ITHEA

International Journal

INFORMATION CONTENT & PROCESSING

2015 Volume 2 Number 4

International Journal
INFORMATION CONTENT & PROCESSING
Volume 2 / 2015, Number 4

EDITORIAL BOARD

IJ ICP is official publisher of the scientific papers of the members of the ITHEA® International Scientific Society

IJ ICP rules for preparing the manuscripts are compulsory.
The **rules for the papers** for ITHEA International Journals as well as the **subscription fees** are given on www.ithea.org .
The papers should be submitted by ITHEA® Submission system http://ij.ithea.org .
Responsibility for papers published in IJ IMA belongs to authors.

International Journal "INFORMATION CONTENT AND PROCESSING" Volume 2, Number 4, 2015

Edited by the **Institute of Information Theories and Applications FOI ITHEA**, Bulgaria, in collaboration with
Institute of Mathematics and Informatics, BAS, Bulgaria,
V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
Universidad Politechnika de Madrid, Spain,
Hasselt University, Belgium
Institute of Informatics Problems of the RAS, Russia,
St. Petersburg Institute of Informatics, RAS, Russia
Institute for Informatics and Automation Problems, NAS of the Republic of Armenia.

Publisher: **ITHEA®**
Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, e-mail: info@foibg.com
Technical editor: Ina Markova

Printed in Bulgaria

ISSN 2367-5128 (printed) **ISSN 2367-5152 (online)**

МЕТОДЫ БАЛАНСИРОВКИ С УЧЕТОМ МУЛЬТИФРАКТАЛЬНЫХ СВОЙСТВ НАГРУЗКИ

Игорь Иванисенко, Людмила Кириченко, Тамара Радивилова

Аннотация: *В работе проведен анализ основных динамических алгоритмов балансировки нагрузки в распределенных компьютерных системах по основным показателям производительности, указаны достоинства и недостатки каждого алгоритма. Приведено математическое описание распределенной системы балансировки нагрузки, которое учитывает время и загрузку каждого сервера. Приведены значения общего уровня дисбаланса системы, а также средний уровень дисбаланса каждого сервера. Предложен динамический метод распределения нагрузки, учитывающий мультифрактальные свойства сетевого трафика, на основании которых пересчитывается распределение потоков.*

Ключевые слова: *алгоритмы балансировки нагрузки, самоподобный и мультифрактальный трафик, балансировка нагрузки, распределенные системы, пропускная способность, использование ресурсов, дисбаланс системы.*

ACM Classification Keywords: *C.2.3 Network Operations - Network management, C.2.4 Distributed Systems - Client/server, Distributed applications*

Введение

В связи с массовым распространением распределенных вычислительных систем стала актуальной проблема их эффективного использования. Одним из аспектов данной проблемы является эффективное планирование и распределение задач внутри распределенных вычислительных систем с целью оптимизации использования ресурсов и сокращения времени вычисления. Весьма часто возникает ситуация, при которой часть вычислительных ресурсов простаивает, в то время, как другая часть ресурсов перегружена и в очереди имеется большое количество ожидающих своего исполнения задач.

Для оптимизации использования ресурсов, сокращения времени обслуживания запросов, горизонтального масштабирования (динамическое добавление / удаление устройств), а также обеспечения отказоустойчивости (резервирования) применяется метод равномерного распределения заданий между несколькими сетевыми устройствами (например, серверами) называемый балансировка нагрузки, или выравнивание нагрузки (Load Balancing).

Проблема балансировки вычислительной нагрузки распределённой системы возникает по следующим причинам:

• структура вычислительного комплекса (например, кластера) неоднородна, т.е. разные вычислительные узлы обладают разной производительностью;

• структура межузлового взаимодействия неоднородна, т.к. линии связи, соединяющие узлы, могут иметь различные характеристики пропускной способности;

• структура распределенного приложения неоднородна, различные логические процессы требуют различных вычислительных мощностей.

Экспериментальные и численные исследования, проведенные в последние десятилетия, свидетельствуют, что трафик в компьютерных сетях имеет самоподобные свойства. Самоподобный трафик вызывает значительные задержки и потери пакетов, даже если суммарная интенсивность всех потоков далека от максимально допустимых значений. Самоподобные свойства информационных потоков обнаружены во многих локальных и глобальных телекоммуникационных сетях [Sheluchin, 2007; Kopparapu, 2008; Шелухин, 2011; Erl, 2013]. В связи с вышеизложенным начали активно исследоваться механизмы повышения качества обслуживания и методов управления трафиком в компьютерных сетях, функционирующих в условиях самоподобного и мультифрактального трафика [Кириченко, 2011; Ivanisenko, 2015].

При появлении новых заданий программное обеспечение, реализующее балансировку, должно принять решение о том, на каком вычислительном узле следует выполнять вычисления, связанные с этим новым заданием. Кроме того, балансировка предполагает перенос части вычислений с наиболее загруженных вычислительных узлов на менее загруженные узлы. При выполнении задач процессоры обмениваются между собой коммуникационными сообщениями. В случае низких затрат на коммуникацию, некоторые процессоры могут простаивать, в то время как остальные будут перегружены. Также будут нецелесообразны большие затраты на коммуникацию. Следовательно, стратегия балансировки должна быть такой, чтобы вычислительные узлы были загружены достаточно равномерно, но и коммуникационная среда не должна быть перегружена [Cardellini, 1999; Гуревич, 2010; Ghuge, 2014; Red Hat, 2015].

Большое внимание в научной литературе уделяется вопросам управления нагрузкой в распределенных системах. Теоретические исследования и разработка фундаментальных основ распределения нагрузки, создание математического аппарата, моделей и методов управления для распределения нагрузки в распределенных системах рассматривались в работах ученых [Kameda, 1997; Cardellini, 2001; Casalicchio, 2001; Hong, 2006; Randles, 2010; Mehta, 2011; Sran, 2013]. Разрабатывали и усовершенствовали алгоритмы балансировки нагрузки зависимые от

времени завершения задачи на машине такие ученые как [Keshav, 1997; Elzeki, 2012], алгоритмы грубой силы и основанные на статистических данных рассматривались в работах таких ученых как [Hu, 1998; Ghanbari, 2012; Chen, 2013; Liu, 2013], алгоритмы, основанные на биологических феноменах рассматривались [Mishra,2012; Dhinesh, 2013], а также многие другие исследователи, работающие над проблемами распределения нагрузки.

Целью данной работы является математическое описание распределенной системы балансировки нагрузки и разработка динамического метода распределения нагрузки в компьютерных системах на основе мониторинга загруженности серверов, с учетом самоподобной структуры трафика и результатов анализа основных алгоритмов балансировки нагрузки распределенных систем по различным показателям производительности.

1. Классификация алгоритмов балансировки нагрузки

Алгоритмы балансировки нагрузки могут быть классифицированы по нескольким типам [Gupta, 2013; Kashyap, 2014].

- **На основании текущего состояния системы**

  Статический алгоритм.

  В этом случае текущее состояние узла не учитывается, требуется предварительная база знаний о статистике каждого узла и пользовательских требованиях, не гибкий, не масштабируемый, не совместим с меняющимися требованиями пользователей и загрузки. Используется в однородной среде.

  Динамический алгоритм.

  Этот тип алгоритма работает в соответствии с динамическими изменениями состояния узлов, т.е. он собирает, хранит и анализирует информацию о состоянии системы. Из-за этого возникают большие накладные расходы на балансировку. Необходимо учитывать расположение процессора, которому передается нагрузка от перегруженного процессора, оценку нагрузки, ограничение числа миграций. Если какой-либо узел дал сбой, то это не остановит работу всей сети, но повлияет на производительность системы. Используется в гетерогенной среде.

- **На основании инициатора алгоритма**

  Инициированый отправителем: отправитель определяет, что количество узлов большое, и отправитель инициирует выполнение алгоритма балансировки нагрузки.

  Инициированный получателем: требование о балансировке нагрузки могут быть определены получателем / сервером в облаке, и тогда сервер инициирует выполнение алгоритма балансировки нагрузки.

  Симметричный: Это сочетание типов инициированных отправителем и получателем.

Динамический подход балансировки нагрузки подразделяется на два типа: распределенный и нераспределенный (централизованный) подходы. Они определяется следующим образом:

- **в нераспределенном подходе** один узел или группа узлов отвечают за управление и распределение по всей системе. Другие узлы не распределяют задачи и не выполняют управляющие функции, следовательно этот тип алгоритмов не отказоустойчив, может произойти перегрузка центрального процессора принятия решений. Полезны в небольших сетях с низкой нагрузкой.
- **в распределенном подходе** каждый узел независимо строит свой вектор нагрузки. Все процессоры в сети ответственны за распределение нагрузки и наполнение собственной локальной базы данных для принятия эффективных решений балансировки. Это приводит к большим коммуникационным затратам и сложности алгоритмов. Полезны в больших и гетерогенных сетях.

При распределенном подходе балансировка нагрузки может иметь две формы: кооперативную и некооперативную. В кооперативной узлы работают бок-о-бок для достижения общей цели, например, улучшение общего времени отклика. При некооперативном подходе балансировки узел работает независимо от цели, например, улучшение времени выполнения местной задачи.

Узлы постоянно взаимодействуют друг с другом и при распределенном подходе генерируют больше сообщений, чем при нераспределенном. Передача сообщений между узлами для обмена информацией об обновлениях системы может привести к снижению производительности системы [Kameda, 1997]. Один узел или группа узлов решает задачу балансировки нагрузки при нераспределенном подходе, он может принимать две формы централизованной и полу распределенной.

В централизованных алгоритмах один узел несет исключительную ответственность за балансировку нагрузки всей системы и называется центральным узлом. Используется в небольших сетях. В полу распределенных алгоритмах кластер формируется группой узлов системы, балансировка нагрузки происходит в каждом кластере по централизованному типу. Среди узлов в кластере центральный узел инициализирует балансировку нагрузки в этой группе. Полураспределенные алгоритмы обмениваются большим количеством сообщений по сравнению с централизованными.

Обычно динамический алгоритм балансировки содержит четыре элемента: политику балансировки (transfer policy); алгоритм выбора партнера (location policy); алгоритм выбора задачи для передачи (selection policy); механизм сбора необходимой информации о состоянии системы (information policy).

Политика балансировки определяет, является ли узел объектом балансировки [Kaur, 2014]. Различные виды политик балансировки используют пороговые значение загруженности узлов, отклонение величины нагрузки узла от среднего значения по системе и другие методы. Алгоритм выбора задачи определяет, какую задачу необходимо передать. При выборе задачи для отправки алгоритм может учитывать, что накладные расходы, связанные с пересылкой задачи, должны быть минимальны, сложность выполнения задачи должна быть большой, число связей в задаче с локальными ресурсами должно быть минимально. Алгоритм выбора партнера отвечает за выбор подходящего узла для операции балансировки. Распространенной техникой в распределенных алгоритмах является опрос (polling) узлов. Опрос может быть последовательным или параллельным, использовать результаты предыдущих опросов. В централизованных алгоритмах узел обращается к специализированному координатору для определения подходящего партнера для балансировки. Координатор собирает и поддерживает в актуальном состоянии информацию о загруженности узлов системы, а алгоритм поиска партнера использует эти данные.

Механизм сбора информации о загруженности системы определяет источник информации, время сбора данных о загруженности, место хранения информации. Существует несколько классов механизмов сбора информации.

1. Сбор данных по необходимости. В данном классе используются распределенные алгоритмы, которые собирают информацию о загруженности, когда узел нуждается в балансировке нагрузки.

2. Периодический сбор данных. Алгоритмы данного класса могут быть как централизованными, так и распределенными. В зависимости от собранных данных, алгоритм инициирует балансировку нагрузки.

3. Сбор данных по изменению состояния. В системах, реализующих алгоритмы данного класса, узлы сами распространяют информацию об изменении загруженности при изменении внутреннего состояния. В случае распределенных алгоритмов данные направляются соседним узлам, в случае централизованных алгоритмов данные направляются координатору.

Классификация алгоритмов балансировки нагрузки схематично представлена на рис. 1.

Балансировщики нагрузки используют различные алгоритмы для управления трафиком с целью распределения нагрузки и / или максимального использования всех серверов в кластере.

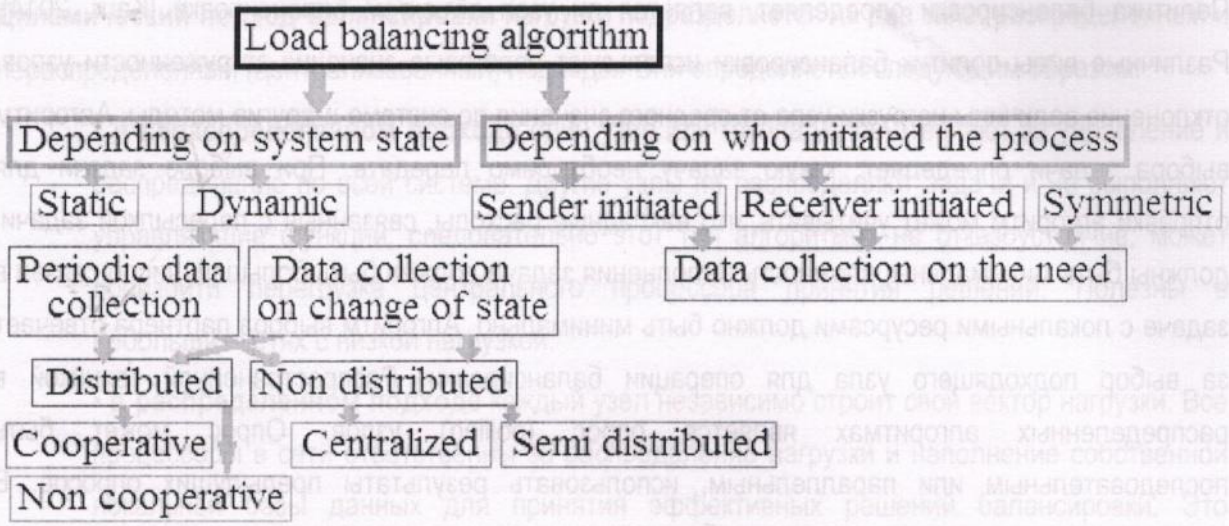

Рисунок 1. Классификация алгоритмов балансировки нагрузки

2. Анализ алгоритмов балансировки нагрузки

Благодаря алгоритмам балансировки достигается более высокая пропускная способность и улучшается время отклика в распределенным системах. Однако каждый из алгоритмов имеет как преимущества, так и недостатки [Gupta, 2013; Kashyap, 2014]. Рассмотрим основные алгоритмы балансировки нагрузки.

1. В Round Robin Scheduling [Keshav, 1997; Ghuge, 2014; Rajwinder, 2014] (управление задачами в системах с распределением времени) алгоритм определяет кольцо как очередь и квант фиксированного времени. Каждое задание может быть выполнено только в этот квант времени и в свою очередь. Если задача не может быть завершена в течение одного кванта, она вернется в очередь на ожидание следующего круга. Однако существует проблема определения подходящего кванта времени. Когда квант времени очень большой, то RR алгоритм планирования работает так же, как FCFS Scheduling. А когда квант времени слишком мал, то Round Robin Scheduling известен как Processor Sharing алгоритм.

2. Max-Min Algorithm [Elzeki, 2012; Katyal, 2013; Ghuge, 2014; Kashyap, 2014; Rajwinder, 2014]: в этом алгоритме выяснив сначала, минимальное время выполнения задач, выбирается максимальное значение, которое является максимальным временем среди всех задач на всех ресурсах. Далее задача с найденным максимальным временем назначается на выполнение на конкретно выбранный узел. Затем перерасчитывается время выполнения всех заданий на этом узле путем добавления времени выполнения поставленной задачи ко времени выполнения других задач на этом узле. Затем поставленная задача удаляется из списка системы.

3. Алгоритм Compare and Balance [Hu, 1998; Gupta, 2013; Rajwinder, 2014] используется для достижения равновесного состояния и управления несбалансированной нагрузкой системы. В

этом алгоритме на основе вероятности (номер виртуальной машины, запущенной на текущем хосте и всей облачной системы), текущий хост случайным образом выбирает хост и сравнивает их нагрузку. Если нагрузка текущего хоста больше выбранного хоста, он передает дополнительную нагрузку на этот конкретный узел. Затем каждый узел системы выполняет ту же процедуру. Этот алгоритм балансировки нагрузки также разработан и реализован для уменьшения времени миграции виртуальных машин. Общая память используется для уменьшения времени миграции виртуальных машин.

4. Ant Colony Optimization [Mishra, 2012; Dhinesh, 2013; Katyal, 2013; Rajwinder, 2014]: это распределенные алгоритм. В этом алгоритме, информация о ресурсах динамически обновляется при каждом движении муравьев. Множественные колонии муравьев описываются таким образом, что узел посылает цветные колонии по всей сети. Раскрашенные колонии муравьев используются для предотвращения движения муравьев из одного гнезда следующих одним маршрутом, а также обеспечения их распределения по всем узлам в системе, где каждый муравей действует как мобильный агент, который несет обновленную информацию балансировки нагрузки в следующий узел.

5. В алгоритме Shortest Response Time First [Singhal, 2011; Liu, 2013; Kashyap, 2014] главной идеей является прямая переадресация. В нем каждому процессу назначается приоритет для его запуска. В процессы с равными приоритетами планируется в FCFS порядке. SJF алгоритм является частным случаем общего алгоритма приоритетного планирования. В алгоритме SJF приоритет является обратным по отношению к следующему всплеску процессора (CPU). Это означает, что если всплеск процессора увеличивается, то приоритет понижается. Политика SJF выбирает задачу с кратчайшим временем обработки. В этом алгоритме короткие задачи выполняются перед длинными задачами. В SJF очень важно знать или оценить время обработки каждого задания, что является главной проблемой SJF.

6. Active Clustering load balancing Algorithm [Hu, 1998; Singhal, 2011; Katyal, 2013]. Алгоритм самоагрегации оптимизирует задачи, подключив похожие услуги с использованием локального переприсваивания. Работает на основе группирования похожих узлов. Процесс группирования основан на концепции рефери узла. Рефери узел образует соединение между соседями, которое подобно инициализирующему узлу. Затем рефери узел разрывает соединение между собой и начальным узлом. Далее совокупность процессов снова и снова повторяется. Производительность системы возрастает на основе высокой доступности ресурсов, из-за этого пропускная способность также увеличивается.

7. ACCLB [Singhal, 2011]: методика балансировки загрузки, основанная на муравьиной колонии и теории сложной сети (ACCLB) в открытых облачных вычислениях. Она использует низкоуровневые и безмасштабные характеристики сложной сети, чтобы добиться лучшего

распределения нагрузки. Эта методика позволяет преодолеть неоднородность, является адаптивной к динамичной среде, превосходна по устойчивости к ошибкам и имеет хорошую масштабируемость, следовательно, помогает в улучшении производительности системы.

8. Join-Idle-Queue [Singhal, 2011, Gupta, 2013]: алгоритм балансировки нагрузки для динамически масштабируемых веб-сервисов, обеспечивает масштабную балансировку нагрузки по распределенным отправителям. Сначала вычисляет доступность простаивающих процессоров в каждом отправителе, а затем назначает задания процессорам для уменьшения средней длины очереди в каждом процессоре. При удалении задач балансировки нагрузки из критического пути обработки запроса он эффективно снижает нагрузку системы, не несет никакой коммуникационной нагрузки на вновь прибывшие задачи и не увеличивает фактическое время отклика.

9. Central queuing [Rajwinder, 2014]. Этот алгоритм работает по принципу динамического распределения. Каждая новая задача прибывает к менеджеру очередей и становится в очередь. Когда запрос для выполнения задачи принимается менеджером очередей, он удаляет первую задачу из очереди и передает ее запрашивающей стороне. Если в очереди нет готовых задач, то запрос буферизируется, пока новая задача не будет доступна. Но в случае записи новой задачи в очередь, пока есть неотвеченные запросы в очереди, первый такой запрос удаляется из очереди, а новая задача ставиться перед ней. Когда загрузка процессора падает ниже порогового значения, то локальный менеджер загрузки посылает запрос на новую задачу центральному менеджеру загрузки. Затем центральный менеджер отвечает на запрос, если найдена готовая задача, в противном случае соблюдается очередность запросов до поступления новой задачи.

10. Connection mechanism [Ray, 2012; Gupta, 2013; Liu, 2013]: алгоритм балансировки нагрузки основан на механизме наименьшего количества соединений, который является частью динамического алгоритма планирования. Он необходим для подсчета количества соединений для каждого сервера и динамической оценки нагрузки. Балансировщик нагрузки записывает количество соединений каждого сервера. Количество соединений увеличивается, когда новое соединение отправляется к серверу, и уменьшается, когда соединение завершается или происходит прерывание соединения.

11. Least connections [Mishra, 2012; Kashyap, 2014]. Алгоритм минимума соединений посылает запросы на сервер, который в настоящее время обслуживает наименьшее количество подключений. Балансировщик нагрузки будет отслеживать количество соединений серверов и отправит следующий запрос к серверу с минимумом соединений.

Эффективность алгоритмов балансировки нагрузки определяется несколькими показателями, которые представлены ниже [Ghuge, 2014; Raghava, 2014].

- *Пропускная способность* используется для оценки общего количества задач, которые успешно завершены. Высокая пропускная способность необходима для общей производительности системы.
- *Время миграции* определяется как общее время перехода задачи от одного узла или ресурса к другому. Оно должно быть сведено к минимуму.
- *Время отклика* измеряется как интервал времени между отправкой запроса и получением ответа. Оно должно быть сведено к минимуму, чтобы повысить общую производительность.
- *Отказоустойчивость* измеряет способность алгоритма равномерно выполнять балансировку нагрузки в случае любого сбоя. Хороший алгоритм балансировки нагрузки должен быть нечувствительным к неисправностям.
- *Затраты* связаны с работой любого алгоритма балансировки нагрузки и указывают на стоимость процессов, участвующих в решении задачи, перераспределении процессов. Они должны быть как можно меньше.
- *Использование ресурсов* используется для обеспечения надлежащего использования всех ресурсов, которые включены в систему. Данный показатель должен быть оптимизирован для эффективности алгоритма балансировки нагрузки.
- *Масштабируемость* – способность алгоритма выполнять равномерную балансировку нагрузки в системе в соответствии с требованиями при увеличении числа узлов. Предпочтительным является алгоритм с высокой масштабируемостью.
- *Производительность* может быть определена как эффективность системы. Данный показатель должен быть улучшен при разумных затратах, например, уменьшая время отклика сохраняя допустимые задержки.

В табл. 1 приведено сравнение рассмотренных алгоритмов балансировки нагрузки по возможности их оценивания по различным показателям производительности.

Таблица 1. Сравнительный анализ алгоритмов по показателям производительности

Алгоритм / Метрика	Пропускная способность	Время миграции	Время отклика	Отказоустойчивость	Затраты	Использование ресурсов	Масштабируемость	Производительность
Round Robin Sheduling	да	нет	да	нет	да	да	нет	да
Max-Min	да	нет	да	нет	да	да	нет	да
Compare and Balance	да	да	да	да	да	нет	да	да
Ant Colony Optimization	да	нет	нет	нет	да	да	да	да

Shortest Response Time First	нет	да	да	нет	да	да	нет	да
Active Clustering	нет	да	нет	нет	да	да	нет	нет
ACCLB	да	нет	нет	нет	да	да	да	да
Join-Idle-Queue	да	да	да	нет	да	нет	да	да
Central queuing	нет	да	да	нет	да	да	нет	да
Connection mechanism	да	нет	нет	да	да	да	да	нет
Least connections	нет	нет	нет	да	да	нет	нет	нет

3. Самоподобный и мультифрактальный трафик

Самоподобие случайных процессов заключается в сохранении статистических характеристик при изменении масштаба времени и характеризуется показателем Херста H, который является степенью самоподобия. Стохастический процесс $X(t)$ является статистически самоподобным если процесс $a^{-H}X(at)$ обладает теми же статистическими характеристиками второго порядка, что и $X(t)$. Параметр H, называемый параметром Херста, представляет собой меру самоподобия стохастического процесса. Моменты самоподобного случайного процесса можно выразить как $\mathrm{M}\left[|X(t)|^q\right] = C(q)\cdot t^{qH}$, где величина $C(q)$ – некоторая детерминированная функция. Для мультифрактальных процессов выполняется отношение $\mathrm{M}\left[|X(t)|^q\right] = c(q)\cdot t^{qh(q)}$, где $c(q)$ –детерминированная функция; $h(q)$ – обобщенный показатель Херста, являющийся в общем случае нелинейной функций. Значение $h(q)$ при $q=2$ совпадает со значением степени самоподобия H. [Kantelhardt, 2008].

Самоподобный трафик имеет особую структуру, которая сохраняется на многих масштабах – в реализации всегда присутствует некоторое количество очень больших выбросов при относительно небольшом среднем уровне трафика. Мультифрактальный трафик определяется как расширение самоподобного трафика за счет учета масштабируемых свойств статистических характеристик второго и выше порядков.

Как характеристику неоднородности мультифрактального потока данных в работе предложено считать диапазон обобщенного показателя Херста $\Delta h = h(q_{\min}) - h(q_{\max})$. Для монофрактальных процессов обобщенный показатель Херста не зависит от параметра q и является

прямой линией: $h(q) = H$, $\Delta h = 0$. Чем больше неоднорость процесса, т.е. большее число выбросов присутствует в трафике, тем больше диапазон Δh.

4. Описание системы балансировки нагрузки

Рассматриваемая система балансировки нагрузки состоит из группы серверов и балансировщика нагрузки [Hui, 2008; Ignatenko, 2010; Zaborowski, 2013]. Представленная на рис. 2 система балансировки нагрузки построена на основе подсистемы балансировки нагрузки и подсистемы управления и мониторинга, которые тесно взаимодействуют друг с другом:

• Подсистема балансировки нагрузки: алгоритм балансировки нагрузки, информация о текущем состоянии системы, гибкие настройки QoS, динамическое распределение трафика по различным каналам связи и узлам в зависимости от их текущего состояния, степени загрузки, административных политик балансировки нагрузки.

• Подсистема управления и мониторинга: сбор и анализ статистики о текущем состоянии системы, определение мультифрактальных свойств входящего потока данных, расчет распределения потоков по узлам сети с учетом классификации трафика и загруженности серверов и каналов связи.

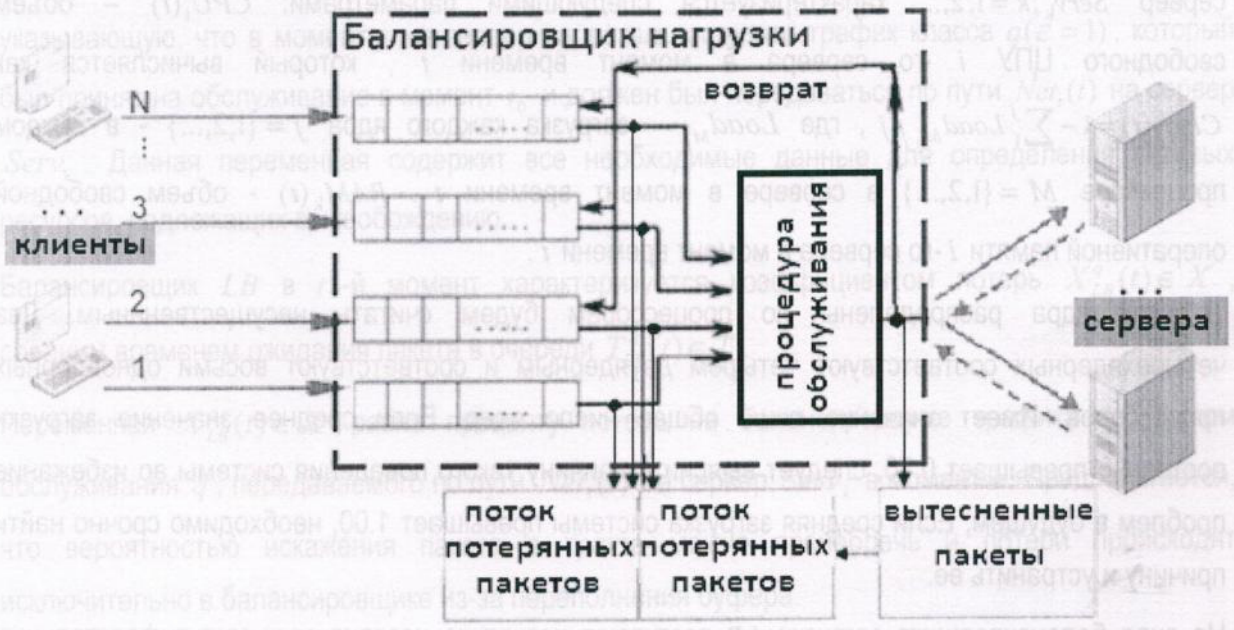

Рисунок 2. Система балансировки нагрузки

В каждый момент $t \in T$ на балансировщик LB поступает трафик интенсивностью $\lambda = [\lambda_1, \lambda_2, .., \lambda_N]$, относящийся к q -му классу обслуживания, который необходимо доставить на сервер $Serv_k$ для обработки, не превышая заданных максимально допустимых значений задержки τ_q и максимально допустимого процента потерь l_q в зависимости от их текущей загрузки и реальной пропускной способности в конкретный момент времени.

Трафик обладает множеством характеристик $V = \{\lambda, h, \mu\}$, где $\lambda = [\lambda_1, \lambda_2, .., \lambda_N]$ – потоки заявок (пакетов) различной интенсивности; $h = [H, h(q), \Delta h]$, где $h(q)$ – выборочное значение функции обобщенного показателя Херста, $H = h(2)$ – значение параметра Херста, $\Delta h = h(q_{\min}) - h(q_{\max})$ – диапазон значений обобщенного показателя Херста для участка трафика, μ – трудоемкость запроса. Трудоемкость запроса определяется как вектор требуемых ресурсов $\mu = (CPU, Net_i, RAM)$ для выполнения запроса. Каждому q -му классу обслуживания соответствует набор векторов требуемых ресурсов $\mu_r = (CPU, Net_i, RAM), r = 1, 2, ...$

Балансировщик нагрузки и сервера соединены между собой двусторонними сетевыми связями [Kameda, 1997; Tian, 2014] с максимальной пропускной способностью $Link_i = \{L_i\}$, $i = 1, 2, .., N$, которые имеют доступную пропускную способность $Net_i(t) = \{Net_i\}$ в момент времени t. Каждый сервер $Serv_k, k = 1, 2, ...$ характеризуется следующими параметрами: $CPU_i(t)$ – объем свободного ЦПУ i -го сервера в момент времени t, который вычисляется как $CPU_i(t) = 1 - \sum_1^j Load_{M_j} \Big/ j$, где $Load_{M_j}$ – загрузка каждого ядра $j = \{1, 2, ...\}$ - в каждом процессоре $M = \{1, 2, ...\}$ в сервере в момент времени t. $RAM_k(t)$ - объем свободной оперативной памяти i -го сервера в момент времени t.

То, как ядра распределены по процессорам будем считать несущественнным. Два четырехядерных соответствуют четырем двуядерным и соответствуют восьми одноядерных процессоров. Имеет значение лишь общее число ядер. Если среднее значение загрузки постоянно превышает 0.70, следует выяснить причину такого поведения системы во избежание проблем в будущем; Если средняя загрузка системы превышает 1.00, необходимо срочно найти причину и устранить ее.

На вход балансировщика загрузки LB поступают несколько независимых мультифрактальных потоков пакетов с различной интенсивностью $\lambda_1, \lambda_2, .., \lambda_N$ каждый из которых отправляется в очередь Q_i ограниченной емкости. Время обслуживания заявок зависит от класса обслуживания q, то есть учитывается приоритетность заявок (наивысший приоритет – первым). Пока все приоритетные запросы на обслуживание не будут обработаны, пакеты других типов остаются в очереди до истечения их времени жизни. Вновь поступившие приоритетные запросы прерывают

обработку неприоритетных и с вероятностью равной единице вытесняют их в накопитель (если есть свободные места ожидания), либо за пределы системы (если свободных мест нет). Вытесненные с обслуживания пакеты присоединяются к очереди неприоритетных требований и могут быть дообслужены после всех приоритетных. Накопители являются раздельными для каждого входного потока, свободные места ожидания полнодоступны для любого вновь поступившего запроса.

В отличие от типовых приоритетных СМО рассматриваемая система снабжена вероятностным выталкивающим механизмом. Приоритетный пакет, заставший все места ожидания занятыми в момент обработки другого приоритетного пакета, с заданной вероятностью вытесняет из накопителя один из менее приоритетных пакетов и занимает его место. Вытесненный пакет теряется либо отправляется обратно в очередь. Подсистема балансировки загрузки LB, в соответствии с заложенным в нее алгоритмом осуществляет извлечение задач из очередей Q_i и назначение их на свободные вычислительные ядра подходящих серверов.

Для описания механизма высвобождения занятых трафиком сетевых ресурсов при окончании передачи трафика q -го класса обслуживания, (это происходит на основе данных, поступающих от протокола маршрутизации, поддерживающего сообщения о доступной полосе пропускания и доступных ресурсах на сервере, например, CSPF, SNMP), введем переменную $LB\ \varepsilon_{C_{L_i}}^{q,t_0}(t)=\{0,1\}$ указывающую, что в момент t на сервер перестал поступать трафик класса $q(\varepsilon=1)$, который был принят на обслуживание в момент t_0 и должен был передаваться по пути $Net_i(t)$ на сервер $Serv_k$. Данная переменная содержит все необходимые данные для определения сетевых ресурсов, подлежащих высвобождению.

Балансировщик LB в t -й момент характеризуется коэффициентом потерь $X_{LB}^q(t)\in X$, средним временем ожидания пакета в очереди $T_{LB}^q(t)\in T$.

Переменная $X_{LB}^q(t)\in X$ равна проценту потерь на балансировщике трафика с классом обслуживания q, передаваемого по пути $Net_i(t)$ на сервер $Serv_k$ в момент t. Предполагается, что вероятностью искажения пакета в тракте можно пренебречь и потери происходят исключительно в балансировщике из-за переполнения буфера.

Мониторинг состояния серверов и свободной пропускной способности [Игнатенко, 2011] можно осуществить тремя способами:

- после каждого поступившего запроса;
- в фиксированные промежутки времени, определяемые статическим алгоритмом;
- в нефиксированные промежутки времени, определяемые динамическим алгоритмом.

Информация, полученная первым способом, является наибольшей по объему, т.к. измерения проводятся после каждого поступившего запроса. При втором способе количество информации постоянно, но необходимо определить интервал съема информации, чтобы объем информации не был избыточным и недостаточным. При третьем способе количество информации зависит от частоты интервалов контроля, который должен приспосабливаться к структуре поступающего трафика, учитывая его самоподобную структуру.

5. Описание общего уровня дисбаланса системы и каждого сервера

Необходимо ввести интегрированные значения общего уровня дисбаланса системы, а также средний уровень дисбаланса каждого сервера [Tian, 2014]. Одна из интегрированных метрик баланса нагрузки описывается следующим образом:

$$V = \frac{1}{(1-CPU_i)(1-RAM_k)(1-Net_i)}, \quad (1)$$

где CPU_i, RAM_k, Net_i - средняя загрузка процессора, памяти и пропускной способности сети, соответственно, в течение каждого наблюдаемого периода. Большое значение V означает высокую степень комплексного использования. Поэтому алгоритмы миграции могут быть основаны на данном измерении. Это на самом деле является стратегией минимизации комплексного использования ресурсов путем преобразования трехмерной информации (3D) ресурсов в одномерную величину (1D). Это преобразование может привести к многомерной потери информации. В [Zheng, 2006] была предложена другая интегрированная метрика балансировки нагрузки:

$$B = \frac{aN1_i C_i}{N1_m C_m} + \frac{bN2_i M_i}{N2_m M_m} + \frac{cNet_i}{Net_m}. \quad (2)$$

Первым выбирается физический сервер m. Затем, другие физические сервера i сравниваются с сервером m. $N1_i$ это объем ЦПУ, $N2_i$ это объем памяти. Здесь C_i и M_i обозначают средние значения использования процессора и памяти, соответственно. Net_i представляет собой сетевую пропускную способность. Параметры a, b, c обозначают весовые коэффициенты для процессора, памяти и пропускной способности сети, соответственно. Основная идея этого алгоритма заключается в выборе наименьшего значения B среди всех физических серверов. Этот метод также преобразует 3D информацию ресурсов в значение 1D.

Учитывая преимущества и недостатки существующих метрик для планирования ресурсов, была разработана стратегия балансировки нагрузки, которая включает в себя комплексное измерение общего уровня дисбаланса системы, а также среднего уровня дисбаланса каждого сервера.

Также могут быть разработаны другие показатели для различных стратегий планирования. Рассмотрены следующие параметры:

1. Средняя загрузка ЦПУ CPU_i^u i-го сервера определяется как средняя загрузка процессора в течение наблюдаемого периода. Например, если период наблюдений составляет 1 минута, а загрузка процессора записывается через каждые 10 секунд, то есть CPU_i^u – это среднее значение из шести записанных значений i-го сервера.

2. Средний коэффициент использования всех процессоров в системе. Пусть CPU_i^u – общее число ЦПУ i-го сервера,

$$CPU_u^A = \frac{\sum_i^N CPU_i^u CPU_i^n}{\sum_i^N CPU_i^n}, \quad (3)$$

где N – общее число физических серверов в системе. Аналогичным образом, средний коэффициент использования памяти, пропускной способности сети i-го сервера, вся память, и вся пропускная способность сети в системе может быть определена как RAM_i^u, Net_i^u, RAM_u^A и Net_u^A соответственно.

3. Комплексное значение дисбаланса нагрузки SIL_i i-го сервера. Используя дисперсию, интегрированное значение дисбаланса нагрузки i-го сервера определяется (4)

как: $SIL_i = a(D_i - CPU_u^A)^2 + b(D_i - RAM_u^A)^2 + c(D_i - Net_u^A)^2$,

где

$$D_i = aCPU_i^u + bRAM_i^u + cNet_i^u. \quad (5)$$

Значение дисбаланса SIL_i применяется для обозначения уровня дисбаланса нагрузки путем сравнения коэффициентов использования процессора, памяти и пропускной способности сети.

4. Значение дисбаланса всех процессоров, памяти и пропускной способности сети. Используя дисперсию, значение дисбаланса всех процессоров в центре обработки данных определяется как

$$ISL_{cpu} = \sum_i^N (CPU_i^u - CPU_u^A)^2. \quad (6)$$

Точно так же могут быть рассчитаны значения дисбаланса памяти и пропускной способности сети. В этом случае суммарные значения дисбаланса всех серверов в системе записывается как:

$$ISL_{tot} = \sum_i^N ISL_i. \quad (7)$$

5. Среднее значение дисбаланса физического i-го сервера определяется как:

$$ISL_D^{PM} = \frac{ISL_{tot}}{N}, \quad (8)$$

где N – общее количество серверов. Как следует из названия, это значение используется для измерения уровня дисбаланса всех физических серверов.

6. Среднее значение дисбаланса системы определяется как:

$$ISL_D^{sys} = \frac{ISL_{cpu} + ISL_{RAM} + ISL_{Net}}{N}. \quad (9)$$

7. Средняя продолжительность работы при одинаковом количестве задач позволяет сравнивать различные алгоритмы планирования.

8. Период обработки определяется как максимальная нагрузка (или средняя загрузка) на любом сервере.

9. Эффективность использования определяется как минимальная нагрузка на любом сервере разделенная на максимальную нагрузку на любом сервере.

5. Балансировка нагрузки с учетом мультифрактальных свойств трафика

Очереди и потери, порождаемые трафиком с мультифрактальными свойствами, зависят от мультифрактальной характеристики: функции обобщенного показателя Херста. Диапазон значений обобщенного показателя Херста соответствует степени неоднородности трафика, т.е. характеризует разброс данных. Значение обычного параметра Херста, полученное из обобщенного показателя, соответствует степени долгосрочной зависимости реализации и характеризует корреляционные свойства трафика.

Использование динамически изменяющегося алгоритма является наиболее приемлемым с точки зрения уменьшения избыточности данных. При таком способе частота мониторинга будет зависеть от значений обобщенного показателя Херста $h(q)$ входящего потока. Интервал мониторинга должен учащаться, если во входящем потоке диапазон значении обобщенного показателя Херста принимает значения, большие заданной величины $\Delta h = h(q_{\min}) - h(q_{\max}) > \Delta_{MAX}$, и при значении показателя Херста $H = h(q=2) > H_{MAX} > 0.8$.

5.1. Описание компонентов балансировки нагрузки

Система балансировки нагрузки состоит из компонентов, представленных на рисунке 3. Балансировщик нагрузки имеет веб-интерфейс для отслеживания, настройки и администрирования распределения нагрузки [Игнатенко, 2011; Zhihao, 2013; Tian, 2014; Red Hat, 2015].

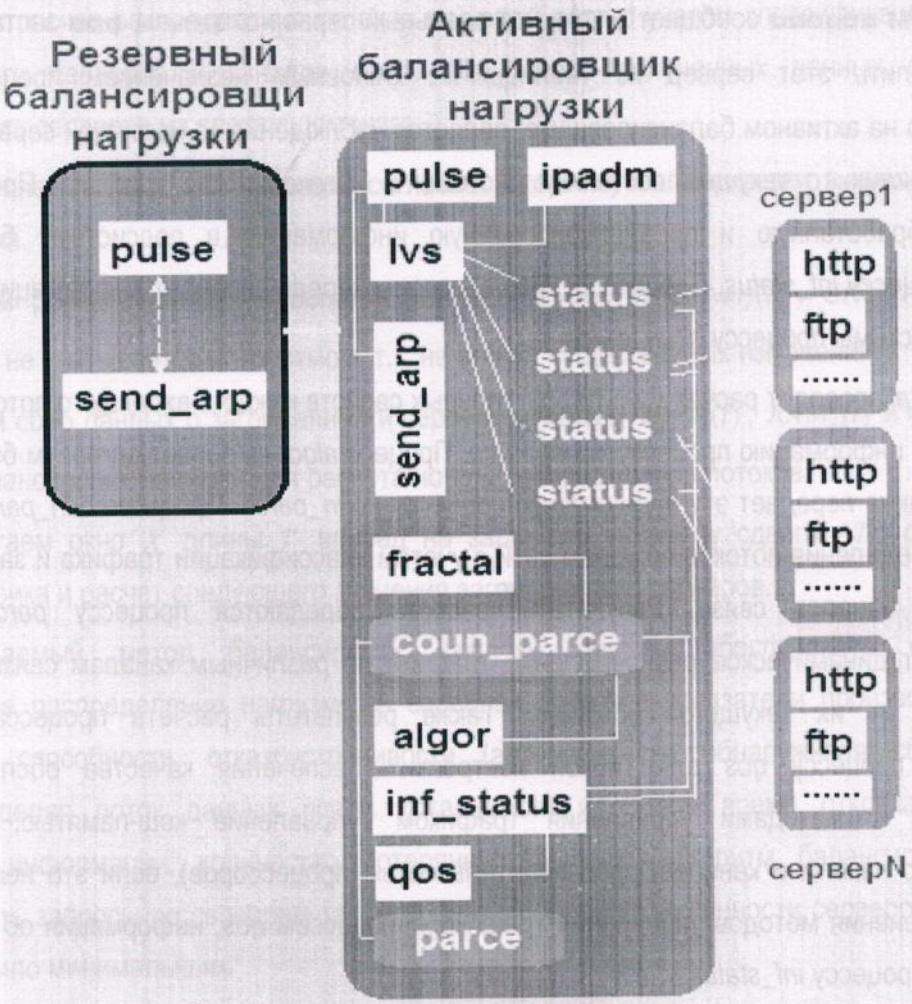

Рисунок 3. Компоненты распределения нагрузки

Процесс ▪▪▪▪▪ – это управляющий процесс, который на активном балансировщике запускает процесс *lvs*, а на резервном балансировщике будет отслеживать состояние активного, периодически опрашивая его. Если активный балансировщик не отвечает в течение заданного периода времени, будет инициирован процесс передачи его функций резервному балансировщику. При этом ▪▪▪▪▪ на резервном балансировщике отправляет процессу ▪▪▪▪▪ на активном балансировщике команду остановки всех служб *lvs*, запускает ▪▪▪▪▪▪▪▪ для присвоения виртуальных IP-адресов MAC-адресу резервного балансировщика и запускает процесс ▪▪▪. Процесс ▪▪▪▪▪▪▪▪ рассылает широковещательные пакеты ARP при переходе виртуального IP-адреса от одного узла другому. [Erl, 2015]

Процесс ▪▪▪ запускается на активном балансировщике по вызову ▪▪▪▪▪. Он вызывает службу ▪▪▪▪▪ для создания, добавления, изменения и удаления записей в таблице маршрутизации IP. Процесс ▪▪▪ запускает процесс ▪▪▪▪▪▪ для каждой настроенной службы распределения

нагрузки. Если ▪▪▪▪▪▪ сообщает о том, что реальный сервер отключен, ▪▪▪ заставит утилиту ▪▪▪▪▪ удалить этот сервер из таблицы IP. Основным назначением процесса *status*, работающего на активном балансировщике, является наблюдение за нагрузкой серверов, сбор и анализ статистики о текущем состоянии системы и интенсивности трафика. Процесс *status* передает обработанную и проанализированную информацию в подсистему балансировки нагрузки процессу *inf_status*. Процесс *inf_status* в свою очередь передает информацию о текущем состоянии системы процессу *coun_parce.*

Процесс *fractal* проводит расчет мультифрактальных свойств каждого входящего потока данных и передает эту информацию процессу *coun_parce*. Процесс *algor* выбирает алгоритм балансировки нагрузки и также передает эту информацию процессу *coun_parce*. Процесс *coun_parce* проводит расчет распределения потоков по узлам сети с учетом классификации трафика и загруженности серверов и каналов связи. Результаты расчета передаются процессу *parce*, который осуществляет динамическое распределение трафика по различным каналам связи и узлам в зависимости от их текущего состояния. Также результаты расчета процесса coun_parce передаются процессу *qos* для гибкой настройки обеспечения качества обслуживания в соответствии с методами управления трафиком (управление кеш-памятью, пропускной способностью памяти и каналов, производительностью процессоров), если это необходимо. В случае применения методов управления трафиком процессом *qos*, информация об изменениях передается процессу *inf_status*.

4.2. Описание динамического метода балансировки мультифрактального трафика

На основании мультифрактальных свойств входящего трафика предлагается динамический метод балансировки трафика. В зависимости от изменений параметров входящего потока используются различные методы управления трафиком. Приведем пошаговое описание динамического метода балансировки нагрузки:

1. В поступающем на балансировщик трафике выделяется окно X, фиксированной длины T.

2. Находим выборочное значение функции обобщенного показателя Херста $h(q)$, значение параметра Херста $H = h(2)$ и диапазон значений обобщенного показателя Херста $\Delta h = h(q_{\min}) - h(q_{\max})$ для участка трафика в выделенном окне;

3. Проводим сбор и анализ статистической информации: интенсивности входящих потоков $\lambda_1, ..., \lambda_N$, доступной пропускной способности $Net_i(t)$, состояния серверов $CPU_i(t)$, $RAM_i(t)$ – объем свободного ЦПУ и объем свободной оперативной памяти i-го сервера в момент времени t соответственно.

3. На основе мультифрактальных свойств трафика (значения из п.2), интенсивности трафика вычисляем необходимое количество ресурсов для каждого q-го класса трафика.

4. Проводим расчет распределения потоков по узлам сети с учетом классификации трафика и загруженности серверов и каналов связи. На основе полученных данных определяется загруженность серверов на следующем шаге.

5. Распределяем трафик по серверам, согласно алгоритму балансировки в пределах каждого класса потока.

6. Проводим распределение недооценки количества ресурсов $Net_i(t)$, $CPU_i(t)$, $RAM_i(t)$. Переоценка не учитывается алгоритмом, т.к. не вносит существенных изменений.

7. Проводим сбор данных о загруженности серверов $Net_i(t)$, $CPU_i(t)$, $RAM_i(t)$ и передачу их в систему балансировки нагрузки для расчета нового распределения потоков.

8. Передвигаем окно X длины T вперед на заданную величину сдвига ΔT; осуществляем анализ трафика и расчет следующего значения загруженности серверов.

Разрабатываемый метод балансировки нагрузки должен обеспечивать статистически равномерное распределение нагрузки на серверах, высокие показатели производительности, пропускной способности, отказоустойчивости (автоматически обнаруживая сбои узлов и перераспределяя поток данных среди оставшихся) и низкое время отклика, количество служебной информации, количество потерянных данных. Алгоритм балансировки должен распределять запросы по серверам так, чтобы отклонение загруженности серверов от среднего значения было минимальным.

Благодарности

Работа опубликована при поддержке проекта ITHEA XXI общества ITHEA ISS (www.ithea.org) и ADUIS (www.aduis.com.ua).

Выводы

В работе проведен анализ основных алгоритмов балансировки нагрузки по различным показателям производительности, т.е. для каждого алгоритма указаны показатели эффективности, которые в нем используются. Указаны основные особенности и сфера применения каждого алгоритма, определены достоинства и недостатки каждого алгоритма. Таким образом, можно выбрать конкретный алгоритм балансировки нагрузки, исходя из специфики выполняемой задачи или проекта и из целей, которые планируется достичь.

В работе предложена математическая модель системы балансировки нагрузки, в которой балансировщик нагрузки описывается с помощью системы массового обслуживания. Состояния серверов описываются объемом свободного ЦПУ и объемом свободной оперативной памяти. Все значения параметров модели имеют зависимость от времени. Такая модель позволяет описывать поведение распределенной сети во времени для различных классов обслуживания

входящего трафика. Предложена стратегия балансировки нагрузки, которая включает в себя комплексное измерение общего уровня дисбаланса системы, а также среднего уровня дисбаланса каждого сервера.

Разработан динамический метод распределения нагрузки, который учитывает мультифрактальные свойства трафика. Предлагаемый метод балансировки нагрузки благодаря анализу и учету мультифрактальных свойств входного потока обеспечивает статистически равномерное распределение нагрузки на серверах, высокие показатели производительности и пропускной способности, а также снижение времени отклика и количества потерянных данных.

Литература

[Cardellini, 1999] Valeria Cardellini, Michele Colajanni, Philip S. Yu. Dynamic Load Balancing on Web-server Systems. IEEE Internet Computing. - Vol.3. - No.3. – 1999.– P.28-39.

[Cardellini, 2001] V. Cardellini. A performance study of distributed architectures for the quality of web services. Proceedings of the 34th Conference on System Sciences. – Vol.10. – 2001. - P.213-217.

[Casalicchio, 2001] E. Casalicchio, M.Colajanni, "A client aware dispatching algorithm for web clusters providing multiple services," Proceeding of the 10th International Conference on WWW, 2001, pp.535–544.

[Chen, 2013] H. Chen, F. Wang, N. Helian, G. Akanmu. User-priority guided min-min scheduling algorithm for load balancing in cloud computing. National Conference Parallel Computing Technologies. - 2013. – P.1-8.

[Dhinesh, 2013] Dhinesh Babu L.D., P. Venkata Krishna, Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing. – Vol 13(5). - 2013. - P.2292–2303.

[Elzeki, 2012] O. Elzeki, M. Reshad, M. Elsoud. Improved max-min algorithm in cloud computing. International Journal of Computer Applications. - Vol. 50(12). – 2012. - P. 22–27.

[Erl, 2013] Thomas Erl, Ricardo Puttini, Zaigham Mahmood. Cloud Computing: Concepts, Technology & Architecture. Prentice Hall. Ed.1st. – 2013. – P.528.

[Erl, 2015] Thomas Erl, Robert Cope, Amin Naserpour. Cloud Computing Design Patterns. Prentice Hall. Ed.1st. – 2015. – P.592.

[Ghanbari, 2012] Shamsollah Ghanbari, Mohamed Othman. A Priority based Job Scheduling Algorithm in Cloud Computing. International Conference on Advances Science and Contemporary Engineering (ICASCE). –V. 50. - 2012. – P.778-785.

[Ghuge, 2014] Kalyani Ghuge, Minaxi Doorwar. A Survey of Various Load Balancing Techniques and Enhanced Load Balancing Approach in Cloud Computing. International Journal of Emerging Technology and Advanced Engineering. - Volume 4(10). 2014. – P.410-414.

[Gupta, 2013] Rohit O. Gupta, Tushar Champaneria. A Survey of Proposed Job Scheduling Algorithms in Cloud Computing Environment. International Journal of Advanced Research in Computer Science and Software Engineering. – Vol.3(11). – 2013. – P.782-790.

[Hong, 2006] Y.S. Hong, J.H. No, S.Y. Kim. DNS-based load-balancing in distributed web-server systems. Proceeding, in: Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (WCCIA 2006). – 2006. – P.251-254.

[Hu, 1998] Y. Hu, R. Blake, D. Emerson. An optimal migration algorithm for dynamic load balancing. Concurrency: Practice and Experience. – V.10(6). – 1998. P. 467–483.

[Hui, 2008] Hui Li, Workload characterization, modeling and prediction in Grid computing, Thesis Universiteit Leiden, 2008, P.141.

[Ignatenko, 2010] E.G. Ignatenko, V.I. Bessarab, V.V. Turupalov, The algorithm of adaptive load balancing in cluster systems, Modeling and information technologies, Kyiv: IPME G.E. Puhova NAN of Ukraine, № 58, 2010, pp. 142-150.

[Ivanisenko, 2015] Ivanisenko I., Kirichenko L., Radivilova T. Investigation of self-similar properties of additive data traffic. Proc. X international scientific and technical conference CSIT. September, 2015. Pp.169-171.

[Kameda, 1997] Hisao Kameda, Lie Li, Chonggun Kim, Yongbing Zhang. Optimal Load Balancing in Distributed Computer Systems. Springer, Verlag London Limited.- London. - 1997. - P. 238.

[Kantelhardt, 2008] J.W. Kantelhardt. Fractal and Multifractal Time Series. 2008: http://arxiv.org/abs/0804.0747

[Kashyap, 2014] Dharmesh Kashyap, Jaydeep Viradiya. A Survey Of Various Load Balancing Algorithms In Cloud Computing. International Journal Of Scientific & Technology Research. - Vol.3(11). – 2014. - P.115-119.

[Katyal, 2013] Mayanka Katyal, Atul Mishra. A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment. International Journal of Distributed and Cloud Computing. – Vol.1(2). – 2013. – P.6-14.

[Kaur, 2014] Rajwinder Kaur, Pawan Luthra. Load Balancing in Cloud Computing. DOI: 02.ITC.2014.5.92 Association of Computer Electronics and Electrical Engineers. – 2014. – P.374-381.

[Keshav, 1997] S.Keshav. An Engineering Approach to Computer Networking. Addison-Wesley, Reading, MA. – 1997. - P. 215-217.

[Kopparapu, 2008] Kopparapu Chandra. Load balancing servers, firewalls, and caches. Published by John Wiley & Sons, Inc. – 2002. – P.208.

[Liu, 2013] Jing Liu, Xing-Guo Luo, Xing-Ming Zhang, Fan Zhang, Bai-Nan Li. Job Scheduling Model for Cloud Computing Based on Multi-Objective Genetic Algorithm. IJCSI International Journal of Computer Science. – V.10(1). - № 3. - 2013. – P.134-139.

[Mehta, 2011] H. Mehta, P. Kanungo, M. Chandwani, "Decentralized content aware load balancing algorithm for distributed computing environments," ICWET '11 Proceedings of the International Conference & Workshop on Emerging Trends in Technology, N-Y, 2011, pp. 370-375.

[Meyer, 1998] Richard A. Meyer, Rajive Bagrodia Parsec. User Manual. Release 1.1. UCLA Parallel Computing Laboratory. - 1998. – URL: pcl.cs.ucla.edu/projects/parsec.

[Mishra, 2012] Ratan Mishra, Anant Jaiswal. Ant colony Optimization: A Solution of Load balancing in Cloud. International Journal of Web & Semantic Technology (IJWesT). - Vol.3. - No.2. - 2012. - P.335-338.

[Raghava, 2014] N. S. Raghava, Deepti Singh. Comparative Study on Load Balancing Techniques in Cloud Computing. Open journal of mobile computing and cloud computing. – Vol.1. – No.1. – 2014. – P.18-25.

[Rajwinder, 2014] Rajwinder Kaur, Pawan Luthra. Load Balancing in Cloud Computing. Association of Computer Electronics and Electrical Engineers. – 2014. - P.374-381.

[Randles, 2010] Martin Randles, David Lamb, A. Taleb-Bendiab. A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing. IEEE 24th International Conference on Advanced Information Networking and Applications Workshops. – 2010. –P.551-556.

[Ray, 2012] Soumya Ray, Ajanta De Sarkar. Execution analysis of load balancing algorithms in cloud computing environment. International Journal on Cloud Computing: Services and Architecture (IJCCSA). - Vol.2. - No.5. – 2012. - P.2657-2664.

[Red Hat, 2015] Red Hat Enterprise Linux. Обзор планирования распределения нагрузки. – 2015. - URL: https://access.redhat.com/documentation/ru-RU/Red_Hat_Enterprise_Linux/6/html/Virtual_Server_Administration/s1-lvs-scheduling-VSA.html.

[Sheluchin, 2007] Sheluchin O. I. Self-Similar Processes in Telecommunications / O. I. Sheluchin, S. M. Smolskiy, A. V. Osin // New York : John Wiley & Sons. – 2007. – 320 p.

[Singhal, 2011] Priyank Singhal, Sumiran Shah. Load Balancing Algorithm over a Distributed Cloud Network. 3rd IEEE International Conference on Machine Learning and Computing. – Singapore. – 2011. - P.37-42.

[Sran, 2013] Nayandeep Sran,Navdeep Kaur, "Comparative Analysis of Existing Load Balancing Techniques in Cloud Computing", International Journal of Engineering Science Invention, vol 2. 2013, pp.60-63.

[Tian, 2014] Wenhong Tian, Yong Zhao. Optimized Cloud Resource Management and Scheduling: Theories and Practices. Morgan Kaufman. 2014. P.284.

[Zaborowski, 2013] V.S. Zaborowski, A.S. Il'yashenko, V.A. Mulyuha, Simulation modeling of telematics systems. Proc., St. Petersburg: Publishing House of of the SPbSPU, 2013, P.58.

[Zheng, 2006] H. Zheng, L. Zhou, J. Wu, Design and implementation of load balancing in web server cluster system, Journal of Nanjing University of Aeronautics & Astronautics, Vol. 38, No. 3, Jun. 2006

[Zhihao, 2013] Zhihao Shang, Wenbo Chen, Qiang Ma, Bin Wu. Design and implementation of server cluster dynamic load balancing based on OpenFlow. Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA). - 2013. – P.691 – 697.

[Гуревич, 2010] Григорий Гуревич. Crescendo Networks - эволюция в мире WEB балансировки. – 2010. - URL: http://profyclub.ru/docs/99.

[Игнатенко, 2011] Е.И.Игнатенко. Адаптивный алгоритм мониторинга загруженности сети кластера в системе балансировки нагрузки. / Е.И.Игнатенко, В.И.Бессараб, И.В.Дегтяренко // Наукові праці ДонНТУ. Вип.21(183). 2011. С.95-102.

[Кириченко, 2011] Л.О. Кириченко, Э. Кайали, Т.А. Радивилова. Анализ методов повышения QoS в сетях MPLS с учетом самоподобия трафика. Системні технології. – 2011. – Вип. 3. – С. 52–59.

[Шелухин, 2011] Шелухин О. И. Мультифракталы. Инфокоммуникационные приложения / О. И. Шелухин. – М. : Горячая линия – Телеком, 2011. – 576 с.

Authors' Information

Igor Ivanisenko *– post graduate student, department of Applied Mathematics, Kharkiv National University of Radioelectronics; 14 Lenin Ave., 61166 Kharkiv, Ukraine;*

e-mail: ivanisenko79@yahoo.com.

Major Fields of Scientific Research: Modeling of Network traffic behavior, Self-similar traffic properties in Cloud technologies, Modeling of Queuing Systems, Computer networks

Lyudmyla Kirichenko *– Doctor of Technical Sciences, Professor, Kharkiv National University of Radioelectronics; 14 Lenin Ave., 61166 Kharkiv, Ukraine; e-mail: ludmila.kirichenko@gmail.com.*

Major Fields of Scientific Research: Time series analysis, Stochastic self-similar and multifractal processes, Wavelets , Chaotic systems

Tamara Radivilova *– Ph. D., Associate professor, Kharkiv National University of Radioelectronics; 14 Lenin Ave., 61166 Kharkiv, Ukraine; e-mail: tamara.radivilova@gmail.com.*

Major Fields of Scientific Research: Wavelets and fractals, Computer systems and networks, NGN and MPLS technologies

методы балансировки с учеТом мультифрактальныХ СВОЙСТВ нагрузки

Игорь Иванисенко, Людмила Кириченко, Тамара Радивилова

Аннотация: *В работе проведен анализ основных динамических алгоритмов балансировки нагрузки в распределенных компьютерных системах по основным показателям производительности, указаны достоинства и недостатки каждого алгоритма. Приведено математическое описание распределенной системы балансировки нагрузки, которое учитывает время и загрузку каждого сервера. Приведены значения общего уровня дисбаланса системы, а также средний уровень дисбаланса каждого сервера. Предложен динамический метод распределения нагрузки, учитывающий мультифрактальные свойства сетевого трафика, на основании которых пересчитывается распределение потоков.*

Ключевые слова: *алгоритмы балансировки нагрузки, самоподобный и мультифрактальный трафик, балансировка нагрузки, распределенные системы, пропускная способность, использование ресурсов, дисбаланс системы.*

ACM Classification Keywords: *C.2.3 Network Operations - Network management, C.2.4 Distributed Systems - Client/server, Distributed applications*