Математические структуры и моделирование 2024. № 2 (70). С. 87–99

УДК 004.75
DOI 10.24147/2222-8772.2024.2.87-99

БАЛАНСИРОВКА НАГРУЗКИ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ

М.Н. Ламановский[1]
аспирант, e-mail: maximlamanovksiy@gmail.com
Д.Н. Лавров[2]
канд. техн. наук, доцент, e-mail: dmitry.lavrov72@gmail.com

[1]Омский государственный университет им. Ф.М. Достоевского, Омск, Россия

[2]Нижневартовский государственный университет, Нижневартовск, Россия

Аннотация. Облачные вычисления становятся всё более распространёнными и популярными. Они предоставляют доступ к различным компьютерным услугам посредством Интернета. И распределение нагрузки играет ключевую роль как в качестве, так и в стоимости предоставляемых услуг. В данной работе будут рассмотрены популярные алгоритмы, которые применяются для балансировки нагрузки, а также способы использования машинного обучения для этих целей.

Ключевые слова: балансировка нагрузки, облачные вычисления, машинное обучение.

Введение

Облачные вычисления – это модель предоставления через Интернет компьютерных ресурсов, к которым могут относиться серверы, хранилища, базы данных, сети и так далее. Преимущество такого подхода заключается в том, что пользователь платит только за те ресурсы, которые он использует, и ему не надо беспокоиться о серверах и их техническом обслуживании, т. е. вся физическая реализация от него скрыта.

В настоящее время облачные вычисления становятся всё популярнее, что приводит к росту числа приложений и, соответственно, нагрузки на облачные сервисы. Это, в свою очередь, порождает некоторые проблемы и трудности, решение которых принесёт пользу как поставщику облачных ресурсов, так и конечному пользователю. К ним относят [1, 2]:

1. Контроль и управление. Требуется понимать, что все ресурсы доступны, используются все виртуальные машины и нет ошибок в конфигурации. Данная проблема обусловлена слишком большим разнообразием рабочих нагрузок, ограничением человеческого вмешательства и разнообразием совместно используемых приложений.

2. Масштабирование и динамическое распределение вычислительных ресурсов. Требуется понимать, куда направить очередной запрос, обеспечивать стабильную и эффективную работу всей системы, а также уметь определять, когда по-

являются новые узлы или старые становятся недоступными. Данная проблема обусловлена тем, что в облачных вычислениях задействовано несколько серверов, которые могут выходить из строя, также могут добавляться новые. Кроме того, необходимо следить за выделенными ресурсами, так как при возникновении большой нагрузки производительность всей системы может резко и сильно упасть, если не выделить дополнительную виртуальную машину под них. И ещё стоит учитывать, что не все существующие приложения рассчитаны на эффективную работу с распределёнными ресурсами.

3. Безопасность и конфиденциальность. Данная проблема возникает при совместном использовании одной физической машины несколькими облачными сервисами. Кроме того, стоит учитывать юридическую сторону вопроса, например, хранение персональных данных россиян только на серверах, расположенных в России, а также разнообразные атаки на облачный сервис.

4. Отказоустойчивость – ещё одна проблема облачных вычислений. Любой узел в любой момент может выйти из строя, и все данные на нём становятся недоступны или, что ещё хуже, потеряны полностью. Поэтому необходимо следить не только за активностью узлов, но и за сохранностью данных и запущенных вычислений. Также отказоустойчивость описывает поведение системы, когда она загружена.

Данная статья посвящена балансировке нагрузки в облачных вычислениях, которая может решать все проблемы, описанные выше [3].

1. Алгоритмы балансировки нагрузки

Балансировка нагрузки – это техника распределения запросов, вычислений и другой рабочей нагрузки между набором серверов с целью минимизации времени ответа и максимизации полученного результата, также она препятствует перегрузке какого-то конкретного сервера и может обеспечивать отказоустойчивость [3]. А сами алгоритмы балансировки нагрузки – это некоторый набор правил, используя которые балансировщик нагрузки решает, на каком конкретно сервере запустить конкретную задачу. Некоторые алгоритмы нацелены на то, чтобы как можно сильнее минимизировать время ответа, другие – на наибольшую оптимизацию использования ресурсов, третьи стремятся максимизировать результаты, а также существуют и такие, которые пытаются достигнуть компромисса между всеми метриками.

Существует несколько классификаций алгоритмов балансировки нагрузки в зависимости от учитываемых существенных признаков. Мы рассмотрим следующую классификацию алгоритмов балансировки нагрузки [4, 5, 7]:

1. Статические и динамические алгоритмы. Различаются уровнем учёта динамики. Так, в статических алгоритмах весь план распределения нагрузки известен заранее и не меняется со временем. В динамических алгоритмах план распределения нагрузки под воздействием разнообразных факторов, условий или просто по прошествии определённого количества времени перерасчитывается и изменяется.

2. Адаптивные и неадаптивные алгоритмы. Различаются по тому, как сильно они умеют приспосабливаться к изменению нагрузки на всю инфраструктуру. Как следует из названия, адаптивные алгоритмы умеют перераспределять ресурсы при изменении конфигурации всей системы или какой-то её части, а также выравнивать нагрузку при её резких изменениях. Неадаптивные алгоритмы, в свою очередь, не следят за изменением конфигурации и плохо реагируют на резкое изменение в нагрузке на систему.

3. Зависимые и независимые алгоритмы. Различаются по способу смены плана распределения. Так, в независимых алгоритмах есть заранее составленный набор статических планов, которые сменяют один другой в заранее определённой последовательности, а также могут зависеть от событий, происходящей в системе. В зависимом алгоритме отслеживаются изменения о характеристике нагрузки, временные метки и разнообразные внешние события, а когда случается любое из них – строится новый план балансировки, опирающийся на новую информацию.

4. Централизованные, иерархические и децентрализованные алгоритмы. Различаются способом управления. В централизованном, как можно судить из названия, существует некоторый центральный балансировщик, отвечающий за распределение нагрузки во всей системе. В иерархическом алгоритме вычислительные узлы объединяются в группы, внутри каждой группы выделяется свой балансировщик нагрузки, который и отвечает за всю группу, а также обмен данными о нагрузке в группе с остальными. И децентрализованном алгоритме в таком подходе каждый вычислительный узел отвечает сам за себя и решает самостоятельно, что делать с пришедшей задачей.

5. Универсальные и специализированные алгоритмы. Различаются по универсальности стратегии. Так, универсальные применимы к большинству архитектур, топологий сетей, алгоритмов и инфраструктур. Специализированные могут быть рассчитаны под какую-то определённую архитектуру, или топологию сети, или алгоритм, или инфраструктуру, или все перечисленные.

6. Прогностические и без способности предсказывать будущие изменения состояний. Как следует из названия, прогностические умеют с какой-либо точностью и долгосрочностью предсказывать ожидаемую нагрузку на систему и использовать данный прогноз для балансировки нагрузки. Алгоритмы без способности предсказывать будущие изменения состояний балансируют нагрузку, опираясь только на текущую информацию о системе.

7. Алгоритмы с учётом причин и без учёта причин разбалансировки стратегии. Алгоритмы, учитывающие причины разбалансировки нагрузки, могут рассматривать как внешние, так и внутренние причины и затем, опираясь на них, строить новые стратегии для балансировки нагрузки. Алгоритмы без учёта причин разбалансировки стратегии, в свою очередь, не учитывают какие-либо причины разбалансировки и не используют их для построения новой стратегии.

8. Алгоритмы, учитывающие только производительность узлов, и алгоритмы, которые учитывают производительность как узлов, так и сетевой подсистемы. Как можно судить из названия, алгоритмы первого типа учитывают производительность, пропускную способность и загруженность только узлов всей системы. Второй вид, кроме узлов, также учитывает и анализирует пропускную способность сети и характеристику трафика, распространяющегося по сети.

9. Приближенные и реалистичные алгоритмы. Различаются по степени оценки точности свободных ресурсов. Приближенные алгоритмы не используют или используют небольшое количество информации о степени загруженности системы. Реалистичные, в свою очередь, опираются на максимально детализированную информацию о системе для получения более точной оценки свободных ресурсов системы.

10. Алгоритмы, зависящие от того, кто выступает инициатором балансировки нагрузки – приёмник нагрузки или источник нагрузки. Если инициатором нагрузки выступает приёмник, то это означает, что балансировку нагрузки запускают недогруженные узлы системы. А в случае, когда инициаторы – источники нагрузки, то запуск балансировки нагрузки идёт со стороны перегруженных узлов системы.

Приведённая выше классификация демонстрирует, что какое-то универсальное решение, позволяющее покрывать все случаи, сделать очень сложно и проблематично, также позволяет лучше разобраться во внутренних принципах работы балансировщиков нагрузки и их устройстве. На данный момент одним из самых перспективных балансировщиков нагрузки можно считать динамическую адаптивную стратегию балансировки с возможностью прогнозирования будущих изменений нагрузки как в конкретном узле, так и во всей сети [3]. Давайте разберёмся почему это так, а также подробно рассмотрим существующие и предлагаемые алгоритмы, которые подходят или почти подходят под эту классификацию.

2. Статические и динамические алгоритмы

В данной главе мы детальнее рассмотрим статические и динамические алгоритмы, их преимущества и недостатки, также постараемся разобраться, почему динамические являются более перспективными для дальнейшего исследования и работы.

2.1. Статические алгоритмы

Алгоритмы статической балансировки нагрузки чаще всего основываются на знаниях об узлах системы, её внутреннем устройстве и о том, как система ведёт себя во время эксплуатации и под нагрузкой. Чаще всего такие алгоритмы стараются обеспечивать равномерное распределение трафика, хотя и есть исключения, например, рандомные алгоритмы. Статические алгоритмы очень хорошо подходят

для систем с предсказуемой нагрузкой, а в других случаях распределение нагрузки будет неэффективно.

Преимущества данного вида алгоритмов [7, 8]:

- Простота реализации. Данный вид алгоритмов часто очень простой, им не надо опираться на какие-то динамические данные и создавать новые планы для балансировки нагрузки. План балансировки появляется во время инициализации алгоритма, и вся работа идёт строго по нему.

- Скорость работы. Поскольку статические алгоритмы либо не учитывают информацию о системе, либо учитывают, но только во время первоначальной настройки или запуска, то они очень быстро могут решить, куда назначить появившуюся задачу, причём не задействуя большого количества ресурсов системы для тяжёлых вычислений.

- Хорошая эффективность в предсказуемых системах. Если система предсказуема, то и план балансировки нагрузки для неё можно определить заранее, что позволяет без дополнительных издержек использовать статическую балансировку с двумя преимуществами, описанными выше.

Недостатки статических алгоритмов балансировки нагрузки [7, 8]:

- Плохая эффективность в системах с переменной нагрузкой. Тут идём от противного: если в системе нагрузка переменная, да ещё и меняется довольно часто, то сделать для неё какой-то один план может быть сложно, из-за чего весь алгоритм начинает показывать плохую производительность.

- Невозможность обнаруживать изменения в системе. Поскольку статические алгоритмы учитывают информацию о системе либо во время составления плана, либо во время инициализации, то они не могут в реальном времени узнать о появлении нового узла или о том, что узел вышел из строя и ему не надо назначать новую нагрузку. В этом случае придётся либо переписывать план, либо перезапускать весь балансировщик.

- Задачи назначаются только во время создания и не могут быть перенесены на другой узел во время выполнения. Например, в данный момент система равномерно загружена, на каждом узле выполняется какая-то работа, появляется новая задача, для выполнения которой не хватает ресурсов ни на одном узле, из-за чего данная задача либо отправится в очередь, либо её выполнение будет отменено.

2.2. Динамические алгоритмы

Как уже было отмечено ранее, динамические алгоритмы балансировки нагрузки умеют перестраивать свой план балансировки под воздействием разнообразных внутренних факторов, внешних условий или просто по истечении определённого количества времени. Для составления нового плана данные алгоритмы учитывают

текущее состояние системы, её загруженность, свободные ресурсы. Они опираются на некоторый заранее заданный набор метрик системы для принятия решения о том, что делать с пришедшей задачей и на какой узел её отправлять.

К преимуществам динамических алгоритмов балансировки нагрузки можно отнести следующее:

- Способность адаптироваться к изменяющейся нагрузке. Поскольку такие алгоритмы умеют перестраивать свои планы балансировки нагрузки, то какие-то изменения в нагрузке оказывают меньшее влияние на их эффективность.

- Возможность учитывать изменяющиеся ресурсы системы. Если в системе появился новый узел или какой-то из уже запущенных узлов стал недоступен, то нет нужды в том, чтобы перезапускать балансировщик нагрузки, он сам учтёт и перестроит планы балансировки нагрузки с учётом произошедших изменений в системе.

- Точность распределения задач. Такие алгоритмы учитывают разнообразные факторы и метрики системы, соответственно, и задачи будут назначаться так, чтобы обеспечить более равномерную загруженность системы, чем в статических алгоритмах балансировки нагрузки.

- Возможность перераспределять задачи после начала их выполнения. Так, например, если в определённый момент времени нам понадобилось освободить какой-то узел, мы можем переназначить выполняемые задачи с него на другие узлы, а на освободившийся узел поставить новую задачу. Однако стоит учитывать накладные расходы на перемещение задач между узлами.

К недостаткам данного класса алгоритмов балансировки нагрузки стоит отнести следующие пункты:

- Накладные расходы. Динамические алгоритмы балансировки нагрузки требуют дополнительных ресурсов для сбора метрик, различных вычислений, перемещения задач и принятия решения о назначении задачи, что важно учитывать, так как это создаёт дополнительную нагрузку на систему и может повлиять на доступность ресурсов.

- Сложность реализации. Данный класс алгоритмов требует больше логики и осознанного решения, какие метрики будут учитываться, а какие – нет, а также насколько подробные данные нужно собирать.

В [6] проводится подробное сравнение двух статических и двух динамических алгоритмов балансировки нагрузки. Важно отметить, что динамические были реализованы из статических. Это делает данное сравнение более наглядным. В результате эксперимента было получено, что динамические алгоритмы, и правда, показывают себя лучше в распределении нагрузки, однако стоит очень внимательно рассматривать накладные расходы. Так, если накладные расходы будут слишком высоки, то эффективность статических и динамических алгоритмов становится минимальна, что делает динамические не особо привлекательными из-за их сложности.

Также важно учитывать, как часто балансировщик нагрузки получает данные о системе. Так, например, если время получения данных слишком высоко, то решения будут неточными и эффективность будет сильно ниже, а если время получения данных слишком низкое, то появится дополнительная нагрузка на систему. Ещё стоит отметить, что существуют исследования алгоритмов динамической балансировки нагрузки, которые иллюстрируют, что использование большого количества подробно собранных данных о системе даёт минимальный прирост производительности относительно алгоритмов, которые для принятия решения используют минимальное количество информации о системе. Из всего написанного выше можно сделать следующий вывод: динамические алгоритмы показывают себя значительно лучше, но необходимо правильно учесть все накладные расходы, определить оптимальную частоту получения информации и метрик о системе, а также определить оптимальный набор всех необходимых метрик и их структуру.

3. Адаптивные и неадаптивные алгоритмы

В этой главе мы рассмотрим классификацию алгоритмов, которые различаются по умению приспосабливаться как к изменениям в нагрузке на систему, так и к изменениям в самой системе. Стоит сразу отметить, что данная классификация возможна только для динамических алгоритмов, поскольку, как было отмечено ранее, статические алгоритмы не умеют приспосабливаться к каким-либо изменениям и их все сразу же можно отнести к неадаптивным алгоритмам. Динамические могут учитывать изменения, поэтому их можно разделить на адаптивные и неадаптивные.

3.1. Неадаптивные алгоритмы

Неадаптивные алгоритмы не учитывают любые происходящие изменения, к ним относятся все статические алгоритмы и те динамические алгоритмы, которые не отслеживают изменений или не реагируют на них, например, динамические алгоритмы, которые изменяют свои планы балансировки только по определённому расписанию.

Из преимуществ данного класса алгоритмов балансировки нагрузки можно выделить:

- Простота реализации. Статические алгоритмы реализовывать просто, так как в них нет никакой динамики и они следуют одному плану. А в динамических алгоритмах, которые не учитывают изменения, резко снижается количество параметров, которые надо учитывать, и схемы, по которым один план сменяется другим, становятся более лёгкими.

- Меньше накладных расходов. Поскольку мы не учитываем изменения в системе и нагрузке, то нам не надо хранить эту информацию, а также тратить вычислительные мощности на сбор и обработку данных об изменениях.

К недостаткам можно отнести следующие моменты:

- Недостаточная эффективность. Такие алгоритмы обеспечивают недостаточно хорошее распределение нагрузки, особенно в ситуациях с часто и сильно меняющимися условиями.

- Неуниверсальность. Поскольку данные алгоритмы не могут подстраиваться под изменяющиеся условия среды, то для сложных и динамических систем они будут показывать себя плохо, так как их придётся всё время донастраивать и обновлять.

3.2. Адаптивные алгоритмы

Адаптивные алгоритмы, как уже отмечалось ранее, отслеживают состояние системы и её частей. И опираясь на получаемые данные, указанный класс алгоритмов может как перераспределять, если появились новые машины в сети, или переназначать старые задачи, если вдруг какие-то узлы стали недоступны. А также в случае резких изменений загруженности они могут выравнивать нагрузку на всей системе.

К плюсам данного класса алгоритмов мы можем отнести следующее:

- Гибкость и адаптивность. Данные алгоритмы могут реагировать на изменения как в самой системе, так и в нагрузке на систему.

- Эффективность. За счёт динамической настройки и учёта различных факторов они обеспечивают более эффективное использование ресурсов всей системы при разных уровнях нагрузки.

- Масштабируемость. Заключается в способности алгоритмов учитывать появление новых узлов и исчезновение старых, что позволяет задействовать новые ресурсы и исключать старые без дополнительной настройки и перезапуска.

Минусы данных алгоритмов:

- Сложность реализации. Нужно больше времени, чтобы сделать такие алгоритмы, нужно понимать, какие параметры и условия учитывать, а также понимать, как обрабатывать их изменения.

- Накладные расходы. Тут всё очевидно, нам нужно постоянно собирать, запоминать и обрабатывать некоторый дополнительный набор данных. А значит, нам требуется больше вычислительных мощностей для работы таких алгоритмов. Также требуется понимать, что для миграции задачи между узлами тоже потребуются какие-то ресурсы.

Для того чтобы сделать выводы про неадаптивные и адаптивные алгоритмы, мы воспользуемся всё той же работой, в которой было проведено сравнение статических и динамических алгоритмов балансировки нагрузки [6]. Мы можем это сделать, поскольку неадаптивные отличаются от адаптивных примерно так же, как и статические от динамических. Во втором случае всегда требуется больше данных, больше вычислений, но взамен мы получаем более эффективный и точный алгоритм

балансировки нагрузки. Поэтому адаптивные алгоритмы показывают себя лучше, чем статические, но до тех пор, пока издержки на дополнительные вычисления, а теперь ещё и на миграцию задач между узлами, не становятся слишком большими, по сравнению с задачами в системе. И важно отметить, что если система небольшая, статичная и с предсказуемой нагрузкой, то стоит выбирать всегда неадаптивные алгоритмы, тут они покажут себя эффективнее. Но в настоящее время таких систем мало, нагрузки варьируются, в системах могут появляться новые узлы и исчезать старые, в таких условиях приоритет однозначно уходит к адаптивным алгоритмам. И теперь из всего написанного выше можно сделать вывод, что в современном мире с динамическими системами и разнообразной нагрузкой в приоритете всегда адаптивные алгоритмы из-за возможности подстраиваться под новые условия и умения работать с резкими изменениями в нагрузке.

4. Прогностические и без способности предсказывать будущие изменения состояний алгоритмы

Теперь перейдём к способности алгоритмов предсказывать будущие изменения состояний. В данной категории выделяют два класса, первый умеет предсказывать будущие изменения состояний, а второй – нет. В данной главе мы разберёмся, какие бывают виды предсказаний для алгоритмов балансировки нагрузки, а также почему прогностические алгоритмы являются более перспективными.

4.1. Алгоритмы без способности предсказывать будущие изменения состояний

Алгоритмы, которые не умеют предсказывать изменения, или, как их ещё можно назвать, реактивные алгоритмы балансировки нагрузки. Их суть заключается в том, что они реагируют на конкретную нагрузку, конкретное состояние системы и сети в текущий момент времени. Могут происходить ситуации, когда алгоритм, пытаясь обеспечить равномерную загрузку, распределил задачи равномерно на все узлы, но потом пришла задача или набор задач, которым необходимо полностью занять всю машину, из-за чего либо данная задача попадёт в очередь и будет ожидать, пока один из узлов освободится, либо начнётся миграция уже запущенных задач с одного узла на другой.

К плюсам данного вида алгоритмов можно отнести:

- Простота реализации. Тут всё очевидно – меньше кода, меньше логики, а значит, и меньше проблем с реализацией, а также не нужно проектировать этот самый алгоритм прогнозирования.
- Меньше накладных расходов. Поскольку нет дополнительной логики вычислений, не надо запоминать какие-то дополнительные данные, а значит, и накладных расходов для системы меньше.
- Меньшая точность и эффективность. Как уже было описано выше, может случиться такая ситуация, что очень сильно скажется на эффективности всей системы.

- Уязвимость к перегрузкам. Внезапные пики нагрузки могут затруднить работу всей системы, поскольку данные алгоритмы не умеют предсказывать их и предпринимать какие-то действия заранее.

4.2. Прогностические алгоритмы

Алгоритмы со способностью предсказывать будущее состояние системы и за счёт этого принимать более эффективные действия. Предсказания могут быть как краткосрочными, так и долгосрочными [4]. Краткосрочные, или микропрогнозирование, чаще всего выполняются на уровне квантифицируемых порций вычислений, критерием эффективности для них можно считать получаемую разницу между использованием прогноза с учётом накладных затрат и неиспользованием прогноза. Долгосрочное, или макропрогнозирование, используется для планирования нагрузки на достаточно большие периоды времени, когда важна оценка общих объёмов ресурсов с учётом факторов, которые действуют довольно длительное время. Также стратегии прогнозирования можно разделить на централизованные, когда все данные поступают в один узел и он выдаёт прогнозы для остальных узлов, и децентрализованные, когда каждый узел сам для себя вычисляет прогноз. Прогнозирование можно использовать для предсказывания состояния всей системы и, возможно, будущей нагрузки [3, 4, 9, 10].

На данный момент существует два основных способа для вычисления прогноза:

- Прогнозирование с использованием временных рядов [4, 9]. Для данного способа характерно использование исторических данных для предсказания будущих значений. Он основан на том, что сетевой трафик обладает свойством фрактальности [11], которое позволяет утверждать, что на больших масштабах сетевой трафик обладает свойством самоподобия. Поэтому формируется некоторая выборка достаточного размера, формируются временные ряды и применяются традиционные и распространённые статистические методы для их анализа, например, экспоненциальное сглаживание или ARIMA.

- Прогнозирование с использованием алгоритмов машинного обучения [3, 10]. Данный способ сформировался относительно недавно и за последние несколько лет продвинулся довольно далеко, сейчас существуют алгоритмы, которые позволяют прогнозировать развитие разнообразных болезней, изменения климата и многое другое. Нейронные сети так хорошо показывают себя в прогнозировании за счёт их способности обобщения и выделения скрытых зависимостей между входными и выходными данными [12, 13]. Однако для них очень важно собрать правильный набор данных, подготовить его и провести весь процесс обучения, что может занимать довольно много времени. Однако, существуют предобученные модели, которые очень хорошо подойдут на начальных и прототипных стадиях, но затем лучше обучить модель на своих данных для повышения точности прогноза.

К плюсам прогностических алгоритмов балансировки нагрузки можно отнести:

- Эффективность. Если алгоритм может предсказать изменение состояния, то он может выбрать лучший план действия для обеспечения равномерной загрузки всей системы.

- Оптимизация. Даёт возможность лучше регулировать динамические ресурсы, что позволит обработать внезапные наплывы запросов или, наоборот, снизить количество активных узлов, если в ближайшее время не ожидается большого числа задач.

Из минусов стоит выделить:

- Сложность реализации. Нужно собрать, проанализировать данные, реализовать алгоритм или обучить модель нейронной сети. Всё это требует времени и специализированных знаний.

- Дополнительные вычислительные издержки. Для анализа входящих данных и формирования прогноза требуются дополнительные вычислительные мощности.

В настоящее время актуальными являются прогностические алгоритмы на основе методов машинного обучения, так как они показывают себя лучше статистических методов за счёт выявления скрытых зависимостей, возможности дообучения на новых данных и гибкости, особенно если речь идёт о долгосрочных прогнозах. Однако важно учитывать точность прогноза и необходимые вычислительные издержки, поскольку в [4] даётся критерий эффективности использования прогностических алгоритмов, если затраты на их вычисления ниже, чем разность работы системы с прогнозом и без него. В настоящее время ведётся активная работа по использованию нейронных сетей для прогнозирования, применяются различные виды нейронных сетей и способы их обучения. Да и сами нейронные сети не стоят на месте и развиваются всё сильнее и сильнее с каждым годом. А как уже отмечалось выше, нагрузка, да и сами системы, в современном мире могут меняться довольно часто, что делает алгоритмы без прогнозирования неэффективными, по этой причине в большинстве систем предпочтение будет отдаваться прогностическому алгоритму, но важно оптимизировать все вычислительные издержки.

Выводы

В статье представлена классификация алгоритмов балансировки нагрузки и предложен перспективный вид алгоритма для балансировки нагрузки. Разобраны основные классификации, опираясь на которые и был предложен алгоритм. Выбран динамический адаптивный алгоритм с прогнозированием на основе методов машинного обучения, поскольку данное решение будет наиболее универсальным и эффективным в современном мире. Вся дальнейшая работа будет по проектированию и реализации балансировщика нагрузки, использующего данный алгоритм. Также требуется исследовать и выбрать метод прогнозирования нагрузки, построить математическую модель, собрать данные и провести эксперименты.

Литература

1. Коваленко О.С., Курейчик В.М. Обзор проблем и состояний облачных вычислений и сервисов // Известия ЮФУ. Технические науки. 2012. № 7. С. 146–153.
2. Маслова М.А., Кузьминых Е.С. Проблемы облачных сервисов и методы защиты от рисков и угроз // Научный результат. Информационные технологии. 2022. № 3. С. 14–22.
3. Muchori J., Peter M. Machine Learning Load Balancing Techniques in Cloud Computing: A Review // International Journal of Computer Applications Technology and Research. 2022. Vol. 11, No. 6. P. 179–186. DOI: 10.7753/IJCATR1106.1002.
4. Бершадский А.М., Курилов Л.С., Финогеев А.Г. Исследование стратегий балансировки нагрузки в системах распределённой обработки данных // Известия ВУЗов. Поволжский регион. Технические науки. 2009. № 4. С. 38–47.
5. Фраленко В.П., Агроник А.Ю. Средства, методы и алгоритмы эффективного распараллеливания вычислительной нагрузки в гетерогенных средах // Программные системы: теория и приложения. 2015. № 3 (26). С. 73–92.
6. Penmatsa S., Chronopoulos A.T. Dynamic multi-user load balancing in distributed systems // Parallel and Distributed Processing Symposium. IEEE International. IEEE Publisher, 2007. P. 1–10.
7. Данешманд Б., Ту Л.А. Исследование и обзор механизмов балансировки нагрузки на основе SDN в 5G/IMT-2020 // Вестник ВГТУ. 2022. № 1. С. 102–111.
8. Шуляк А.В. Сравнительный анализ алгоритмов балансировки нагрузки в среде облачных вычислений // Научный журнал. 2021. № 6 (61). С. 6–11.
9. Грушин Д.А., Кузюрин Н.Н. Балансировка нагрузки в системе Unihub на основе предсказания поведения пользователей // Труды ИСП РАН. 2015. № 5. С. 23–34.
10. Cordeiro-Costas M, Villanueva D, Eguía-Oller P, Martínez-Comesaña M, Ramos S. Load Forecasting with Machine Learning and Deep Learning Methods // Applied Sciences. 2023. Vol. 13, No. 13. Art. 7933. DOI: 10.3390/app13137933.
11. Барсуков И.С., Ряполов М.П. Использование фрактальных свойств трафика в цифровых сетях связи для детектирования сетевых аномалий // Вестник ВГУ. Серия: Системный анализ и информационные технологии. 2018. № 3. С. 73–81. DOI: 10.17308/sait.2018.3/1233.
12. Bojer C.S. Understanding machine learning-based forecasting methods: A decomposition framework and research opportunities // International Journal of Forecasting. 2022. Vol. 38, No. 4. P. 1555–1561. DOI: 10.1016/j.ijforecast.2021.11.003. URL: https://www.sciencedirect.com/science/article/pii/S0169207021001771 (дата обращения: 23.02.2024).
13. Любимова Т.В., Горелова А.В. Решение задачи прогнозирования с помощью нейронных сетей // Инновационная наука. 2015. № 4–2. С. 39–42.

LOAD BALANCING IN CLOUD COMPUTING

M.N. Lamanovsky[1]
Student, e-mail: maximlamanovskiy@gmail.com
D.N. Lavrov[2]
Ph.D. (Techn.), Associate Professor, e-mail: dmitry.lavrov72@gmail.com

[1]Dostoevsky Omsk State University, Omsk, Russia
[2]Nizhnevartovsk State University, Nizhnevartovsk, Russia

Abstract. Cloud computing is becoming more and more widespread and popular. They provide access to the variety of computer services via internet. And load balancing is a key both in the quality and in the cost of provided services. In this article. In this paper, popular algorithms that are used for load balancing will be considered, as well as ways to use machine learning for these purposes.

Keywords: load balancing, cloud computing, machine learning.

Дата поступления в редакцию: 06.03.2024