возрастания цены на привозное топливо до 3500 т у.т., оптимальный объем производства возрастет от 4600 до 6600 т у.т. (в 1,4 раза). В этом случае максимальный экономический эффект возрастет с 6,5 до 13,7 млн. руб. в год (в 2,1 раза).

Список литературы

1. Моор В., Чермных И., Белухин Н. Швеция и возобновляемая энергетика // Информационно-аналитическое агентство «Деловые новости», 2017. [Эл. ресурс]. URL: http://delonovosti.ru/analitika/3966-shveciya-i-vozobnovlyaemaya-energetika.html. (дата обращения: 19.01.2018).

2. Павличенко В.В., Протопопова М.В., Гамбург К.З., Байрамова Э.М., Рудых А.В., Войников В.К. Генно-инженерный подход к созданию быстрорастущих форм древесных растений // Экосистемы озера Байкал и Восточной Азии. Материалы Всероссийской научной конференции с международным участием, 2014. С. 72–75.

3. Панцхава Е.С. Биоэнергетика в современном и будущем сельскохозяйственном производстве. Продовольственная безопасность. Гелиоэнергетика – новая научно-техническая революция XXI века. М.: РУСАЙНС, 2017. 306 с.

4. Цибульская С. Энергетическая верба как вариант для агробизнеса. Кейс Salix energy // Пропозиция, 2017. [Эл. ресурс]. URL: http://propozitsiya.com/energeticheskaya-verba- kak-variant- dlya-agrobiznesa-keys-salix-energy (дата обращения 29.01.2018).

5. Энциклопедия систем жизнеобеспечения. Знания об устойчивом развитии / Редактор Е.Е. Демидова [и др.]. - М.: МАГИСТР-ПРЕСС/ – 2005. – Том 2. – 1208 с.

Елена Валерьевна Губий –
ст. инженер Института систем энергетики СО РАН; 664033, Иркутск;
e-mail: egubiy@gmail.com

Валерий Иванович Зоркальцев –
д.т.н., в.н.с. Отдела региональных, экономических и социальных проблем Иркутского научного центра СО РАН; 664033, Иркутск; e-mail: zork@isem.irk.ru

ПРОЕКТИРОВАНИЕ И ИССЛЕДОВАНИЕ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ ПРЕДОСТАВЛЕНИЯ УСЛУГ

Джолдасбаев С.К., Балакаева Г.Т., Айдаров К.А., Куламбаев Б.О., Даркенбаев Д.К.

Институт Информационных и Вычислительных Технологий
Казахский национальный университет имени аль-Фараби

Аннотация. *В данной статье был проведен анализ алгоритмов систем предоставления услуг при балансировке нагрузки на сервер, подробно описаны все преимущества или недочеты того или иного решения с целью дельнейшего совершенствования в данном направлении.*

Так как в данной статье ведется обзор, введем краткие пояснения определения, что такое интеллектуальная система предоставления услуг, и какие задачи она решает. Никакая вычислительная система не может сравниться ни по пиковой производительности, ни по объему оперативной или дисковой памяти с теми суммарными ресурсами, которыми обладают компьютеры, подключенные к глобальным сетям. Сейчас в мире насчитывается множество компьютеров, в том числе рабочие станции, ПК, мощный сервер и кластеры, суперкомпьютерные системы. Многие из них объединены в сети. Благодаря различным службам, таким, как e-mail, ftp, www, в Интернете возможно обмениваться информацией между компьютерами. С развитием сети Интернет появилась концепция использования вычислительных ресурсов географически распределенных вычислительных систем, в том числе обычных персональных компьютеров, для решения сложных задач [6].

Объединяя несколько информационно-коммуникационных средств (компьютеры, смартфоны и т.д.) можно создавать серверы для определенных целей. Собственно говоря, сервер – это компьютер, выделенный из группы персональных компьютеров (или рабочих станций) для выполнения какой-либо сервисной задачи без непосредственного участия человека [1]. Качество откликов и переработки информации сервера будет заметно меняться с увеличением запросов потребителей, то есть оно начнет работать медленнее, выходить из строя и т.д., определенным образом снижать качество обслуживания клиентов. В начальных этапах увеличение мощности сервера можно решить путем увеличения аппаратного количества сервера. Но в определенный момент становится ясно, что ресурсы имеют ограничение, потому что, во-первых, оно дорого стоит, во-вторых, занимает огромное пространство. К тому же если беспорядочно подключить два и более сервера, может произойти так, что к одному серверу будут подключаться небольшое количество клиентов, а к другому выстраиваться огромная очередь (рис-2).

Рис. 1 – общее представление сервера

Более эффективным решением вышесказанной проблемы является балансировка нагрузки. Необходимость решения комплексных вычислительных задач, выполнение которых с использованием бытовых компьютеров потребует выделения существенного временного ресурса, обуславливает востребованность различных вариантов систем балансировки нагрузки, различающихся по архитектуре построения, сложности и стоимости. Системы балансировки по применяемым решениям нагрузок делятся на три категории: аппаратные устройства (hardware appliances), сетевые коммутаторы и программные решения [4]. Применения данных

решений соответствует принципу построения высокопроизводительных систем, под названием «кластер». Кластер является системой, внешне функционирующей как единое целое, но внутри представленной множеством связанных компьютеров, служащих для решения общих задач [5]. Таким образом, их вычислительные ресурсы объединены с целью повышения эффективности решения общей задачи. Как правило, включенные в кластер компьютеры, по своей производительности не сильно превосходят обычные стационарные персональные компьютеры. Одним из основных составляющих любого кластера является модуль балансировки нагрузки. Он выполняет задачи управления ресурсами кластера, то есть, распределения поступающих на обработку вычислительных задач. Разработано множество алгоритмов функционирования модуля балансировки нагрузки, учитывающих существенное количество параметров, которые являются отражением особенностей системы, результатом их интерпретации конкретным проектировщиком [2].

Эффективность функционирования кластеров зависит от модуля балансировки нагрузки, а именно, от применяемого в нем **алгоритма балансировки нагрузки**.

Сервер 1 Сервер 2

Рис. 2 – Неравномерная нагрузка на 2 сервера

Существует множество решений, методов и алгоритмов данной задачи. В данной работе рассматриваются часто применяемые решения, учитывая их специфику, преимущества и недостатки в сравнительной степени.

Основные цели балансировки нагрузки следующие:

1) Распределение нагрузки между серверами – основная задача балансировки, то есть равномерное или же оптимальное распределение нагрузки на сервера;

2) Повышение отказоустойчивости системы – в случае выхода из строя одного из компонентов перераспределение нагрузки по другим узлам;

3) Защита от некоторых видов атак на сервер.

Определим требования к системе:

1) Справедливость. Балансировка должна дать возможность обслужить любой запрос, пришедший в систему.

2) Эффективность. Балансировка должна обеспечить такую работу, чтобы все сервера в кластере работали равномерно и брали равномерную нагрузку или распределялись таким образом, что более мощные брали больше перерабатываемых данных и т.д.

3) Сокращение времени выполнения запроса, а так же сокращение времени отклика. С помощью балансировки должно сокращаться время выполнения запроса.

4) Предсказуемость. Четкое определение какой алгоритм балансировки в каком случае эффективнее всего можно будет применить.

5) Равномерность загрузки системы. То есть, правильное распределение пакетов по узлам. Это схоже с эффективностью.

6) Масштабируемость. При резком скачке нагрузки система балансировки должна предоставлять стабильную работу данного сервиса.

По масштабности или географическому признаку серверов балансировку нагрузки можно на две группы: локальная – все сервера расположены внутри одного дата-центра и глобальная – ресурс раскидан на сервера по разным дата-центрам [3]. Локальную систему балансировки можно применять на канальном, сетевом и транспортном уровне. При глобальной балансировке применяются более сложные и развернутые алгоритмы, в основным прикладного уровня.

1. Round Robin

Самым простым, с точки зрения, как реализации, так и функционирования, является алгоритм «циклическая очередность» (Round Robin). Round Robin (далее – RR), по-другому алгоритм кругового обслуживания, представляет собой перебор по круговому циклу: первый запрос передаётся первому серверу, и каждый раз, когда сервер освобождается, следующий запрос передаётся второму и так далее, до достижения конечного сервера, затем циклический всё начинается сначала. Рассмотрим каким образом алгоритм может быть аппроксимирован технически.

Например, пусть в рассматриваемом маршрутизаторе N очередей, пронумеруем их i [1, N], и пусть обслуживание начинается с очереди номер 1 и продолжается в соответствии с увеличением номера очереди - в этом случае порядок обслуживания будет выглядеть следующим образом: 1, 2, ... N, 1,2, 3, ... и т.д. Если в очереди, когда к ней обращается планировщик, отсутствует пакет, то планировщик обращается к следующей по порядку очереди [9].

Существенным недостатком, делающим алгоритм непригодным для реализации в реальном оборудовании - RR не обеспечивает принцип «справедливого распределения ресурсов» для случаев, когда пакеты имеют переменную длину.

Самой распространённой имплементацией этого алгоритма является, конечно же, метод балансировки RR DNS. Как известно, любой DNS-сервер хранит пару «имя хоста — IP-адрес» для каждой машины в определённом домене. Этот список может выглядеть, например, так:

example.com xxx.xxx.xxx.2
www.example.com xxx.xxx.xxx.3

С каждым именем из списка можно ассоциировать несколько IP-адресов:

example.com xxx.xxx.xxx.2
www.example.com xxx.xxx.xxx.3
www.example.com xxx.xxx.xxx.4
www.example.com xxx.xxx.xxx.5
www.example.com xxx.xxx.xxx.6

DNS-сервер проходит по всем записям таблицы и отдаёт на каждый новый запрос следующий IP-адрес: например, на первый запрос — xxx.xxx.xxx.2, на второй — xxx.xxx.xxx.3, и так далее. В результате все серверы в кластере получают одинаковое количество запросов.

Положительными сторонами данного алгоритма можно указать, первое, независимость от протокола высокого уровня. Для работы по алгоритму RR используется любой протокол, в котором обращение к серверу идёт по имени. Балансировка на основе алгоритма Round Robin никак не зависит от нагрузки на сервер: кэширующие DNS-серверы помогут справиться с любым наплывом клиентов. Второе, использование алгоритма Round Robin не требует связи между серверами, поэтому он может использоваться для локальной и глобальной балансировки. Третье, решения алгоритма Round Robin характеризуется низкой стоимостью - для работы достаточно добавить несколько записей в DNS.

Стоит отметить, алгоритм RR имеет целый ряд недостатков. Чтобы распределение нагрузки по данному алгоритму отвечало упомянутым выше критериям справедливости и эффективности, у каждого сервера должен быть в наличии одинаковый набор ресурсов. При выполнении всех операций также должно быть задействовано одинаковое количество ресурсов. В реальной практике вышесказанные условия в большинстве случаев невыполнимы. Также при балансировке по алгоритму Round Robin совершенно не учитывается загруженность того или иного сервера в составе кластера. Допустим, что один из узлов загружен на 100%, в то время как другие узлы всего на 10-15%. Алгоритм RR возможности возникновения такой ситуации не учитывает, поэтому перегруженный узел будет получать запросы, в таком случае, не может быть и речи о справедливости, эффективности и предсказуемости. Еще один большой недостаток – в RR совершенно не учитывается количество активных на данный момент подключений [7]. Отметим, что в силу описанных выше обстоятельств, сфера применения алгоритма RR весьма ограничена.

2. Weighted Round Robin

Weighted Round Robin – усовершенствованная версия алгоритма RR. Суть улучшения алгоритма заключается в том, что каждому серверу присваивается весовой коэффициент в зависимости с производительностью и мощностью узла. Это предоставит возможность распределять нагрузку более разумно, то есть, серверы с большим весом обрабатывают больше запросов. Однако, данный метод не решает всех проблем с отказоустойчивостью. Для более эффективной балансировки при планировании и распределении нагрузки следует учитывать большее количество параметров.

3. Least Connections

Как указывалось выше, одним из недостатков алгоритма RR – количества активных подключений на момент обращения, рассмотрим практический пример. Имеется два сервера – обозначим их условно как S1 и S2. К серверу S1 подключено меньше пользователей, чем к серверу S2. При этом, сервер S1 оказывается более перегруженным. Данное положение обосновывается тем, что подключения к серверу S1 поддерживаются в течение более долгого времени по сравнению с подключениями к другому серверу (S2). Описанную проблему можно решить с помощью алгоритма, известного под названием least connections (leastconn). Данный алгоритм учитывает количество подключений, поддерживаемых серверами в текущий момент времени. Каждый следующий вопрос передаётся серверу с наименьшим количеством активных подключений.

Существует усовершенствованный вариант данного алгоритма, предназначенный очередь для использования в кластерах, состоящих из серверов с

разными техническими характеристиками и разной производительностью. Он называется Weighted Least Connections и учитывает при распределении нагрузки не только количество активных подключений, но и весовой коэффициент серверов.

В числе других усовершенствованных вариантов алгоритма Least Connections следует прежде всего выделить Locality-Based Least Connection Scheduling и Locality-Based Least Connection Scheduling with Replication Scheduling.

Первый метод был создан специально для кэширующих прокси-серверов. Его суть заключается в том, что наибольшее количество запросов передаётся серверам с наименьшим количеством активных подключений. За каждым из клиентских серверов закрепляется группа клиентских IP. Запросы с этих IP направляются на главный сервер, если он не загружен. В противном случае запрос будет перенаправлен на другой сервер.

В алгоритме Locality-Based Least Connection Scheduling with Replication Scheduling каждый IP-адрес или группа IP-адресов закрепляется не за отдельным сервером, а за целой группой серверов. Запрос передаётся наименее загруженному серверу из группы. Если же все серверы из главной группы перегружены, то будет зарезервирован новый сервер. Этот новый сервер будет добавлен к группе, обслуживающей IP, с которого был отправлен запрос. В свою очередь наиболее загруженный сервер из этой группы будет удалён – это позволяет избежать избыточной репликации.

4. Destination Hash Scheduling, Source Hash Scheduling

Алгоритм Destination Hash Scheduling был создан для работы с кластером кэширующих прокси-серверов, но часто используется и в других случаях. В этом алгоритме сервер, обрабатывающий запрос, выбирается из статической таблицы по IP-адресу получателя. Алгоритм Source Hash Scheduling основывается на тех же самых принципах, что и предыдущий, только сервер, который будет обрабатывать запрос, выбирается из таблицы по IP-адресу отправителя.

5. Sticky Sessions

Sticky Sessions алгоритм распределения входящих запросов, при котором соединения передаются на один и тот же сервер группы. Он используется, например, в веб-сервере Nginx [10,11]. Сессии пользователя могут быть закреплены за сервером с помощью метода IP hash [12]. С помощью данного метода запросы распределяются по серверам на основе IP-адреса клиента. Как указано в документации (см. ссылку выше), «метод гарантирует, что запросы одного и того же клиента будет передаваться на один и тот же сервер». Если закреплённый за конкретным адресом сервер недоступен, запрос будет перенаправлен на другой сервер. Пример фрагмента конфигурационного файла:

```
upstream backend {
ip_hash;

server backend1.example.com;
server backend2.example.com;
server backend3.example.com;
server backend4.example.com;
}
```

Начиная с версии 1.2.2 в Nginx для каждого сервера можно указывать вес. Применение этого метода сопряжено с некоторыми проблемами. Проблемы с привязкой сессий могут возникнуть, если клиент использует динамический IP. В ситуации, когда большое количество запросов проходит через один прокси-сервер, балансировку вряд ли можно назвать эффективной и справедливой. Описанные проблемы, однако, можно решить, используя cookies. В коммерческой версии Nginx имеется специальный модуль sticky, который как раз использует cookies для балансировки. Есть у него и доступные аналоги, такие как ***nginx-sticky-module***.

Заключение

В данной работе были описаны основные алгоритмы балансировки нагрузки серверов. Далее мы собираемся рассмотреть алгоритмы динамической балансировки в GRID технологиях [13,14]. Данные методы обусловлены тем, что при значительно масштабных ресурсах вышеуказанные алгоритмы балансировки нагрузки не удовлетворяют многих возложенных на них требовании. В работе так же не описаны виды топологии для серверов, что является немаловажным при решении в данном направлении. Для применения более гибких методов планируется работы по приоритетному планированию балансировки [15]. Обусловлены данные направления тем, что к определенной задаче можно использовать конкретный метод решения. Указанные задачи будут рассмотрены в дальнейшем.

Литература

[1] Windows Server Administration Fundamentals. Microsoft Official Academic Course. 111 River Street, Hoboken, NJ 07030: John Wiley & Sons. 2011. pp. 2–3. ISBN 978-0-470-90182-3.

[2] А.В. Бабич, Г.Б. Берсенев «Алгоритмы динамической балансировки нагрузки в распределенной системе активного мониторинга» Известия ТулГУ. Технические науки. 2011, Управление, вычислительная техника и информационные технологии, выпуск 5.Ч.3, УДК 681.3

[3] Сергей Зубов «Сравнительный анализ методов балансировки трафика» URL: https://habr.com/company/oleg-bunin/blog/319262/, дата обращения 03.05.2018

[4] Тао Чжоу «Системы балансировки нагрузки Web-серверов», 24.04.2000, URL: https://www.osp.ru/winitpro/2000/03/174228/, дата обращения 07.05.2018.

[5] Черняк Л. Crid как будущее компьютинга / Л. Черняк // Открытые системы. – 2003. – № 1. – С. 16-19.

[6] Андрей Емельянов «Балансировка нагрузки: основные алгоритмы и методы» URL: https://blog.selectel.ru/balansirovka-nagruzki-osnovnye-algoritmy-i-metody/, дата обращения 5.02. 2018

[7] К. А. Баркалов, В. В. Рябов, С. В. Сидоров, О некоторых способах балансировки локального и глобального поиска в параллельных алгоритмах глобальной оптимизации, Выч. мет. программирование, 2010, том 11, выпуск 4, 382–387

[8] Алгоритм Round Robin URL: http://www.opengl.org.ru/upravlenie-trafikom-i-kachestvo-obsluzhevaniya-v-seti/algoritm-round-robin.html дата обращения 11.06.2018

[9] И. Сысоев Nginx [engine x], URL: https://nginx.org/ru/, дата обращения 10.05.2018

[10] Nginx URL: https://www.nginx.com/?_ga=2.153782648.1323571196.1529313780-883474894.1529313780, дата обращения 10.05.2018

[11] Understanding IP Hash load balancing (2006129) URL: https://kb.vmware.com/s/article/2006129, дата обращения 10.05.2018

[12] Сатымбеков М.Н, Динамическая балансировка загруженности узлов кластера с приминением мультиагентных систем, "Вестник КазАТК", КАЗАХСТАН, рекомендуемый ККСОН МОН РК, издательство: Вестник КазАТК № 2 (101), 2017

[13] Pak I. T., Naizabayeva L., Nurzhanov Ch. A. Multi-agent grid system Agent-GRID with dynamic load balancing of cluster nodes (2017) Open Engineering, 7 (1), pp. 485-490. (Scopus).

[14] Айдаров Канат, Балакаева Гульнар "Исследование алгоритмов и методов балансировки нагрузки и построение моделей для сетей массового обслуживания", Марчуковские научные чтения – 2017, Труды Международной научной конференции. 2017, Издательство: Институт вычислительной математики и математической геофизики Сибирского отделения РАН (Новосибирск)

УДК 517.622.1, 519.624.1

ОБ ОДНОМ МЕТОДЕ РЕШЕНИЯ ЛИНЕЙНОЙ КРАЕВОЙ ЗАДАЧИ С ПАРАМЕТРОМ

Джумабаев Д.С., Минглибаева Б.Б.

Институт математики и математического моделирования, Алматы, Казахстан

Аннотация. *Рассматривается двухточечная краевая задача с параметром для обыкновенного дифференциального уравнения. Интервал разбивается на части, вводятся дополнительные параметры и исследуемая задача сводится к эквивалентной многоточечной краевой задаче с параметрами. Построена система линейных алгебраических уравнений относительно параметров. Коэффициенты и правые части системы определяются решением матричных и векторных задач Коши для обыкновенных дифференциальных уравнений на подинтервалах. Предложен численный метод решения рассматриваемой задачи, основанный на решении построенной системы.*

Ключевые слова: *краевая задача с параметром, дифференциальное уравнение, алгоритм.*

Рассмотрим двухточечную краевую задачу

$$\frac{dx}{dt} = A(t)x + B(t)\mu + f(t), \quad x \in R^n, \ \mu \in R^m, \quad t \in (0,T), \tag{1}$$

$$C_0\mu + C_1x(0) + C_2x(T) = d, d \in R^{n+m}, \tag{2}$$

где матрицы $A(t)$, $B(t)$ и вектор-функция $f(t)$ непрерывны на отрезке $[0,T]$, C_0 - $((n+m)\times m)$-матрица, C_1, C_2 -матрицы размерности $((n+m)\times n)$.