 0 \leq N_3 \leq 5 \\ 5, & 6 \leq N_3 \leq N \end{cases} \tag{3.16}$$

Выигрыш сервера M_0 выражает его расходы в виде общей длительности выполнения проекта в секундах. Сервер стремится минимизировать расходы, т. е. максимизировать выигрыш (3.17), выбором распределения заданий $u = (N_1, N_2, N_3)$

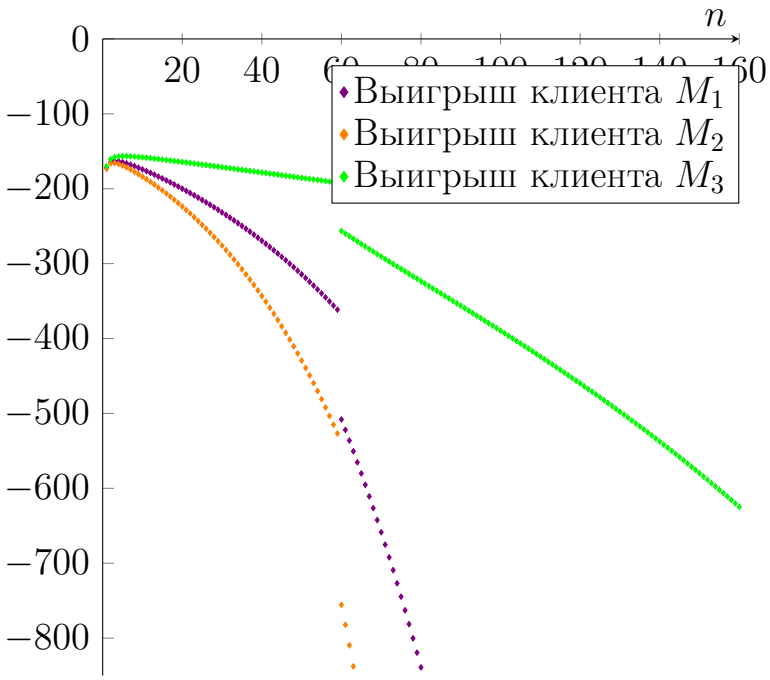

Рис. 3.2. Выигрыши клиентов в зависимости от размеров пакетов ($N_{1,2,3} = 10\,000$).

при ограничениях на переменные (3.18).

$$C_0(N_1, N_2, N_3, n_1^*, n_2^*, n_3^*) = -\sum_{i=1}^{3} \frac{N_i}{n_i^*} \times B(n_i^*) - \sum_{i=1}^{3} \frac{N_i}{n_i^*} \times S(n_i^*, p_i) - \\ - \max_{1 \le i \le 3} \left(t_e \times \frac{N_i}{p_i^{n_i^*}} \right) \longrightarrow \max_{(N_1, N_2, N_3)} \quad (3.17)$$

$$0 \le N_i \le N,\ i = 1, 2, 3;\ N_1 + N_2 + N_3 = N;\ N_i \in \mathbb{Z},\ i = 1, 2, 3 \quad (3.18)$$

Таким образом, равновесием по Нэшу в игре (3.13) является точка (N_1^*, N_2^*, N_3^*, n_1^*, n_2^*, n_3^*) — решение задачи целочисленного нелинейного программирования (3.15)–(3.18). Для решения задачи была реализована программа на языке C. При $N = 10\,000$ $N_1^* = 3\,328$, $N_2^* = 3\,258$, $N_3^* = 3\,414$, $n_1^* = 3$, $n_2^* = 3$, $n_3^* = 5$. Выигрыш сервера составляет $C_0^* = -226\,486.66$.

Оценим общее время выполнения $N = 10\,000$ заданий без группировки в пакеты, когда сервер посылает клиентам новые задания по мере выполнения ими старых (подобная дисциплина распределения заданий реализована в Desktop Grid на базе программной платформы BOINC по умолчанию). В этом

случае ожидаемая длительность выполнения N_i заданий клиентом M_i с учетом коммуникаций с сервером и ошибок составит

$$T(N_i) = \frac{N_i \times t_e}{p_i} + N_i \times S(1, p_i) = N_i \left(\frac{t_e}{p_i} + \frac{A_{\text{req}}(1) + a_{\text{err}}}{p_i} + A_{\text{rep}}(1) - a_{\text{err}} \right),$$

$$T(N_1) = N_1 \cdot 64.8396\,\text{с},\ T(N_2) = N_2 \cdot 65.3061\,\text{с},\ T(N_3) = N_3 \cdot 63.9907\,\text{с}.$$

Клиент M_i получит N_i заданий, где $N_1 + N_2 + N_3 = N$ и $N_1 : N_2 : N_3 \approx \frac{1}{T(N_1)} : \frac{1}{T(N_2)} : \frac{1}{T(N_3)}$, то есть $N_1 \approx 3\,330$, $N_2 \approx 3\,308$, $N_3 \approx 3\,362$. Общее ожидаемое время выполнения составит

$$\sum_{i=1}^{3} N_i \times B(1) + \sum_{i=1}^{3} N_i \times S(1, p_i) + \max_{1 \le i \le 3} \{ t_e \times \frac{N_i}{p_i} \} = 222\,038.19\,\text{с},$$

что меньше, чем полученное в примере время $|C_0^*| = 226\,486.66$ с. Однако в примере за счет увеличения общего времени выполнения достигнуто снижение общей нагрузки на сервер.

Суммарная стоимость коммуникаций с сервером в примере составляет

$$S^* = \sum_{i=1}^{3} \frac{N_i}{n_i^*} \times B(n_i^*) + \sum_{i=1}^{3} \frac{N_i}{n_i^*} \times S(n_i^*, p_i) = 1771.47.$$

Суммарная стоимость коммуникаций с сервером при дисциплине распределения заданий по умолчанию составляет

$$S = \sum_{i=1}^{3} N_i \times B(1) + \sum_{i=1}^{3} N_i \times S(1, p_i) = 6028.85.$$

Таким образом, при увеличении общего времени выполнения на 2% нагрузка на сервер, благодаря выбору оптимальных размеров пакетов, снизилась в 3.4 раза.

3.5.2. Модель Desktop Grid с m клиентами

Пусть в состав Desktop Grid входит m_1 настольных компьютеров, m_2 узлов вычислительного кластера и m_3 веб-серверов, $m_1+m_2+m_3 = m$. Предположим, что каждый клиент i-го типа имеет Q_i центральных процессоров (ЦП) и может обрабатывать Q_i пакетов заданий одновременно, и что каждому клиенту i-го типа назначается одно и то же количество заданий K_i. Таким образом, имеется игра

$$\Gamma = \langle \{M_0, M_1, \ldots, M_m\}, \{U, V_1, \ldots, V_m\}, \{C_0, C_1, \ldots, C_m\} \rangle \tag{3.19}$$

Построим ситуацию равновесия по Нэшу в игре Г.

Функции выигрыша клиентов имеют прежний вид (3.14). Оптимальные размеры пакетов для клиентов 1, 2 и 3 типов определяются выражениями (3.15), (3.16). Сервер стремится максимизировать свой выигрыш (3.20) выбором распределения заданий $u = (N_1, \ldots, N_m)$ при ограничениях (3.21), (3.22).

$$C_0(N_1, \ldots, N_m, n_1^*, \ldots, n_m^*) = -\sum_{i=1}^{m} \frac{N_i}{n_i^*} \times B(n_i^*) - \sum_{i=1}^{m} \frac{N_i}{n_i^*} \times S(n_i^*, p_i) - \\ - \max_{1 \le i \le m} \left(t_e \times \frac{N_i}{Q_i p_i^{n_i^*}} \right) \longrightarrow \max_{(N_1, \ldots, N_m)} \tag{3.20}$$

$$0 \le N_i \le N,\ i = 1, \ldots, m;\ N_1 + \ldots + N_m = N;\ N_i \in \mathbb{Z},\ i = 1, \ldots, m \tag{3.21}$$

$$N_1 = \ldots = N_{m_1};\ N_{m_1+1} = \ldots = N_{m_1+m_2};\ N_{m_1+m_2+1} = \ldots = N_m \tag{3.22}$$

Тогда равновесием по Нэшу в игре (3.19) является решение задачи целочисленного нелинейного программирования (3.15), (3.16), (3.20), (3.21), (3.22).

В ходе выполнения проекта в состав Desktop Grid вошли клиенты трех типов $m_1 = 25$, $m_2 = 10$, $m_3 = 10$ со средним числом ЦП $Q_1 = 4$, $Q_2 = 8$, $Q_3 = 12$. Для решения задачи была реализована программа на языке C. При

$N = 10\,000$ получено решение:

$$N_1^* = \ldots = N_{25}^* = 133;\ N_{26}^* = \ldots = N_{35}^* = 259; N_{36}^* = \ldots = N_{45}^* = 409,$$

$$n_1^* = \ldots = n_{25}^* = 3;\ n_{26}^* = \ldots = n_{35}^* = 3; n_{36}^* = \ldots = n_{45}^* = 5.$$

Выигрыш сервера составляет $C_0^* = -3\,963.10$. Аналогично, как в предыдущем примере, можно подсчитать, что полученное решение приводит к снижению общего времени выполнения $N = 10\,000$ заданий в 2.1 раза и снижению нагрузки на сервер в 3.5 раза по сравнению с дисциплиной распределения заданий по умолчанию.

Таким образом, по результатам первого этапа проведения виртуального скрининга в системе Desktop Grid была поставлена задача оптимизации вычислительного процесса и сформулирована игровая математическая модель. На основе данных, полученных в ходе проведения виртуального скрининга, были произведены оценки вида функций и значений параметров математической модели, найдены значения функций выигрыша и сделаны выводы об эффективности решения. Был реализован и апробирован прототип программного расширения платформы BOINC для управления процессом вычислений по предложенной модели, включающий в себя сценарии командной оболочки Bash для генерации пакетов рабочих заданий и обработки результатов пакетов на сервере под управлением операционной системы Debian Linux, а также обработку пакетов заданий на клиенте.

Глава 4

Метод кумулятивных сумм для обнаружения вторжений в вычислительную систему и борьбы с ними

4.1. Постановка задачи

В данной главе рассматривается применимость метода кумулятивных сумм для обнаружения момента изменения характеристик входного потока заданий в вычислительную систему, или момента разладки. Предполагается, что поступившее в систему задание характеризуется числом 1 или 0 с вероятностью, соответственно, α_0 или $1 - \alpha_0$. Подобная характеристика может описывать различные ситуации, возникающие на практике в вычислительной системе. Например, задание:

- с вероятностью α_0 сразу запускается на вычисление, а с вероятностью $1 - \alpha_0$ становится в очередь;
- с вероятностью α_0 будет обслуживаться в нормальном режиме, а с вероятностью $1 - \alpha_0$ получает высокий приоритет обслуживания;
- с вероятностью α_0 будет выполняться ненулевое время, а с вероятностью $1 - \alpha_0$ завершится сразу же после старта;
- и т. д.

В каждом случае оба варианта будут возникать на практике и в отсутствие вторжения, и при его наличии. Однако при наличии искусственного потока заданий параметр α_0 изменится на $\alpha \neq \alpha_0$:

- вследствие приходов искусственных заданий, занимающих значительный объем ресурсов системы, увеличится доля заданий, вынужденных становиться в очередь;
- вследствие получения злоумышленником доступа к учетной записи администратора увеличится доля заданий с высоким приоритетом обслуживания;
- вследствие приходов искусственных заданий, не прошедших отладку в данной системе, увеличится доля заданий, сразу после запуска завершающихся с ошибкой;
- и т. д.

Таким образом, задача обнаружения момента начала атаки сводится к задаче обнаружения разладки — изменения характеристики α_0. В работе [29] впервые было предложено использовать метод кумулятивных сумм для обнаружения момента разладки, то есть изменения среднего значения неотрицательных, независимых, одинаково распределенных случайных величин $\{x_n\}$, $n = 1, 2, ...$ Предполагается, что до момента разладки случайные величины подчиняются распределению $F(x, \alpha_0)$, а начиная с момента разладки $\theta \geq 0$ — распределению того же вида, но с измененным параметром $x_n \sim F(x, \alpha)$, где $\alpha \neq \alpha_0$.

Согласно методу, описанному в [29], для обнаружения момента разладки строится последовательность кумулятивных сумм

$$S_n = (S_{n-1} + q(x_n))^+, \tag{4.1}$$

где $z^+ = \max(0, z)$, $q(x) = \log \frac{dF(x,\alpha)}{dF(x,\alpha_0)}$, $S_0 = s \geq 0$.

Вывод о наличии разладки делается на шаге τ_b, когда значение кумулятивной суммы впервые превышает заданное пороговое значение b:

$$\tau_b = \inf\{n > 0 : S_n \geq b\} \tag{4.2}$$

Процесс обнаружения момента разладки характеризуется двумя основными характеристиками: ARL (Average Run Length) — среднее количество наблюдений до сигнала об обнаружении разладки при условии, что по факту разладка не произойдет никогда ($\theta = \infty$); AD (Average Delay) — среднее количество наблюдений до сигнала при условии, что по факту разладка произошла в нулевой момент времени $\theta = 0$. При начальном условии $S_0 = s$ выражения для характеристик ARL и AD можно записать в следующем виде:

$$ARL = j_\infty(s) = \mathbb{E}_s\{\tau_b | \theta = \infty\} \tag{4.3}$$

$$AD = j_0(s) = \mathbb{E}_s\{\tau_b | \theta = 0\} \tag{4.4}$$

Здесь $\mathbb{E}_s\{\tau_b | \theta = \theta_1\}$ — обозначение математического ожидания при условии истинного момента разладки $\theta = \theta_1$ и начальном условии $S_0 = s$ того, что сигнал будет подан в момент τ_b.

В работе [29] показано, что функция $j_\infty(s) < \infty$ является решением интегрального уравнения типа Фредгольма. В частности, можно показать, что для кумулятивных сумм, имеющих линейную форму $S_n = (S_{n-1} + x_n - a)^+$, где a — const, уравнение (4.3) имеет вид

$$j(s) = 1 + \mathbb{E}_s\{\mathbb{I}(0 < S_1 < b) j(S_1)\} + \mathbb{P}_s\{S_1 = 0\} j(0), \quad s < b. \tag{4.5}$$

Здесь $\mathbb{P}_s$ — вероятностная мера при начальном условии $S_0 = s$, $\mathbb{I}$ — индикатор события. Уравнение для AD выглядит аналогично, с условием $\theta = 0$.

4.2. Аналитическое выражение характеристик процесса

Рассмотрим последовательность дискретных независимых одинаково распределенных с.в. x_n, $n = 1, 2, ...$, имеющих распределение Бернулли с известным

параметром $0 < \alpha_0 < 1$:

$$F(x) = \begin{cases} 0, & x < 0 \\ 1 - \alpha_0, & 0 \le x < 1 \\ 1, & x \ge 1 \end{cases}$$

В предположении, что после разладки с.в. имеют распределение Бернулли с известным параметром $\alpha \neq \alpha_0$, выражение для вычисления кумулятивных сумм принимает вид

$$S_n = \left(S_{n-1} + \ln \frac{\alpha^{x_n}(1-\alpha)^{1-x_n}}{\alpha_0^{x_n}(1-\alpha_0)^{1-x_n}} \right)^+.$$

Далее будем предполагать $\alpha > \alpha_0$. При $\alpha < \alpha_0$ полученные в работе решения остаются верными после замены $\alpha_0' = 1 - \alpha_0$, $\alpha' = 1 - \alpha$.

Для удобства последующих выкладок приведем $q(x_n)$ к линейному виду:

$$q(x_n) = \ln \frac{\alpha^{x_n}(1-\alpha)^{1-x_n}}{\alpha_0^{x_n}(1-\alpha_0)^{1-x_n}} = \left(\ln \frac{\alpha}{\alpha_0} - \ln \frac{1-\alpha}{1-\alpha_0} \right) x_n + \ln \frac{1-\alpha}{1-\alpha_0} = \gamma x_n + \beta.$$

По свойствам логарифма и в предположении $\alpha > \alpha_0$, выполняются неравенства $\gamma > 0$, $\beta < 0$ и $\gamma + \beta > 0$.

Запишем функциональное уравнение для ARL, пользуясь (4.5).

Заметим, что

$$S_1 = \begin{cases} 0, & \text{если } x_1 = 0 \text{ и } s + \beta \le 0 \\ s + \beta > 0, & \text{если } x_1 = 0 \text{ и } s + \beta > 0 \\ s + \gamma + \beta > 0, & \text{если } x_1 = 1. \end{cases}$$

Второе слагаемое (4.5):

$$\mathbb{E}_s\{I(0 < S_1 < b)j(S_1)\} = \sum_{0<k<b} \mathbb{P}_s(S_1 = k)j(S_1) =$$
$$= \mathbb{P}_s(S_1 = s+\gamma+\beta)I(s+\gamma+\beta < b)j(s+\gamma+\beta) +$$
$$+ \mathbb{P}_s(S_1 = s+\beta)I(0 < s+\beta < b)j(s+\beta) =$$
$$= \alpha_0 I(s+\gamma+\beta < b)j(s+\gamma+\beta) + (1-\alpha_0)I(0 < s+\beta < b)j(s+\beta)$$

Третье слагаемое (4.5):

$$\mathbb{P}_s(S_1 = 0) = (1-\alpha_0)I(s \le -\beta)$$

Получаем линейное функциональное уравнение

$$j(s) = 1 + \alpha_0 I(s+\gamma+\beta < b)j(s+\gamma+\beta) +$$
$$+ (1-\alpha_0)I(0 < s+\beta < b)j(s+\beta) + (1-\alpha_0)I(s \le -\beta)j(0), \quad s < b$$

Пользуясь свойствами вероятности и определением функции $j(s)$, запишем его в форме:

$$j(s) = \begin{cases} 1 + \alpha_0 j(s+\gamma+\beta) + (1-\alpha_0)j(s+\beta)^+, \ \ 0 \le s < b \\ 0, \ \ s \ge b \end{cases} \tag{4.6}$$

Аналогичный вид имеет уравнение для нахождения AD после замены коэффициента α_0 на α.

Выполнив преобразование переменных $-\beta = 1$, $b^* = \frac{b}{-\beta}$, $z = \frac{\gamma+\beta}{-\beta}$, можно

переписать уравнение 4.6 в виде системы линейных разностных уравнений:

$$j(s) = \begin{cases} 1 + (1-\alpha_0)j(s-1), & s \in [b -^* z, b^*), \\ 1 + \alpha_0 j(s+z) + (1-\alpha_0)j(s-1), & s \in [1, b^* - z), \\ 1 + \alpha_0 j(s+z) + (1-\alpha_0)j(0), & s \in [0, 1) \end{cases} \tag{4.7}$$

Свойства системы (4.7), основные из которых сформулированы в следующих шести теоремах, подробно исследуются в работе [7].

Теорема 1. Система (4.7) имеет не более одного ограниченного решения.

Теорема 2. Ограниченное решение j_s системы (4.7) положительно, $j(s) \geq \alpha_0^{-1}$ и минимум может достигаться только на $[b^* - 1, b^*]$.

Теорема 3. Решение $j(s)$ системы (4.7) кусочно-постоянно в следующем смысле: если решение имеет ограниченную производную на каком-либо множестве, то эта производная равна нулю.

Поскольку решение системы (4.7) для параметра α в точке $s = 0$ имеет смысл среднего количества наблюдений до подачи сигнала о разладке (AD), существенно следующее свойство решения:

Теорема 4. Система (4.7) не имеет неограниченных решений при рациональном z.

Доказательство. Будем рассматривать систему на любом непустом подмножестве полуинтервала $[0, b)$ при условии, что оно содержит нуль и вместе с любой точкой s оно содержит также $s + z$ и $s - 1$, если эти значения входят в $[0, b)$. Такое множество назовем допустимым.

Выберем произвольное значение s и минимальное по включению допустимое множество, содержащее s. Такое множество назовем порожденным точкой s. Покажем, что в случае рационального z это множество конечно. В самом деле, оно содержит точки $s + mz - n$ при всевозможных натуральных m и n, при условии, что $s + mz - n \in [0, b)$, а также точки $Mz - N$ при всевозможных натуральных M и N, при условии, что $Mz - N \in [0, b)$. Числа вида $mz - n$

при рациональном z — это дроби со знаменателем, не превосходящим знаменателя z. Их конечное количество на отрезке, поэтому указанных чисел — также конечное количество.

Принимая конечное множество значений, решение ограниченно и, следовательно, тождественно равно нулю, в частности, в точке s. В силу произвольности s получаем, что в рациональном случае решение также единственно и неограниченных решений быть не может. ■

Теорема 5. Система (4.7) имеет ограниченное решение при любом $z > 1$.

Теорема 6. Решение $j(s)$ системы (4.7) является невозрастающей функцией s. Свойство монотонности интуитивно ожидаемо: среднее количество наблюдений до достижения кумулятивной суммой порогового значения тем меньше, чем больше текущее значение кумулятивной суммы.

Для случая $z > \frac{1-\alpha_0}{\alpha_0}$ в [7] удалось получить оценку сверху решения в точке 0, т.е. среднего количества наблюдений до подачи сигнала о разладке:

$$j_0 \leq \frac{b^*}{\alpha_0 z - 1 + \alpha_0} + \frac{2 - \alpha_0}{(\alpha_0 z - 1 + \alpha_0)\alpha_0} \tag{4.8}$$

Полученная оценка тем точнее, чем больше значение параметра α_0.

Для рационального z в работе [7] был предложен численный алгоритм вычисления решения за конечное количество шагов. Алгоритм позволяет сократить порядок системы в $[b^*]$ раз. Для целых значений z альтернативой алгоритму является решение системы в аналитическом виде.

Для заданных α_0 и α аппроксимируем значения констант в исходном уравнении (4.6) целыми числами и решим полученное разностное уравнение, рассматривая только целые значения переменной. В зависимости от значений α_0 и α уравнение будет принимать различные формы. Рассмотрим возможные случаи.

1. Пусть $[\gamma+\beta]=[-\beta]$. Без потери общности, положим

$$b^*=\left[\frac{b}{-\beta}\right],\ -\beta=1,\ z=\frac{\gamma+\beta}{-\beta}\ ([z]=1)\,.$$

В новых обозначениях уравнение (4.6) принимает вид

$$j(s)=\begin{cases}1+\alpha_0 j(s+1)+(1-\alpha_0)j(s-1)^+, 0\le s<b^*\\ 0,\ s\ge b^*\end{cases}\tag{4.9}$$

Для целых значений $0\le s\le b^*$ запишем систему рекуррентных уравнений:

$$\begin{cases}j(0) & =1+\alpha_0 j(1)+(1-\alpha_0)j(0)\\ j(1) & =1+\alpha_0 j(2)+(1-\alpha_0)j(0)\\ \dots & \\ j(n) & =1+\alpha_0 j(n+1)+(1-\alpha_0)j(n-1)\\ \dots & \\ j(b^*-2) & =1+\alpha_0 j(b^*-1)+(1-\alpha_0)j(b^*-3)\\ j(b^*-1) & =1+\alpha_0 j(b^*)+(1-\alpha_0)j(b^*-2)\\ j(b^*) & =0\end{cases}\tag{4.10}$$

а) Пусть $\alpha_0=0.5$. Обозначив $j(0)=t$, запишем рекуррентное выражение для $j(n)$, $0<n\le b$:

$$\begin{cases}j(0) & =t\\ j(1) & =t-2\\ j(2) & =t-6\\ \dots & \\ j(n) & =t-2(1+2+\ldots+n)=t-n(n+1)\\ \dots & \\ j(b) & =t-b^*(b^*+1)\end{cases}$$

Поскольку $j(b^*) = 0$, $t = b^*(b^* + 1)$, и решение системы (4.10) имеет вид

$$j(n) = b^*(b^* + 1) - n(n + 1) \tag{4.11}$$

b) Пусть $\alpha_0 \neq 0.5$.

Утверждение 1: Решение системы (4.10) для $\alpha_0 \neq 0.5$ имеет вид

$$j(n) = \frac{(2\alpha_0 - 1)(b^* - n) + \frac{(1-\alpha_0)^{b^*+1}}{\alpha_0^{b^*}} - \frac{(1-\alpha_0)^{n+1}}{\alpha_0^n}}{(2\alpha_0 - 1)^2}, 0 \leq n \leq b^* \tag{4.12}$$

Доказательство: Пусть $j(0) = t$, $j(1) = t - \frac{1}{\alpha_0}$. Заметим, что функция (4.12) может быть записана в виде

$$j(n) = \frac{4\alpha_0^2 t - (1 - \alpha_0)(1/\alpha_0 - 1)^n -}{(2\alpha_0 - 1)^2} \\ - \frac{\alpha_0(2n - 4t + 1) + n + t + 1}{(2\alpha_0 - 1)^2}, \tag{4.13}$$

где

$$t = \frac{(1 - \alpha_0)\big((\frac{1-\alpha_0}{\alpha_0})^{b^*} - 1\big) + b^*(2\alpha_0 - 1)}{(2\alpha_0 - 1)^2}.$$

Для $n = 0$ и $n = 1$ подстановка показывает, что формула (4.13) верна. Пусть формула верна для $n = i-2$ и $n = i-1$. Подставим эти

значения в рекуррентное уравнение системы (4.10) для $j(i), i > 1$.

$$j(i) = \frac{j(i-1) - 1 - (1-\alpha_0)j(i-2)}{\alpha_0} =$$

$$= \frac{1}{\alpha_0}\left(\frac{(2\alpha_0-1)(b^*-i+1) + \frac{(1-\alpha_0)^{b^*+1}}{\alpha_0^{b^*}} - \frac{(1-\alpha_0)^i}{\alpha_0^{i-1}}}{(2\alpha_0-1)^2} - \right.$$

$$\left. - 1 - (1-\alpha_0)\frac{(2\alpha_0-1)(b^*-i+2) + \frac{(1-\alpha_0)^{b^*+1}}{\alpha_0^{b^*}} - \frac{(1-\alpha_0)^{i-1}}{\alpha_0^{i-2}}}{(2\alpha_0-1)^2}\right) =$$

$$= \frac{1}{\alpha_0}\left(\frac{\alpha_0(2\alpha_0-1)(b^*-i) + \frac{(1-\alpha_0)^{b^*+1}}{\alpha_0^{b^*-1}} - \frac{(1-\alpha_0)^{i+1}}{\alpha_0^{i-1}}}{(2\alpha_0-1)^2}\right) =$$

$$= \frac{(2\alpha_0-1)(b^*-i) + \frac{(1-\alpha_0)^{b^*+1}}{\alpha_0^{b^*}} - \frac{(1-\alpha_0)^{i+1}}{\alpha_0^{i}}}{(2\alpha_0-1)^2}. \blacksquare$$

2. Пусть $\gamma + \beta < -\beta$, т.е. $\begin{cases} 0 < \alpha_0 \le 0.5, \\ \alpha > 1 - \alpha_0, \end{cases}$ или $0.5 < \alpha_0 < 1$.

Без потери общности, положим

$$b^* = \left[\frac{b}{\gamma+\beta}\right],\ \gamma + \beta = 1,\ z = \frac{-\beta}{\gamma+\beta}.$$

Пусть $m = [z] > 1$. В новых обозначениях уравнение (4.6) принимает вид

$$j(s) = \begin{cases} 1 + \alpha_0 j(s+1) + \ (1-\alpha_0)j(s-m)^+, 0 \le s < b^* \\ 0, s \ge b^* \end{cases} \tag{4.14}$$

Для целых значений $0 \le s \le b^*$ запишем систему рекуррентных уравне-

ний:

$$\begin{cases} j(0) & = 1 + \alpha_0 j(1) + (1-\alpha_0) j(0) \\ \dots & \\ j(m) & = 1 + \alpha_0 j(m+1) + (1-\alpha_0) j(0) \\ j(m+1) & = 1 + \alpha_0 j(m+2) + (1-\alpha_0) j(1) \\ \dots & \\ j(b^*-1) & = 1 + \alpha_0 j(b^*) + (1-\alpha_0) j(b^*-m-1) \\ j(b^*) & = 0 \end{cases} \tag{4.15}$$

Для решения системы применим метод производящих функций. Введем функцию $\phi(z)$ (здесь $j_n = j(n)$):

$$\begin{aligned} \phi(z) = \sum_{n=0}^{\infty} j_n z^n = j_0 + \frac{j_0 - 1 - (1-\alpha_0) j_0}{\alpha_0} z + \\ + \frac{j_1 - 1 - (1-\alpha_0) j_0}{\alpha_0} z^2 + \ldots + \frac{j_m - 1 - (1-\alpha_0) j_0}{\alpha_0} z^{m+1} + \\ + \frac{j_k - 1 - (1-\alpha_0) j_{k-m}}{\alpha_0} z^{k+1} + \ldots = \\ = -\frac{1}{\alpha_0} \sum_{k=1}^{\infty} z^k + \frac{z}{\alpha_0} \phi(z) - \frac{1-\alpha_0}{\alpha_0} j_0 \sum_{k=1}^{m} z^k - \\ - \frac{1-\alpha_0}{\alpha_0} z^{m+1} \sum_{k=0}^{\infty} j_k z^k + j_0 = j_0 + \frac{1}{\alpha_0} \frac{z}{z-1} + \\ + \phi(z) \Big(\frac{z}{\alpha_0} - \frac{1-\alpha_0}{\alpha_0} z^{m+1} \Big) - \frac{1-\alpha_0}{\alpha_0} j_0 \frac{z^{m+1} - z}{z-1} \end{aligned}$$

Получаем следующий вид производящей функции:

$$\phi(z) = \frac{j_0 \alpha_0 (z-1) - j_0 (1-\alpha_0)(z^{m+1} - z) + z}{(\alpha_0 - z + (1-\alpha_0) z^{m+1})(z-1)}$$

Представим правую часть выражения в виде простых дробей для после-

дующего разложения производящей функции в ряд.

$$\frac{z}{(\alpha_0 - z + (1-\alpha_0)z^{m+1})(z-1)} - j_0 \frac{(1-\alpha_0)z^{m+1} - z + \alpha_0}{(z-1)(\alpha_0 - z + (1-\alpha_0)z^{m+1})} = \\ = \frac{z}{(z-1)((1-\alpha_0)z^{m+1} + \alpha_0 - z)} + j_0 \frac{1}{1-z} \quad (4.16)$$

Для разложения на множители знаменателя дроби в правой части докажем следующее утверждение.

Утверждение 2: У многочлена $(1-\alpha_0)z^{m+1} + \alpha_0 - z = p(z)$ нет кратных корней за исключением случая, когда $\alpha_0 = m/(m+1)$, а в последнем случае единственным кратным корнем, с кратностью 2, является единица.

Доказательство: Если число ξ является кратным корнем многочлена $p(z) = 0$, то оно является также корнем производной $p'(z) = (1-\alpha_0)(m+1)z^m - 1 = 0$. Чтобы выяснить наличие кратных корней, найдем наибольший общий делитель $p(z)$ и $p'(z)$, используя алгоритм Эвклида.

$$p(z) = p'(z)\frac{z}{m+1} + \alpha_0 - z\frac{m}{m+1} = p'(z)q_1(z) + r_1(z)$$

$$p'(z) = r_1(z)\sum_{k=1}^{m}(\alpha_0 - 1)\alpha_0^{m-k}\frac{(m+1)^{m-k+2}}{m^{m-k+1}}z^{k-1} + \\ + \alpha_0^m(1-\alpha_0)\frac{(m+1)^{m+1}}{m^m} - 1 = r_1(z)q_2(z) + r_2,$$

где r_2 — *const*. Если $r_2 \neq 0$, т.е. $\alpha_0 \neq \frac{m}{m+1}$, то кратных корней нет (можно показать, что остальные допустимые значения α_0 не обращают r_2 в 0).

Пусть $\alpha_0 = \frac{m}{m+1}$, тогда $p(z) = \frac{m}{m+1} - z + \frac{z^{m+1}}{m+1}$, $p'(z) = z^m - 1$. Пусть число ξ — корень производной $p'(z) = 0$. Если ξ является и корнем многочлена

$p(z) = 0$, то

$$\frac{m}{m+1} - \xi + \frac{\xi^{m+1}}{m+1} = \frac{m}{m+1} - \xi\frac{m}{m+1} = 0 \Rightarrow \xi = 1.$$

Разделив $p'(z)$ на $z-1$, получим многочлен $\sum_{k=0}^{m-1} z^k$. Единица не является его корнем при $m > 1$. Следовательно, при $\alpha_0 = \frac{m}{m+1}$ корень исходного многочлена $z = 1$ имеет кратность 2. ■

Таким образом, для $\alpha_0 \neq \frac{m}{m+1}$ разложение $\phi(z)$ на простые дроби имеет вид:

$$\phi(z) = \frac{1}{1-\alpha_0} \times \frac{z}{(z-1)^2(\sum_{i=1}^{m} z^i - \frac{\alpha_0}{1-\alpha_0})} + \\ + j_0\frac{1}{1-z} = \frac{1}{1-\alpha_0} \times \Big(\frac{A}{(z-1)^2} + \frac{B}{z-1} + \\ + \frac{C_1}{z-z_1} + \ldots + \frac{C_m}{z-z_m}\Big) + j_0\frac{1}{1-z},$$

где $z_1, ..., z_m$ — корни многочлена $\sum_{i=1}^{m} z^i - \frac{\alpha_0}{1-\alpha_0}$, а константы A, B, C_1,...,C_m заданы системой линейных уравнений, порожденной разбиением на простые дроби:

$$\begin{cases} B + \sum\limits_{i=1}^{m} C_i = 0 \\ A + \sum\limits_{i=1}^{m} C_i(z_i - 1) = 0 \\ A + \sum\limits_{i=1}^{m} C_i z_i(z_i - 1) = 0 \\ \ldots \\ A + \sum\limits_{i=1}^{m} C_i z_i^{m-2}(z_i - 1) = 0 \\ A - \frac{B}{1-\alpha_0} + \sum\limits_{i=1}^{m} C_i(\frac{-2\alpha_0}{(1-\alpha_0)z_i} + \frac{z_i^{m-1}-1}{z_i-1}) = 1 \\ A\frac{\alpha_0}{\alpha_0-1} + B\frac{\alpha_0}{1-\alpha_0} + \sum\limits_{i=1}^{m} C_i\frac{\alpha_0}{(1-\alpha_0)z_i} = 0 \end{cases}$$

Производящая функция раскладывается в ряд следующим образом:

$$\phi(z) = \sum_{n=0}^{\infty} \Big(\frac{A(n+1)}{1-\alpha_0} + j_0 - \frac{B}{1-\alpha_0} - \sum_{i=1}^{m} \frac{C_i}{1-\alpha_0} \frac{1}{z_i^{n+1}} \Big) z^n = \sum_{n=0}^{\infty} j_n z^n$$

Найдем j_0 из уравнения $j(b^*) = 0$:

$$j_0 = \sum_{i=1}^{m} \frac{C_i}{1-\alpha_0} \frac{1}{z_i^{b^*+1}} - \frac{A}{1-\alpha_0}(b^*+1) + \frac{B}{1-\alpha_0}$$

Таким образом, при $\alpha_0 \neq \frac{m}{m+1}$ решением системы (4.15) является функция

$$j(n) = \frac{A}{1-\alpha_0}(n-b^*) + \sum_{i=1}^{m} \frac{C_i(1-z_i^{b^*-n})}{(1-\alpha_0)z_i^{b^*+1}} \tag{4.17}$$

Для $\alpha_0 = \frac{m}{m+1}$ разложение $\phi(z)$ на простые дроби имеет вид:

$$\phi(z) = \frac{1}{1-\alpha_0} \times \frac{z}{(z-1)^3(\sum_{i=1}^{m-1} iz^{m-i} + m)} + \\ + j_0 \frac{1}{1-z} = \frac{1}{1-\alpha_0} \times \Big(\frac{Az^2+Bz+C}{(z-1)^3} + \\ + \frac{D_1}{z-z_1} + \ldots + \frac{D_{m-1}}{z-z_{m-1}} \Big) + j_0 \frac{1}{1-z},$$

где $z_1, ..., z_{m-1}$ — корни многочлена $\sum_{i=1}^{m-1} iz^i + m$, а константы A, B, C, D_1,...,D_{m-1} заданы системой линейных уравнений, порожденной разбиением на простые дроби:

$$
\begin{cases}
2A + B + \sum\limits_{i=1}^{m-1} D_i(z_i - 1) &= 0 \\
3A + 2B + C + \sum\limits_{i=1}^{m-1} D_i z_i(z_i - 1) &= 0 \\
\dots & \\
(m-2)A + (m-3)B + (m-4)C + & \\
+(-1)^{m-4} \sum\limits_{i=1}^{m-1} D_i z_i^{m-4}(1 - z_i) &= 0 \\
(m-1)A + (m-2)B + (m-3)C + & \\
+ \sum\limits_{i=1}^{m-1} D_i \big(\frac{6-3m+(3m-3)z_i - 3z_i^{m-1}}{(z_i-1)^2} - \frac{m}{z_i} \big) &= 0 \\
mA + (m-1)B + (m-2)C + & \\
+ \sum\limits_{i=1}^{m-1} D_i \big(\frac{3m}{z_i} + \frac{2-m+(m-1)z_i - z_i^{m-1}}{(z_i-1)^2} \big) &= 0 \\
mB + (m-1)C + & \\
+ \sum\limits_{i=1}^{m-1} D_i \big(\frac{2-m+(m-1)z_i - z_i^{m-1}}{(z_i-1)^2} - \frac{3m}{z_i} \big) &= 1 \\
mC + \sum\limits_{i=1}^{m-1} D_i \frac{m}{z_i} &= 0
\end{cases}
$$

Производящая функция раскладывается в ряд следующим образом:

$$
j(n) = \sum_{n=0}^{\infty} \Big(j_0 + \frac{An(1-n)}{2(1-\alpha_0)} - \frac{Bn(1+n)}{2(1-\alpha_0)} - \\
- \frac{C(n+2)(n+1)}{2(1-\alpha_0)} - \frac{1}{1-\alpha_0} \sum_{i=1}^{m-1} \frac{D_i}{z_i^{n+1}} \Big) z^n = \sum_{n=0}^{\infty} j_n z^n
$$

Найдем j_0 из уравнения $j(b^*) = 0$:

$$
j_0 = \sum_{i=1}^{m-1} \frac{D_i}{z_i^{b^*+1}} - \frac{1}{2} \big(Ab^*(1-b^*)(m+1) - \\
- b^*(1+b^*)B(m+1) - (1+b^*)(2+b^*)C(m+1) \big)
$$

Таким образом, при $\alpha_0 = \frac{m}{m+1}$ решением системы (4.15) является функция

$$j(n) = \frac{1}{2}(m+1)\big(A(n(1-n) - b^*(1-b^*)) - \\ - B(n(n+1) - b^*(b^*+1)) - C((n+2)(n+1) - \\ - (b^*+2)(b^*+1))\big) + \sum_{i=1}^{m-1} \frac{D_i(1 - z_i^{b^*-n})}{z_i^{b^*+1}} \quad (4.18)$$

3. Пусть $\gamma + \beta > -\beta$, т.е. $\begin{cases} 0 < \alpha_0 < 0.5, \\ \alpha_0 < \alpha < 1 - \alpha_0. \end{cases}$

Без потери общности, положим

$$b^* = \left[\frac{b}{-\beta}\right],\ \beta = -1,\ z = \frac{\gamma + \beta}{-\beta}.$$

Пусть $m = [z] > 1$. В новых обозначениях уравнение (4.6) принимает вид

$$j(s) = \begin{cases} 1 + \alpha_0 j(s+m) + (1-\alpha_0) j(s-1)^+, 0 \le s < b^* \\ 0, s \ge b^* \end{cases} \quad (4.19)$$

Для целых значений $0 \le s \le b^*$ запишем систему рекуррентных уравнений:

$$\begin{cases} j(0) & = 1 + \alpha_0 j(m) + (1-\alpha_0) j(0) \\ j(1) & = 1 + \alpha_0 j(m+1) + (1-\alpha_0) j(0) \\ j(2) & = 1 + \alpha_0 j(m+2) + (1-\alpha_0) j(1) \\ \dots & \\ j(b^*-m) & = 1 + \alpha_0 j(b^*) + (1-\alpha_0) j(b^*-m-1) \\ \dots & \\ j(b^*-1) & = 1 + \alpha_0 j(b^*-1+m) + (1-\alpha_0) j(b^*-2) \\ j(b^*) & = 0 \end{cases} \quad (4.20)$$

Для решения системы используем метод характеристических уравнений. Получим из (4.19) однородное рекуррентное уравнение, введя замену

$$j_n = J_n + \frac{n}{1-\alpha_0(m+1)} \text{ для } \alpha_0 \neq \frac{1}{m+1}$$

$$\text{или } j_n = J_n - \frac{n^2}{m} \text{ для } \alpha_0 = \frac{1}{m+1}:$$

$$J_n = \alpha_0 J_{n+m} + (1-\alpha_0) J_{n-1} \tag{4.21}$$

Запишем характеристическое уравнение для полученного уравнения:

$$\alpha_0 z^{m+1} - z + 1 - \alpha_0 = 0 \tag{4.22}$$

Из Утверждения 3, с заменой $\alpha_0 = 1-\alpha_0'$, следует, что характеристическое уравнение не имеет кратных корней при $\alpha_0 \neq \frac{1}{m+1}$. Тогда решение (4.19) имеет вид

$$j(n) = \frac{n}{1-\alpha_0(m+1)} + \sum_{i=0}^{m} C_i z_i^n, \tag{4.23}$$

где z_0,...,za_m — корни характеристического уравнения ($z_0 = 1$), а константы C_0,...,C_m заданы системой линейных уравнений, порожденной системой (4.20):

$$\begin{cases} \sum\limits_{i=1}^{m} C_i(1-z_i^m) = \frac{1}{\alpha_0} + \frac{m}{1-\alpha_0(m+1)} \\ \alpha_0 C_0 + \sum\limits_{i=1}^{m} C_i(z_i^{b^*-1} - (1-\alpha_0) z_i^{b^*-2}) = \frac{\alpha_0(b^*-1+m)}{\alpha_0-1+\alpha_0 m} \\ \dots \\ \alpha_0 C_0 + \sum\limits_{i=1}^{m} C_i(z_i^{b^*-m} - (1-\alpha_0) z_i^{b^*-m-1}) = \frac{\alpha_0 b^*}{\alpha_0-1+\alpha_0 m} \end{cases}$$

При $\alpha_0 = \frac{1}{m+1}$ характеристическое уравнение имеет корень кратности 2,

и решение (4.19) принимает вид

$$j(n) = -\frac{n^2}{m} + C_0 + C_1 n + \sum_{i=2}^{m} C_i \lambda_i^n, \tag{4.24}$$

где λ_2,...,λ_m — корни характеристического уравнения, отличные от 1, а константы C_0,...,C_m заданы системой линейных уравнений

$$\begin{cases} C_1 m + \sum\limits_{i=2}^{m} C_k(\lambda_i^{b-m} & -1) = -1 \\ \frac{C_0}{m+1} + \frac{b}{m+1} C_1 + & \sum\limits_{i=2}^{m} C_i(1 - \frac{m}{m+1})(\lambda_i^{b-m} - \frac{m\lambda_i^{b-m-1}}{m+1}) = \frac{b^2}{m+m^2} \\ \dots & \\ \frac{C_0}{m+1} + \frac{b+m-1}{m+1} C_1 + & \sum\limits_{i=2}^{m} C_i(1 - \frac{m}{m+1})(\lambda_i^{b-1} - \frac{m\lambda_i^{b-2}}{m+1}) = \frac{(b+m-1)^2}{m+m^2} \end{cases}$$

$j(0)$ является функцией параметров b и α_0, $j_0 = j_0(b, \alpha_0)$. Решение уравнения (4.6), $j(s)$, описывает среднее количество наблюдений до ложной тревоги согласно правилу (4.2), в зависимости от порогового значения b, при условии $S_0 = 0$.

Функция $j_0(b, \alpha_0)$ является возрастающей по b для любых значений α_0. Следовательно, задача минимизации $AD = j_0(b)$ при условии, что среднее количество наблюдений до ложной тревоги будет не менее заданного числа ($ARL \geq a$), сводится к нахождению

$$\mathbf{b} = min\{b : j_\infty(b) \geq a\}. \tag{4.25}$$

Среднее количество наблюдений до обнаружения разладки определяется значением $j_0(\mathbf{b})$.

4.2.1. Пример аналитического вычисления характеристик ARL и AD

Пусть $\alpha_0 = 0.1$, $\alpha = 0.6$. Тогда $q(x_n) = 2.603\, x_n - 0.811$, $m = 2$, и $j(s)$ является решением уравнения (4.23). В таблице (4.1) приведены значе-

ния ARL=$j_\infty(b^*)$ для ряда значений параметра $b^* \geq m+1$ при $S_0 = 0$. Таким образом, для ARL=1000 оптимальное значение $b^* = 7$. Тогда значение $b = b^*/-\beta = 8.63$ минимизирует характеристику AD, которая принимает при этом значение 10.75, согласующееся с оценкой сверху $AD \leq 11.8454$, вычисленной по формуле (4.8).

b^*	3	4	5	6	7	8	9
ARL	62.62	102.56	377.13	763.04	2310.51	5211.67	14459.4

Таблица 4.1. Значение характеристики ARL для целых значений параметра b^*.

В таблице (4.2) представлены примеры значений характеристик ARL и AD, вычисленные по полученным формулам. Прямое решение уравнения (4.6) приводит к аналогичным результатам.

α_0	α	b	ARL	AD
0.1	0.3	4.02	1000	26.6
0.1	0.6	8.63	1000	10.75
0.4	0.7	6.16	1000	25.6
0.45	0.8	5.18	1000	18.8
0.45	0.9	6.24	1000	12.2
0.6	0.8	5.47	1000	43.8
0.6	0.9	5.68	1000	21.7

Таблица 4.2. Значения характеристик ARL и AD для заданных параметров распределения до и после разладки, α_0 и α.

Аналитические решения уравнения (4.6) были получены в предположении целочисленности переменной z. Исследуем характер погрешности, возникающей при округлении нецелочисленного z до ближайшего целого. В Таблице 4.3 приведены значения ARL для ряда значений параметров b^* и z. Представленные данные иллюстрируют, что в рассмотренном примере округление z до ближайшего целого не привело к изменению целого порогового значения b^*, определяемого условием (4.25).

На Рис. 4.1 представлены графики решений исходной системы (4.7) при $z = 3.36$ и двух систем, полученных заменой параметра $m = \lfloor z \rfloor$ и $m = \lceil z \rceil$. Ри-

$b^* = 6$						
z	1.8	1.9	2.0	2.1	2.2	2.3
ARL	2 442.17	2 311.25	763.06	762.87	695.12	574.34
$b^* = 7$						
z	1.8	1.9	2.0	2.1	2.2	2.3
ARL	5 916.5	5 215.9	2 310.61	2 306.89	1 759.26	1 214.5
$b^* = 8$						
z	1.8	1.9	2.0	2.1	2.2	2.3
ARL	19 763.56	14 499.99	5 211.76	5 186.51	3 331.98	2 224.72

Таблица 4.3. Значения характеристики ARL для различных значений параметров b^* и z.

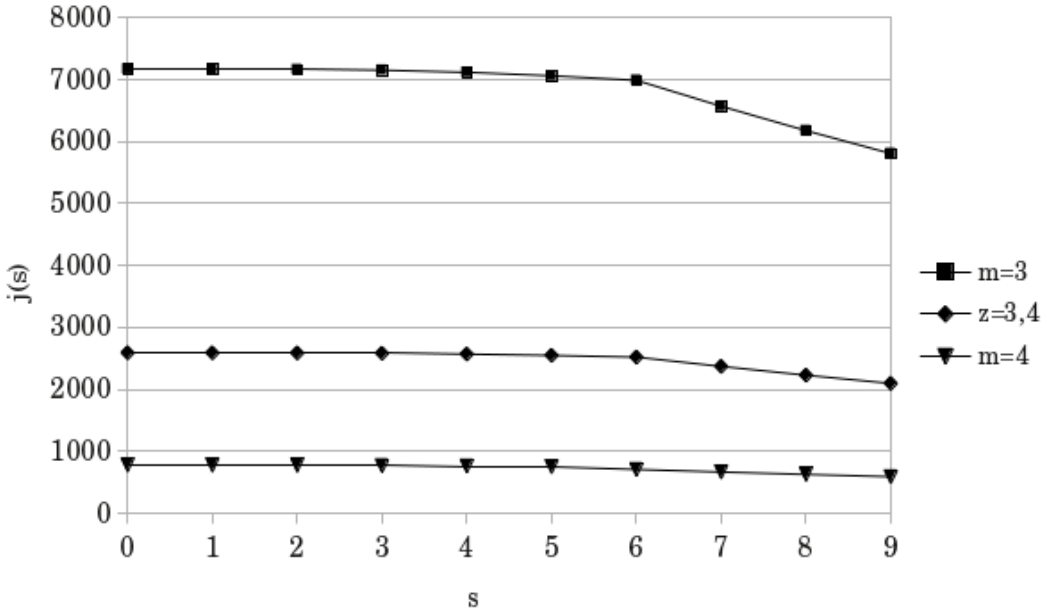

Рис. 4.1. Решения исходной и целочисленных систем ($b = 10, \alpha_0 = 0.06$).

сунок иллюстрирует, что разница решений в т. $j(0)$ может оказаться существенной для дальнейшего выбора порогового значения b. Поэтому вопрос оценки погрешности в общем случае требует дальнейшего исследования.

4.3. Изменение протокола обслуживания

После обнаружения вторжения возможно прекращение или ограничение обслуживания входящих заявок до момента отключения атакующего от системы. Однако более целесообразен и распространен на практике (см., например, [107–109]) подход, заключающийся в разделении входного потока на регуляр-

ный и искусственный и последующий отказ в обслуживании искусственным заявкам, т.е. изменение протокола обслуживания.

Мы предлагаем изменение протокола обслуживания на основе следующей идеи. Атакующему неизвестен закон распределения регулярного входного потока и его параметры. В наших предположениях это единственное отличие искусственных задач от регулярных.

Обозначим как z_n некоторую характеристику задачи в наблюдаемом входном потоке, $n = 1, 2, ...$ Пусть в регулярном потоке заданий z_n распределена по известному нам закону с медианой m. Введем случайную величину x_n, принимающую значение 0, если $z_n < m$, и 1, если $z_n \geq m$. Тогда при регулярном потоке частота появления 0 и 1 примерно одинакова, и x_n имеет распределение Бернулли с $\alpha_0 = 0.5$. При вторжении параметр распределения α меняется в большую или меньшую сторону относительно $\alpha_0 = 0.5$. Если после вторжения характеристики задач в среднем становятся больше, чем в регулярном потоке, то единиц в новой последовательности больше, что соответствует $\alpha > 0.5$, и после обнаружения вторжения в новом протоколе обслуживания следует обслуживать заявки, лежащие в среднем левее генерируемой случайной величины. Верно и обратное.

Изменим протокол обслуживания следующим образом. Пусть задания, поступившие за некоторый промежуток времени, накапливаются в буфере, где могут проходить некоторую обработку, и затем выборочно обслуживаются. Для обслуживания задания выбираются из буфера по одному в определенном порядке. Для обеспечения высокого качества обслуживания регулярных пользовательских заданий при наличии искусственного потока необходимо выбирать из буфера регулярные задания, игнорируя искусственные. Для этого предлагается моделировать случайную величину, имеющую такое же распределение, как и регулярный входной поток заданий. После каждой реализации случайной величины из буфера выбирается задание, по рассматриваемой характеристике ближайшее к ней слева или справа (Рис. 4.2).

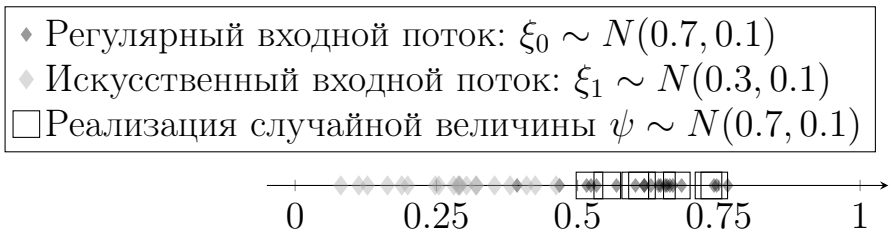

Рис. 4.2. Реализация случайной величины для выбора заданий из буфера.

При реализации данной дисциплины обслуживания возможно возникновение ошибок двух типов: первый — обслуживание искусственной задачи, второй — отказ в обслуживании регулярной. На практике количество ошибок второго типа должно быть сведено к минимуму, поскольку в противном случае цель источника DOS-атаки будет частично или полностью достигнута. При предлагаемой дисциплине обслуживания на количество ошибок оказывает влияние как длительность интервала времени, в течение которого в буфере накапливаются задания, так и количество заданий, которые будут выбираться из буфера для обслуживания. Выбор значений данных параметров зависит от характеристик входного потока заданий. Очевидно, что при выборе слишком малого размера буфера в него с большой вероятностью попадут только искусственные задания, и на их обслуживание будут затрачены ресурсы вычислительной системы, в то время как регулярные заявки будут ждать попадания в следующий буфер. Похожая ситуация возможна и при выборе слишком большого размера буфера.

Выбор значений параметров для алгоритма обслуживания зависит от вида и параметров распределений регулярного и искусственного потоков, мощности вычислительной системы и допустимого количества ошибок первого и второго типов. В целях исследования влияния значений параметров на эффективность алгоритма обслуживания были проведено имитационное моделирование вычислительного процесса, результаты которого приводятся в следующем разделе.

4.4. Вычислительные эксперименты

4.4.1. Выражение характеристик процесса обнаружения атаки

В качестве примера входного потока заданий в вычислительную систему был рассмотрен фрагмент истории функционирования крупного вычислительного кластера LLNL Atlas[1] в период с 18 декабря 2006 г. по 9 мая 2007 г. В состав данного кластера, работавшего в Ливерморской национальной лаборатории им. Э. Лоуренса (США) до июня 2012 г., входили 1152 8-процессорных вычислительных узла. Управление заданиями производилось с использованием системы управления Slurm[2]. Обработанный регистрационный файл был получен из открытого Интернет-архива[3].

В течение рассматриваемого периода на кластере было зафиксировано несколько аномальных всплесков активности отдельных пользователей, которые перечислены и кратко прокомментированы в разделе открытого Интернет-архива, посвященном рассматриваемому регистрационному файлу. Пример такого всплеска приведен на Рис. 4.3. На протяжении 11 часов один из пользователей кластера запускал на выполнение множество заданий, требующих относительно большого количества процессоров.

До начала данного всплеска активности среднее количество запрашиваемых процессоров на протяжении 3,4 месяцев составило 771.05, а медиана выборки — $m_1 = 64$. Однако в присутствие таких заданий среднее количество запрашиваемых процессоров возросло до 1720.42, а медиана выборки составила $m_2 = 1800$. Таким образом, имело место существенное изменение характеристик входного потока заданий, которое может быть рассмотрено как атака.

В предположении, что каждое поступающее в систему задание характеризовалось количеством запрошенных процессоров, входной поток может быть рассмотрен как серия наблюдений за значениями случайной величины x_n, под-

[1] Linux at Livermore. URL: https://computing.llnl.gov/linux/index.html
[2] Simple Linux Utility for Resource Management. URL: http://slurm.schedmd.com
[3] Parallel Workload Archive. URL: http://www.cs.huji.ac.il/labs/parallel/workload

b^*	b	ARL	AD	b^*	b	ARL	AD
6	187.68	126	6.22	11	344.08	3996	11.6
7	218.96	254	7.3	12	375.36	7932	12.7
8	250.24	508	8.4	13	406.64	15740	13.7
9	281.52	1012	9.4	14	437.92	31228	14.8
10	312.8	2012	10.5	15	469.2	61950	15.9

Таблица 4.4. Значения параметров b, ARL и AD для различных пороговых значений b^*

чиненной закону распределения Бернулли. Если количество процессоров, запрашиваемое заданием, меньше m_1, то x_n принимает значение 0, в противном случае 1. Тогда регулярный поток заданий имеет параметр распределения $\alpha_0 = 0.5$, а в присутствии искусственного потока (всплеска активности пользователя) параметр изменяется на $\alpha > \alpha_0$. В качестве значения параметра была выбрана частота появления заданий, запрашивающих не меньше m_1 процессоров, в присутствии искусственного потока: $\alpha = \frac{739}{745} \approx 0.99$. Таким образом, имеем

$$\gamma + \beta \approx 0.6831, \ -\beta \approx 3.9120.$$

Поскольку $\gamma + \beta < -\beta$, уравнение (4.5) при переходе к целочисленной системе примет форму 2, то есть (4.14), и его решением будет являться функция (4.17). Численные значения решения в т. $s = 0$, то есть ARL, зависящие от параметра b^*, приведены в Таблице 4.4. Для каждого b^* приведено соответствующее значение $b = -\beta b^*$, полученное обратной заменой переменных, и соответствующее значение характеристики AD.

Используя регистрационный файл системы LLNL Atlas, построим последовательность кумулятивных сумм вида 4.1 для заданий, поступающих в систему на рассматриваемом интервале времени. В качестве индекса n в 4.1 примем порядковый номер задания, в качестве величины x_n — количество запрашиваемых процессоров. График динамики кумулятивной суммы представлен на Рис. 4.4. Рисунок иллюстрирует рост среднего значения кумулятивной суммы, начиная с момента всплеска активности пользователя.

Штриховой линией на графике показано значение порога $b = 31.28$, выбранного по Таблице 4.4 в предположении, что желаемое среднее количество наблюдений до подачи ложной тревоги $ARL \geq 500$. При имитационном моделировании процесса обнаружения атаки было зафиксировано 27 ложных тревог, среднее количество заданий между которыми составило 526.7, что согласуется с полученным аналитически результатом.

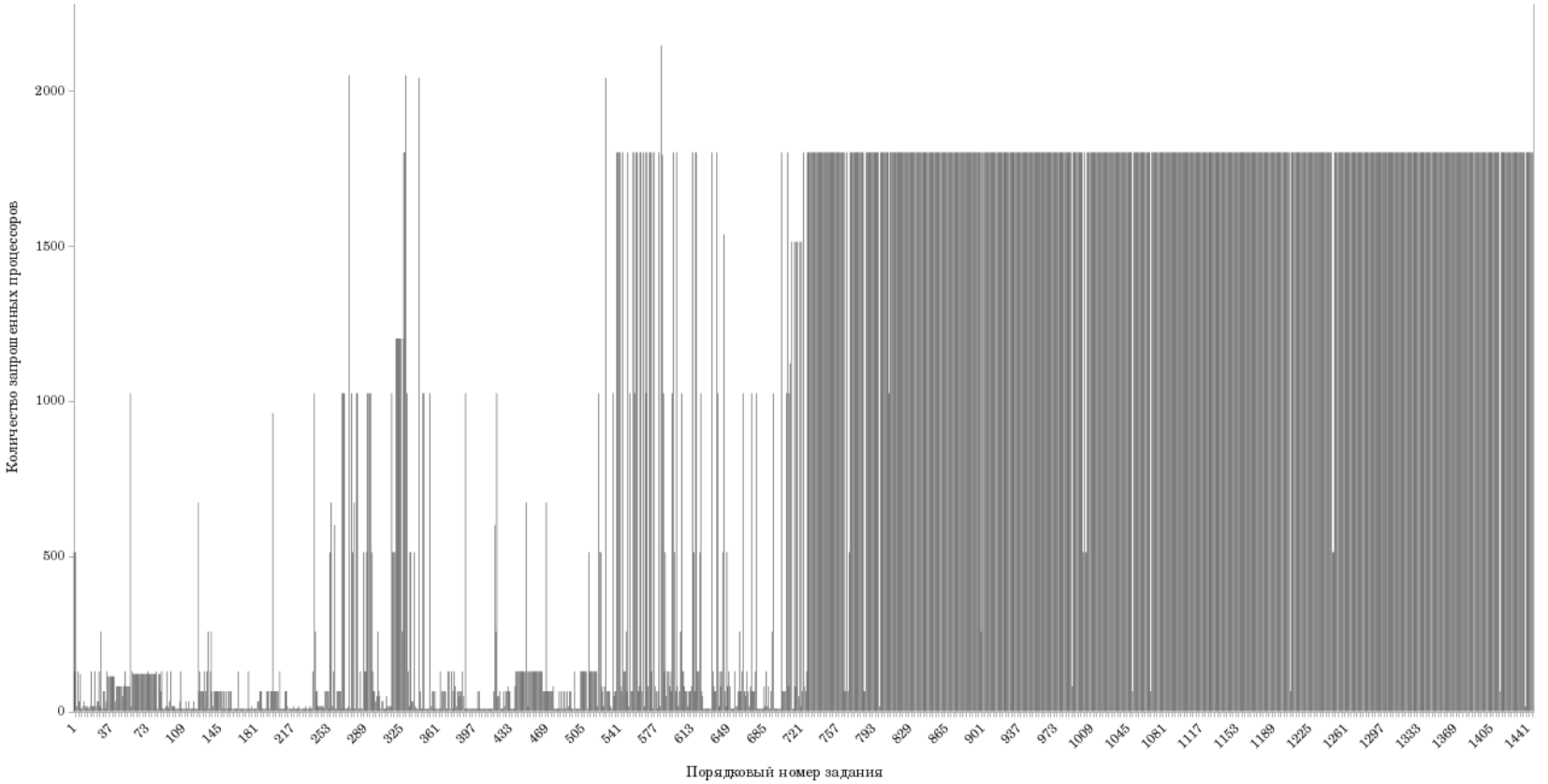

Рис. 4.3. Зависимость количества запрашиваемых процессоров от порядкового номера задания. Фрагмент истории функционирования кластера LLNL Atlas, иллюстрирующий всплеск активности отдельного пользователя на протяжении 11 часов работы.

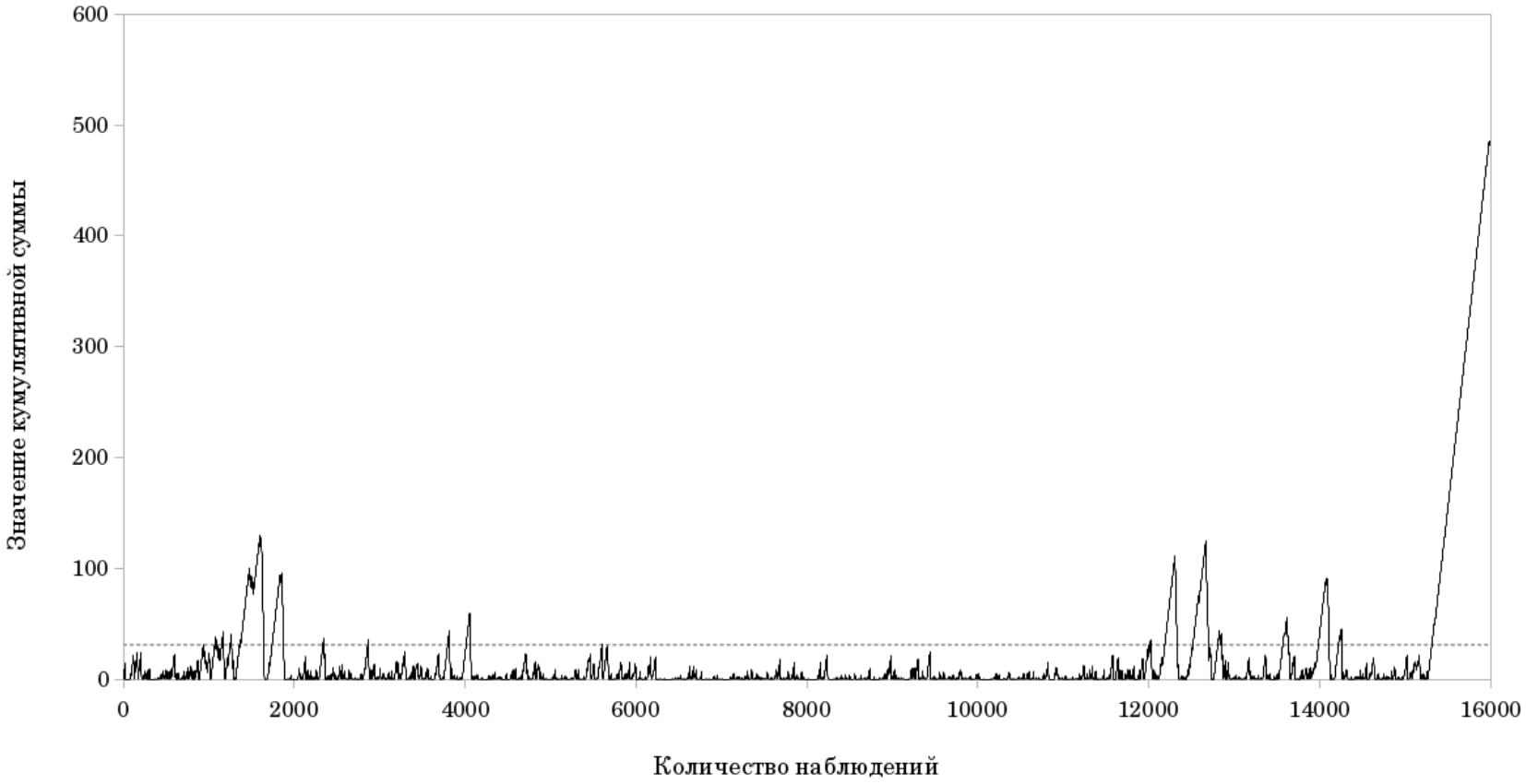

Рис. 4.4. Динамика кумулятивной суммы на протяжении исследуемого периода работы вычислительного кластера LLNL Atlas.

4.4.2. Изменение протокола обслуживания

При фиксированном виде и значениях параметров распределений регулярного и искусственного потоков были смоделированы потоки из $N = 11\,000$ заданий, среди которых было $N_r = 1\,000$ регулярных и $N_a = 10\,000$ искусственных (в предположении, что при наличии атаки искусственные заявки поступают в 10 раз чаще регулярных). Для обслуживания из буфера выбиралось $\mathbf{M}$ заданий. На Рис. 4.5 приведена зависимость доли ошибок второго типа от обслуживаемой доли буфера в предположении, что интересующая нас характеристика регулярного потока заданий имеет нормальное распределение $N(0.7, 0.1)$, а характеристика потока заданий при наличии атаки — $N(0.3, 0.1)$.

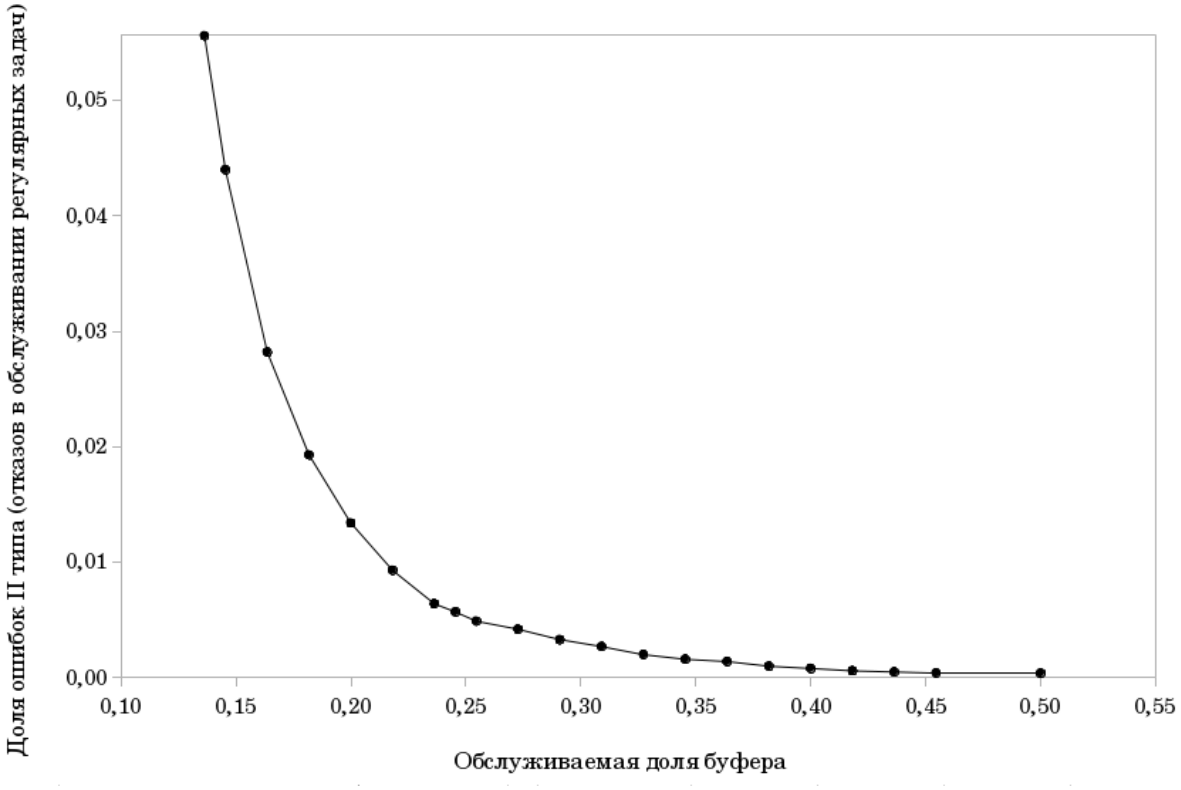

Рис. 4.5. Зависимость доли ошибок II типа от обслуживаемой доли буфера M/N.

Результаты экспериментов показывают, что при наличии точной информации о виде и параметрах распределения характеристик заданий предложенное изменение протокола обслуживания позволяет обрабатывать входящий поток заявок с низкой долей ошибок и низкой средней задержкой, приемлемыми на практике.

Однако на практике точные значения параметров распределения потока заданий определить очень сложно, в частности, потому, что с течением времени могут изменяться самые разные характеристики вычислительного процесса, косвенно влияющие на параметры. Например, с началом учебного года в потоке заданий может появиться значительная доля малопроцессорных «учебных» заданий пользователей-студентов, увеличив тем самым интенсивность входного потока и уменьшив среднее количество запрашиваемых ресурсов. Параметр регулярного потока заданий изменит свое истинное значение с известного нам μ на $\mu^* = \mu + \delta$. На Рис. 4.6 изображена зависимость доли ошибок второго типа от величины изменения параметра, δ. Как и следовало ожидать, количество ошибок становится больше при возрастании абсолютной величины параметра.

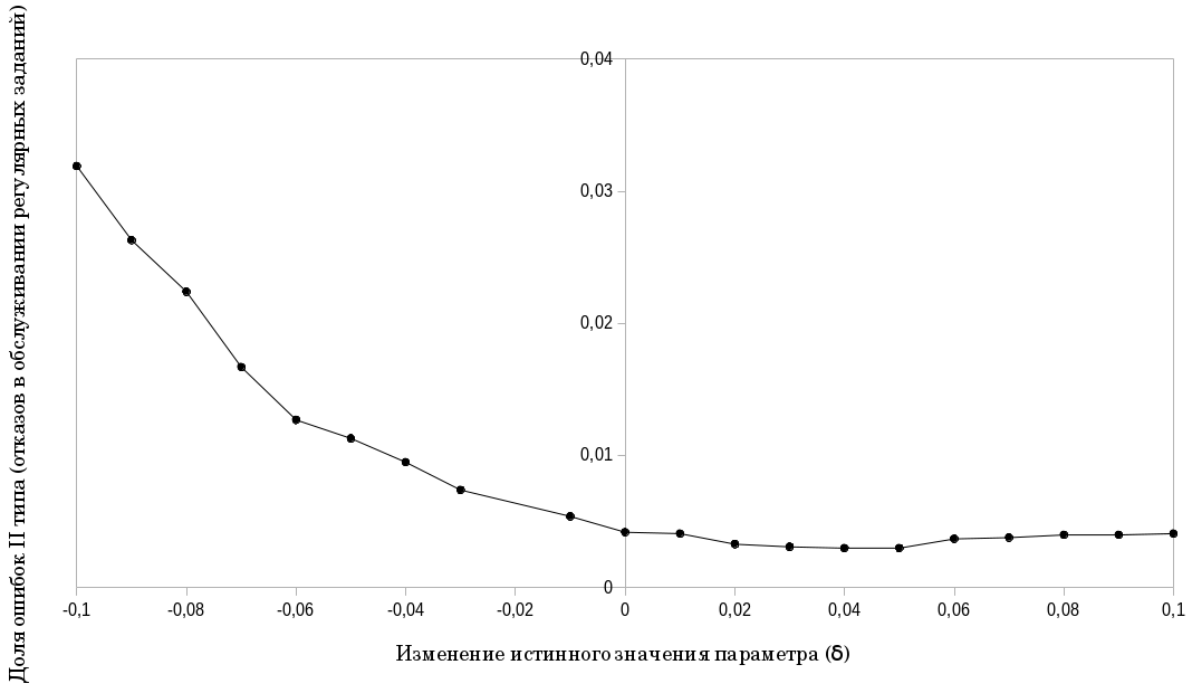

Рис. 4.6. Зависимость доли ошибок II типа от изменения истинного значения параметра δ.

Поток заданий был смоделирован с прежними значениями $N = 11\,000$, $N_r = 1\,000$, $N_a = 10\,000$, с размером обрабатываемой части буфера $M = 3500$, истинным распределением характеристики входного потока заданий $N(0.7 + \delta, 0.1)$, распределением при наличии атаки $N(0.3, 0.1)$.

Заключение

В первой главе диссертации приводится обзор литературы по теме исследования, основные методы классификации высокопроизводительных вычислительных систем и общее описание основных задач, возникающих при управлении вычислительным процессом в них.

Во второй главе предложена модификация алгоритма планирования заданий на вычислительном кластере с применением процедуры Backfill и выполнена оценка ее эффективности. В исходном варианте алгоритма решение о применении процедуры Backfill принимается на основе точечной оценки длительности выполнения задания. Такие оценки могут значительно отличаться от фактической длительности, что приводит к снижению эффективности алгоритма планирования. В описанной модификации алгоритма решение принимается на основе оценки вероятности задержки первоочередного задания. Процедура Backfill применяется, если вероятность задержки не превышает заданное пороговое значение. Приведены результаты имитационного моделирования вычислительного процесса с различными видами и параметрами распределений характеристик заданий. На смоделированных потоках заданий проиллюстрирована работоспособность алгоритма и приведены выводы о ее применимости на практике.

В третьей главе приводится описание задачи виртуального скрининга и системы Desktop Grid, реализованной для ее решения в рамках совместного научно-исследовательского проекта между Институтом прикладных математических исследований КарНЦ РАН и Любекским Институтом экспериментальной дерматологии (до мая 2014 г. — Клиникой дерматологии, аллергологии и венерологии) при университете г. Любек, Германия. Предлагается теоретико-игровая модель процесса управления потоком заданий в централизованной системе распределенных вычислений типа Desktop Grid. В качестве критерия оптимизации принимается минимизация общей нагрузки на сервер. Использованные

в ходе вычислительных экспериментов оценки вида функций и значений параметров модели получены по результатам проведения виртуального скрининга. Приводятся примеры нахождения ситуации равновесия аналитически, а также результаты вычислительных экспериментов, которые иллюстрируют работоспособность модели.

В четвертой главе приводятся аналитические выражения для характеристик процесса обнаружения DoS-атаки на вычислительную систему, в частности, среднего количества наблюдений до ложной тревоги и средней задержки до обнаружения момента атаки. Представлены результаты имитационного моделирования входного потока заданий в вычислительной системе, в котором наблюдается DoS-атака, и предложен протокол обслуживания, при котором из смешанного входного потока в присутствие атаки выделяются задания, ассоциированные с нормальным режимом функционирования системы.

Литература

1. Мазалов В. В., Никитина Н. Н. Метод кумулятивных сумм для обнаружения вторжений и борьбы с ними // Программирование. — 2014. — № 6. — С. 335–342.

2. Численная идентификация модели дегидрирования в грид-системе на базе BOINC / И. А. Чернов, Е. Е. Ивашко, Н. Н. Никитина, И. Е. Габис // Компьютерные исследования и моделирование. — 2013. — Т. 5, № 1. — С. 37–45.

3. Ивашко Е. Е., Никитина Н. Н. Использование BOINC-грид в вычислительноемких научных исследованиях // Вестник НГУ. Серия: Информационные технологии. — 2013. — Т. 11. — С. 53–57.

4. Мазалов В. В., Никитина Н. Н. Оценка характеристик алгоритма Backfill при управлении потоком задач на вычислительном кластере // Вычислительные технологии. — 2012. — Т. 17, № 5. — С. 71–79.

5. Ивашко Е. Е., Никитина Н. Н. Организация квантовохимических расчетов с использованием пакета Firefly в гетерогенной грид-системе на базе BOINC // Вычислительные методы и программирование. — 2012. — Т. 13. — С. 8–12.

6. Основные результаты деятельности ЦКП КарНЦ РАН «Центр высокопроизводительной обработки данных» / В. Т. Вдовицын, А. Д. Сорокин, Е. Е. Ивашко и др. // Труды КарНЦ РАН. — 2011. — Т. 2, № 5. — С. 125–129.

7. Никитина Н. Н., Чернов И. А. Разрешимость системы разностных уравнений для динамики кумулятивной суммы // Проблемы анализа. — 2013. — Т. 2, № 20. — С. 68–81.

8. Ивашко Е. Е., Никитина Н. Н. Организация квантовохимических расчетов с использованием пакета Firefly в гетерогенной Грид на базе BOINC // Научный сервис в сети Интернет: экзафлопсное будущее. Труды Международной суперкомпьютерной конференции (Абрау-Дюрсо, 19-24 сентября 2011 г.). — 2011. — С. 178–181.

9. Ивашко Е. Е., Никитина Н. Н. Использование BOINC-грид в вычислительноемких научных исследованиях. — URL: http://conf.nsc.ru/dicr2012/ru/reportview/140625.

10. Никитина Н. Н. Оценка характеристик алгоритма Backfill при управлении потоком задач на вычислительном кластере // Обозрение прикладной и промышленной математики. — Т. 19. — 2012. — С. 461.

11. Nikitina N. Detection of a change point in Bernoulli distribution and applications in computer networks // Networking Games and Management. Extended abstracts of the International workshop. — Petrozavodsk, 2012. — P. 44.

12. Mazalov V., Nikitina N. Detection of a change point in Bernoulli distribution and applications in computer networks // Russian-Chinese Seminar on Asymptotic Methods in Probability Theory and Mathematical Statistics: Programme and Abstracts. — St. Petersburg, 2013. — P. 26.

13. Румянцев А. С., Никитина Н. Н. Задача оптимизации времени выполнения приложения в Desktop Grid // Высокопроизводительные параллельные вычисления на кластерных системах. Тезисы докладов XII Всероссийской конференции. — ННГУ, 2012. — С. 289–291.

14. Чернов И. А., Никитина Н. Н. Математические модели управления очередями заданий в BOINC. — 2013. — URL: http://boincfast.ru/papers/nikitina.pdf.

15. Mazalov V. V., Nikitina N. N., Ivashko E. E. Hierarchical Two-Level Game Model for Tasks Scheduling in a Desktop Grid // Applied Problems in Theory of Probabilities and Mathematical Statistics Related to Modeling of Information Systems. — St. Petersburg, 2014. — P. 641–645.

16. Мазалов В. В., Никитина Н. Н. Программа изменения дисциплины обслуживания при обнаружении DoS-атаки. — 2013. — Свидетельство о государственной регистрации программы для ЭВМ No.2013618298 от 05.09.2013.

17. Tsafrir D., Etsion Y., Feitelson D. G. Backfilling using system-generated predictions rather than user runtime estimates // IEEE Trans. Parallel and Distributed systems. — 2007. — Vol. 18, no. 6. — P. 789–803.

18. Nissimov A., Feitelson D. G. Probabilistic Backfilling // Job Scheduling Strategies for Parallel Processing. — 2007. — Vol. 4942. — P. 102–115.

19. Penmatsa S., Chronopoulos A. T. Cooperative Load Balancing for a Network of Heterogeneous Computers // Proceedings of the 20th International Conference on Parallel and Distributed Processing. — IPDPS'06. — Washington, DC, USA : IEEE Computer Society, 2006. — P. 162–162. — URL: http://dl.acm.org/citation.cfm?id=1898953.1899089.

20. Zhao H., Li X. Efficient Grid Task-Bundle Allocation Using Bargaining Based Self-Adaptive Auction // Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on. — 2009. — May. — P. 4–11.

21. Kolodziej J., Xhafa F., Bogdanski M. A Stackelberg Game for Modelling Asymmetric Users' Behavior in Grid Scheduling // Computer Modelling and Simulation (UKSim), 2010 12th International Conference on. — 2010. — March. — P. 497–502.

22. Penmatsa S., Chronopoulos A. T. Job Allocation Schemes in Computational Grids Based on Cost Optimization // Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International. — 2005. — April. — P. 180a–180a.

23. Khan S. A Game Theoretical Energy Efficient Resource Allocation Technique for Large Distributed Computing Systems. // PDPTA / Ed. by H. R. Arabnia. — CSREA Press, 2009. — P. 48–54. — URL: http://dblp.uni-trier.de/db/conf/pdpta/pdpta2009.html#Khan09a.

24. Buscemi M. G., Montanari U., Taneja S. A Game-Theoretic Analysis of Grid Job Scheduling // J. Grid Comput. — 2012. — Sep. — Vol. 10, no. 3. — P. 501–519. — URL: http://dx.doi.org/10.1007/s10723-012-9228-1.

25. Kwok Y.-K., Hwang K., Song S. Selfish Grids: Game-Theoretic Modeling and NAS/PSA Benchmark Evaluation // Parallel and Distributed Systems, IEEE Transactions on. — 2007. — May. — Vol. 18, no. 5. — P. 621–636.

26. Donassolo B., Legrand A., Geyer C. Non-cooperative Scheduling Considered Harmful in Collaborative Volunteer Computing Environments // Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on. — 2011. — May. — P. 144–153.

27. Петросян Л. А., Зенкевич Н. А., Шевкопляс Е. В. Теория игр. — Москва : Наука, 2012.

28. Гермейер Ю. Б. Игры с непротивоположными интересами / Под ред. Н. Н. Моисеева. — Москва : Наука, 1976.

29. Page E. S. Continuous Inspection Schemes // Biometrika. — 1954. — Vol. 41. — P. 100–114.

30. Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. — Москва : Мир, 1982. — С. 416.

31. Суперкомпьютерные технологии в науке, образовании и промышленности / Под ред. академика В. А. Садовничего, академика Г. И. Савина, чл.-корр. РАН Вл. В. Воеводина. — Москва : Издательство Московского университета, 2013. — С. 232. — ISBN: 978-5-211-05719-7.

32. High-Performance Computing on Complex Environments / Ed. by J. Zilinskas, E. Jeannot. Wiley Series on Parallel and Distributed Computing. — Wiley, 2014. — P. 512.

33. Центр высокопроизводительной обработки данных ЦКП КарНЦ РАН // Институт прикладных математических исследований Карельского научного центра РАН. — 2012. — URL: `http://cluster.krc.karelia.ru` (дата обращения: 2014.06.30).

34. A survey on resource allocation in high performance distributed computing systems / H. Hussain, S. U. Rehman Malik, A. Hameed et al. // Parallel

Computing. — 2013. — Vol. 39, no. 11. — P. 709–736. — URL: http://www.sciencedirect.com/science/article/pii/S016781911300121X.

35. Dominant Resource Fairness: Fair Allocation of Multiple Resource Types / A. Ghodsi, M. Zaharia, B. Hindman et al. // Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation. — NSDI'11. — Berkeley, CA, USA : USENIX Association, 2011. — P. 323–336. — URL: http://dl.acm.org/citation.cfm?id=1972457.1972490.

36. Stem+: Allocating bandwidth fairly to tasks : Rep. : TR–14–001 / International Computer Science Institute ; Executor: A. Lukyanenko, I. Nikolaevskiy, D. Kuptsov et al. : 2014.

37. Воеводин В. В., Жуматий С. А. Вычислительное дело и кластерные системы. — Москва : Издательство Московского университета, 2007. — С. 150.

38. Narayan A. High-performance Linux clustering, Part 1: Clustering fundamentals // IBM developerWorks. — 2005. — URL: http://www.ibm.com/developerworks/linux/library/l-cluster1.

39. Полежаев П. Н. Исследование алгоритмов планирования параллельных задач для кластерных вычислительных систем с помощью симулятора // Сборник трудов Международной научной конференции «Параллельные вычислительные технологии-2010». — Уфа, 2010. — С. 287–298.

40. Новиков А. Б. Алгоритмы планирования масштабируемых заданий кластерной вычислительной системы // Молодой ученый. — 2011. — Т. 1, № 11. — С. 74–79.

41. Iqbal S., Gupta R., Fang Y.-C. Planning considerations for job scheduling in hpc clusters // Dell Power Solutions. — February 2005. — P. 133–136. — URL: www.dell.com/downloads/global/power/ps1q05-20040135-fang.pdf.

42. Feitelson D. G., Rudolph L. Parallel job scheduling: issues and approaches // Job Scheduling Strategies for Parallel Processing. — 1995. — Vol. 949. — P. 1–18.

43. Мамойленко С. Н., Ефимов А. В. Алгоритмы планирования решения масштабируемых задач на распределенных вычислительных системах // Вестник ГОУ ВПО «СибГУТИ». — 2010. — № 2. — С. 66–78.

44. Time space sharing scheduling: A simulation analysis / A. Hori, Y. Ishikawa, J. Nolte et al. // EURO-PAR '95 Parallel Processing. — Springer Berlin / Heidelberg, 1995. — P. 623–634.

45. Jette M. A. Expanding symmetric multiprocessor capability through gang scheduling // Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing (IPPS/SPDP '98 / Ed. by D. G. Feitelson, L. Rudolph. — Springer-Verlag, London, UK, 1998. — P. 199–216.

46. Mu'alem A. W., Feitelson D. G. Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling // IEEE Transactions on Parallel and Distributed Systems. — 2001. — Vol. 12, no. 6. — P. 529–543.

47. Feitelson D. G. Experimental analysis of the root causes of performance evaluation results: a backfilling case study // Parallel and Distributed Systems, IEEE Transactions on. — 2005. — Feb. — Vol. 16, no. 2. — P. 175–182.

48. Lifka D. The ANL/IBM SP scheduling system // Job Scheduling Strategies for Parallel Processing. — 1995. — Vol. 949. — P. 295–303.

49. Wang Y. T., Morris R. J. T. Load Sharing in Distributed Systems // Computers, IEEE Transactions on. — 1985. — Vol. C-34, no. 3. — P. 204–217.

50. Casavant T. L., Kuhl J. G. A taxonomy of scheduling in general-purpose distributed computing systems // Software Engineering, IEEE Transactions on. — 1988. — Vol. 14, no. 2. — P. 141–154.

51. A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing // Softw. Pract. Exper. — 2002. — Vol. 32, no. 2. — P. 135–164. — URL: http://dx.doi.org/10.1002/spe.432.

52. Чернов И. А. Оптимальное дублирование заданий в вычислительной системе // Труды КарНЦ РАН. Серия Математическое моделирование и информационные технологии. — 2014. — № 4. — С. 130–136.

53. Anta A. F., Georgiou C., Mosteiro M. A. Algorithmic mechanisms for reliable internet-based computing under collusion // CoRR. — 2013. — Vol. abs/1307.1650. — P. 1–23. — URL: http://arxiv.org/abs/1307.1650.

54. Sarmenta L. F. G. Sabotage-Tolerance Mechanisms for Volunteer Computing Systems // Future Generation Computer Systems. — 2002. — Vol. 18. — P. 561–572.

55. Defeating colluding nodes in Desktop Grid computing platforms / G. C. Silaghi, F. Araujo, L. Moura Silva et al. // Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on. — 2008. — P. 1–8.

56. Sonnek J., Chandra A., Weissman J. B. Adaptive Reputation-Based Scheduling on Unreliable Distributed Infrastructures // Parallel and Distributed Systems, IEEE Transactions on. — 2007. — Nov. — Vol. 18, no. 11. — P. 1551–1564.

57. Di S., Wang C.-L., Hu D. H. Gossip-based Dynamic Load Balancing in a Self-organized Desktop Grid // Proceedings for the HPC Asia & APAN 2009 International Conference & Exhibition. — 2009. — P. 85–92.

58. Exploiting dynamic distributed load balance by neighbor-matching on P2P grids / P.-J. Huang, Y.-F. Yu, K.-C. Lai et al. // APSCC. — 2011. — P. 131–138. — URL: http://doi.ieeecomputersociety.org/10.1109/APSCC.2011.52.

59. Ghare G. D., Leutenegger S. T. Improving speedup and response times by replicating parallel programs on a snow // Proceedings of the 10th international conference on Job Scheduling Strategies for Parallel Processing (JSSPP'04) / Ed. by D. G. Feitelson, L. Rudolph, U. Schwiegelshohn. — Berlin, Heidelberg : Springer-Verlag, 2004. — P. 264–287.

60. A Bi-objective Scheduling Algorithm for Desktop Grids with Uncertain Resource Availabilities / L.-C. Canon, A. Essafi, G. Mounié, D. Trystram // Proceedings of the 17th International Conference on Parallel Processing - Volume Part II. — Euro-Par'11. — Berlin, Heidelberg : Springer-Verlag, 2011. — P. 238–249. — URL: http://dl.acm.org/citation.cfm?id=2033408.2033435.

61. Lee C.-Y. Parallel machines scheduling with nonsimultaneous machine available time // Discrete Applied Mathematics. — 1991. — Vol. 30, no. 1. — P. 53–61. — URL: http://www.sciencedirect.com/science/article/pii/0166218X9190013M.

62. Hwang H.-C., Lee K., Chang S. Y. The Effect of Machine Availability on the Worst-case Performance of LPT // Discrete Applied Mathematics. — 2005. — Vol. 148, no. 1. — P. 49–61. — URL: http://dx.doi.org/10.1016/j.dam.2004.12.002.

63. Румянцев А. С. Задача оптимизации времени выполнения проекта в вычислительной сети из персональных компьютеров // Программные системы: теория и приложения. — 2014. — Т. 5, № 1(19). — С. 175–182.

64. Ibarra O. H., Kim C. E. Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors // J. ACM. — 1977. — Vol. 24, no. 2. — P. 280–289. — URL: http://doi.acm.org/10.1145/322003.322011.

65. Fault-aware scheduling for bag-of-tasks applications on desktop grids / C. Anglano, J. Brevik, M. Canonico et al. // Proceedings of the 7th IEEE/ACM International Conference on Grid Computing. — 2006. — P. 56–63.

66. Kondo D., Chien A. A., Casanova H. Resource management for rapid application turnaround on enterprise desktop grids // Proceedings of the 2004 ACM/IEEE conference on Supercomputing (SC '04). — Washington, DC, USA : IEEE Computer Society, 2004. — P. 17–30.

67. ExPERT: Pareto-Efficient Task Replication on Grids and a Cloud / O. A. Ben-Yehuda, A. Schuster, A. Sharov et al. // Parallel Distributed

Processing Symposium (IPDPS), 2012 IEEE 26th International. — 2012. — May. — P. 167–178.

68. Legrand A., Touati C. Non-cooperative scheduling of multiple bag-of-task applications // INFOCOM 2007. 26th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 6-12 May 2007, Anchorage, Alaska, USA. — 2007. — P. 427–435. — URL: http://dx.doi.org/10.1109/INFCOM.2007.57.

69. Методологическое обеспечение интеллектуальных систем защиты от сетевых атак / А. А. Кондратьев, А. А. Талалаев, И. П. Тищенко и др. // Современные проблемы науки и образования. — 2014. — № 2. — URL: www.science-education.ru/116-12875 (дата обращения: 21.09.2014).

70. Лукацкий А. Обнаружение атак (2-е изд.). Мастер систем. — СПб : БХВ-Петербург, 2003. — С. 608. — ISBN: 5-94157-246-8.

71. Зима В., Молдовян А., Молдовян Н. Безопасность глобальных сетевых технологий. Мастер систем. — СПб : БХВ-Петербург, 2003. — С. 368. — ISBN: 5-94157-213-1.

72. Douligeris C., Mitrokotsa A. DDoS attacks and defense mechanisms: classification and state-of-the-art // Computer Networks. — 2004. — Vol. 44, no. 5. — P. 643–666. — URL: http://www.sciencedirect.com/science/article/pii/S1389128603004250.

73. Udhayan J., Hamsapriya T. Statistical Segregation Method to Minimize the False Detections During DDoS Attacks // International Journal of Network Security. — 2011. — Vol. 13, no. 3. — P. 152–160.

74. Kodada B. B., Prasad G., Pais A. R. Protection Against DDoS and Data Modification Attack in Computational Grid Cluster Environment // International Journal of Computer Network and Information Security (IJCNIS). — 2012. — Vol. 4, no. 7. — P. 12–18.

75. Семенов Н. А., Телков А. Ю. Применение статистических методов обнаружения DoS атак в локальной сети // Вестник ВГУ, серия: Системный анализ и информационные технологии. — 2012. — № 1. — С. 82–87.

76. Обнаружение DDoS атак нечеткой нейронной сетью / И. И. Слеповичев, П. В. Ирматов, М. С. Комарова, А. А. Бежин // Изв. Сарат. ун-та. Нов. сер. Сер. Математика. Механика. Информатика. — 2009. — Т. 9, № 3. — С. 84–89.

77. Mirkovic J., Reiher P. D-WARD: A Source-End Defense against Flooding Denial-of-Service Attacks // IEEE Transactions on Dependable and Secure Computing. — 2005. — Vol. 2, no. 3. — P. 216–232.

78. Peng T., Leckie C., Ramamohanarao K. Detecting reflector attacks by sharing beliefs // Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE. — Vol. 3. — 2003. — Dec. — P. 1358–1362.

79. Zunnurhain K. FAPA: A Model to Prevent Flooding Attacks in Clouds // Proceedings of the 50th Annual Southeast Regional Conference. — ACM-SE '12. — New York, NY, USA : ACM, 2012. — P. 395–396. — URL: http://doi.acm.org/10.1145/2184512.2184622.

80. Тарасов Я. В., Макаревич О. Б. Моделирование и исследование низкоинтенсивных DOS-атак на BGP-инфраструктуру // Известия ЮФУ. Технические науки. — 2013. — № 12. — С. 101–111.

81. Carlsson A., Duravkin E. V., Loktionova A. S. Analysis of realization and method of detecting low-intensity HTTP-attacks // Проблеми телекомунікацій. — 2013. — Т. 3, № 12. — С. 61–70.

82. Xiang Y., Li K., Zhou W. Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics // Information Forensics and Security, IEEE Transactions on. — 2011. — Vol. 6, no. 2. — P. 426–437.

83. Jadhav P. N., Patil B. M. Low-rate DDOS Attack Detection using Optimal Objective Entropy Method // International Journal of Computer Applications. — 2013. — Vol. 78, no. 3. — P. 33–38.

84. Tewari N., Bhardwaj A. Flow Statistics Based Detection of Low Rate and High Rate DDoS Attacks // International Journal of Scientific & Engineering Research. — 2013. — Vol. 4, no. 5. — P. 348–353.

85. Shuyuan J., Yeung D. S. A covariance analysis model for DDoS attack detection // Communications, 2004 IEEE International Conference on. — Vol. 4. — 2004. — P. 1882–1886.

86. Sardana A., Joshi R., Kim T. Deciding Optimal Entropic Thresholds to Calibrate the Detection Mechanism for Variable Rate DDoS Attacks in ISP Domain // Proceedings of the 2008 International Conference on Information Security and Assurance (Isa 2008). — ISA '08. — Washington, DC, USA : IEEE Computer Society, 2008. — P. 270–275. — URL: http://dx.doi.org/10.1109/ISA.2008.76.

87. Cheng C.-M., Kung H. T., Tan K.-S. Use of spectral analysis in defense against DoS attacks. // GLOBECOM. — IEEE, 2002. — P. 2143–2148. — URL: http://dblp.uni-trier.de/db/conf/globecom/globecom2002.html#ChengKT02.

88. Proactive detection of distributed denial of service attacks using MIB traffic variables-a feasibility study / J. B. D. Cabrera, L. Lewis, Xinzhou Qin et al. // Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on. — 2001. — P. 609–622.

89. Wang H., Zhang D., Shin K. G. Detecting SYN Flooding Attacks // In Proceedings of the IEEE Infocom. — IEEE, 2002. — P. 1530–1539.

90. Kim H., Rozovskii B., Tartakovsky A. A nonparametric multichart CUSUM test for rapid detection of DOS attacks in computer networks // International Journal of Computing and Information Sciences. — 2004. — Vol. 2, no. 3. — P. 149–158.

91. Ширяев А. Н. Минимаксная оптимальность метода кумулятивных сумм (CUSUM) в случае непрерывного времени // Успехи математических наук. — 1996. — Т. 51, № 4. — С. 173–174.

92. Lucas J. M., Crosier R. B. Fast Initial Response for CUSUM Quality Control Schemes: Give Your CUSUM a Head Start // Technometrics. — 1982. — Vol. 24, no. 3. — P. 199–205.

93. Gan F. F. Exact Run Length Distributions for One-Sided Exponential CUSUM Schemes // Statistica Sinica. — 1992. — Vol. 2, no. 1. — P. 297–312.

94. Vardeman S., Ray D. Average Run Lengths for CUSUM Schemes When Observations Are Exponentially Distributed // Technometrics. — 1985. — Vol. 27, no. 2. — P. 145–150.

95. Analysis of Average Run Length for CUSUM Procedure with Negative Exponential Data / J. Busaba, S. Sukparungsee, Y. Areepong, G. Mititelu // Chiang Mai Journal of Science. — 2012. — Vol. 39, no. 2. — P. 200–208.

96. Мазалов В. В., Журавлев Д. Н. О методе кумулятивных сумм в задаче обнаружения изменения трафика компьютерных сетей // Программирование. — 2002. — Т. 6. — С. 156–162.

97. Морозов Е. В., Румянцев А. С. Модели многосерверных систем для анализа вычислительного кластера // Труды Карельского научного центра РАН. — 2011. — № 5. — С. 75–85.

98. Lo V., Mache J., Windisch K. A comparative study of real workload traces and synthetic workload models for parallel job scheduling // Job Scheduling Strategies for Parallel Processing. — 1998. — Vol. 1459. — P. 25–46.

99. Benchmarks and standards for the evaluation of parallel job schedulers / S. J. Chapin, W. Cirne, D. G. Feitelson et al. // Job Scheduling Strategies for Parallel Processing. — 1999. — Vol. 1659. — P. 66–89.

100. Jackson D., Snell Q., Clement M. Core Algorithms of the Maui Scheduler // Job Scheduling Strategies for Parallel Processing / Ed. by DrorG. Feitelson, Larry Rudolph. — Springer Berlin Heidelberg, 2001. — Vol. 2221 of Lecture Notes in Computer Science. — P. 87–102. — URL: http://dx.doi.org/10.1007/3-540-45540-X_6.

101. Virtual screening strategies in drug design — methods and applications / E. Bielska, X. Lucas, A. Czerwoniec et al. // Journal of Biotechnology, Computational Biology and Bionanotechnology. — 2011. — Vol. 92, no. 3. — P. 249–264.

102. Hit and Lead Generation: Beyond High-Throughput Screening / K. Bleicher, H. Böhm, K. Müller, A. Alanine // Nature Review Drug Discovery. — 2003. — May. — Vol. 2. — P. 369–378.

103. Головко Ю. С., Ивашкевич О. А., Головко А. С. Современные методы поиска новых лекарственных средств // Вестник БГУ. Серия 2, Химия. Биология. География. — 2012. — № 1. — С. 7–15.

104. Trott O., Olson A. J. AutoDock Vina: Improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading // Journal of Computational Chemistry. — 2010. — Vol. 31, no. 2. — P. 455–461. — URL: http://dx.doi.org/10.1002/jcc.21334.

105. Anderson D. P. BOINC: A System for Public-Resource Computing and Storage // Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing. — GRID '04. — Washington, DC, USA : IEEE Computer Society, 2004. — P. 4–10. — URL: http://dx.doi.org/10.1109/GRID.2004.14.

106. Han J., Park D. Scheduling proxy: enabling adaptive-grained scheduling for global computing system // Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on. — 2004. — P. 415–420.

107. Loukas G., Öke G. Protection against denial of service attacks: A survey // The Computer Journal. — 2010. — Vol. 53, no. 7. — P. 1020–1037.

108. Jain P., Jain J., Gupta Z. Mitigation of denial of service (DoS) attack // IJCEM International Journal of Computational Engineering & Management. — 2011. — Vol. 11. — P. 38–44.

109. Hashmi M. J., Saxena M., Saini R. Classification of DDoS Attacks and Their Defense Techniques Using Intrusion Prevention System // International Journal of Computer Science & Communication Networks. — 2012. — Vol. 2, no. 5. — P. 607–614.