ИУ7-54Б, 16_KOZ, Булдаков

ОПРЕДЕЛЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие термины с соответствующими определениями.

Вычислительный узел (узел) — устройство, выполняющее основную логику обработки запроса [1].

Хеш-таблица — структура данных, реализующая интерфейс ассоциативного массива, позволяет хранить пары (ключ, значение) [2].

ВВЕДЕНИЕ

В современном обществе практически все общение и взаимодействие с приложениями осуществляются через интернет. Сетевые приложения востребованы как обычными пользователями, так и крупными корпорациями, проводящими сложные вычисления и обмен данными [3].

За последние 5 лет количество пользователей в интернете выросло на 30 процентов [4], что привело к резкому увеличению нагрузки на многие системы. Обработка выросшей нагрузки требует добавления в систему новых вычислительных узлов, для эффективной работы которых, необходимо осуществлять балансировку нагрузки [1; 5—7].

Целью данной научно-исследовательской работы является описание методов балансировки нагрузки в высоконагруженных системах.

1 Аналитический раздел

С ростом числа запросов к системе, встает вопрос о ее масштабировании. Масштабирование — это процесс роста системы со временем, для эффективной обработки все большего и большего количества запросов в единицу времени [8]. Выделяют два вида масштабирования: горизонтальное и вертикальное [1; 7; 9]. Вертикальное масштабирование происходит за счет увеличения мощности вычислительного узла. Однако, использования только такого подхода часто не хватает, поскольку постоянно увеличивая мощность, однажды будет достигнут «потолок» производительности и дальнейшие аппаратные улучшения будут недоступны. В таком случае для дальнейшего роста производительности применяют горизонтальное масштабирование, которое заключается в добавлении новых вычислительных узлов, выполняющих одинаковые функции. Для расширения возможностей горизонтального масштабирования используются балансировщики нагрузки [7; 9].

Балансировщик нагрузки — это программа, принимающая весь входящий трафик запросов и распределяющая его между несколькими вычислительными системами с целью оптимизации использования ресурсов, сокращения времени обслуживания запросов, а также обеспечения отказоустойчивости [9].

1.1 Постановка задачи

В вычислительных сетях, использование распределения нагрузки для оптимизации использования ресурсов, называют балансировкой нагрузки [6].

Постановка задачи балансировки нагрузки выглядит следующим образом: имеется множество, состоящее из n запросов, которое должно быть обслужено M узлами. Каждый узел может обслуживать не более одного запроса в каждый момент времени. Каждый запрос обслуживается не более, чем одним узлом в каждый момент времени, а процесс обслуживания запроса не может быть прерван.

Под расписанием понимается функция, которая каждому узлу l и моменту времени t сопоставляет запрос, обслуживаемый узлом l в момент времени t, либо указывает, что узел l в момент t простаивает. Каждому запросу i сопоставлена неубывающая функция штрафа $\phi_i(t)$. Тогда решением задачи балансировки нагрузки является составление расписания s, которое минимизирует выражение (1.1) [6].

$$F_{max} = \max_{i \in n}\{\phi_i(t_i(s))\}, \tag{1.1}$$

где $t_i(s)$ — момент завершения обслуживания запроса i при расписании s.

1.2 Алгоритмы решения задачи балансировки

Балансировщик нагрузки работает по одному из алгоритмов, решающих задачу балансировки. На вход этому алгоритму подается некоторое число запросов, приходящих в систему и набор вычислительных узлов, которыми располагает система. Задача алгоритма сводится к минимизации времени обработки запросов, за счет распределения запросов по вычислительным узлам.

Для анализа алгоритмов балансировки могут быть выделены следующие параметры [1]:

— точность прогнозирования — степень соответствия расчетных результатов работы алгоритма их фактическому значению;

— отказоустойчивость — показывает устойчивость алгоритма к возникновению разнообразных ошибок;

— время обработки нового запроса — время от поступления нового запроса до его перенаправления к цели.

Методы балансировки условно разделяют на статические и динамические [1; 10; 11].

При динамической балансировке нагрузки, распределение запросов происходит на основе собранной информации об узлах.

В статических алгоритмах, запрос распределяется в узел при появлении в балансировщике, а состояние узлов не оказывает влияния на это распределение [1; 11].

1.3 Статическая балансировка

К самым распространенным статическим алгоритмам относят: Round Robin и его модификацию Weighted Round Robin [1; 7; 9; 10].

1.3.1 Round Robin

Round Robin — это алгоритм балансировки нагрузки, который направляет каждый следующий запрос на новый вычислительный узел по заранее определенному порядку [9].

Пусть имеется N запросов и M узлов. Алгоритм состоит из следующих шагов.

1) Сформировать массив, содержащий узлы.

2) Создать переменную $i = 0$.

3) Для каждого запроса из N:

   — отправить текущий запрос на i-й узел в массиве;

   — увеличить значение i;

   — если $i >= M$, то $i = 0$.

Особенности алгоритма Round Robin:

— высокая степень точности прогнозирования;

— невозможно отследить вышел ли из строя узел, в результате, возможно, что на неработающий узел будут посылаться запросы, т. е. низкая отказоустойчивость;

— никакие затратные по времени операции не производятся и попавший в балансировщик запрос практически сразу направляется в узел;

— различия в технических характеристиках узлов не учитываются, что может привести к неравномерному распределению нагрузки.

Weighted Round Robin

Алгоритм Weighted Round Robin представляет собой модификацию алгоритма Round Robin, в которой каждому узлу вручную назначается некоторый параметр, называемый весом, с помощью которого можно варьировать количество запросов, отправляемых на конкретный узел [7]. Пусть помимо множества запросов и узлов, задан массив весов $weights$, длины M. Тогда алгоритм состоит из следующих шагов.

1) Сформировать массив $nodes$ содержащий узлы, при этом повторить каждый j-й узел $weights[j]$ раз.

2) Сохранить длину массива $nodes$ в переменную L.

3) Создать переменную $i = 0$.

4) Для каждого запроса из N:

   — распределить текущий запрос на i-й узел в массиве;

   — увеличить значение i;

   — если $i >= L$, то $i = 0$.

Особенности алгоритма Weighted Round Robin:

— высокая степень точности прогнозирования;

— низкая отказоустойчивость, поскольку невозможно отследить вышел ли из строя некоторый узел, при этом, если вышел из строя узел с самым высоким весом, то на него все еще будет посылаться большее число запросов;

— благодаря весам, возможно осуществить настройку алгоритма таким образом, чтобы он учитывал различия в технических характеристиках узлов.

1.4 Динамическая балансировка

Особенностью динамических алгоритмов балансировки является необходимость в постоянном обмене актуальной информацией о узлах. Простой способ периодического децентрализованного обмена информацией о состоянии заключается в том, что каждый узел периодически отправляет свое текущее состояние всем другим узлам [10].

К динамическим алгоритмам относят Dynamic Round Robin [10; 12].

1.4.1 Dynamic Round Robin

Алгоритм Dynamic Round Robin динамически изменяет расписание, т. е. распределение запросов по узлам, в зависимости от текущих характеристик

узлов [10]. Dynamic Round Robin может исключать недоступные узлы, перенаправляя задачи на доступные узлы, что позволяет избежать проблем при работе с неисправными узлами. В алгоритме Dynamic Round Robin на расписание могут влиять следующие характеристики [1; 7; 12]:

— количество соединений;

— среднее значение загрузки системы за период в 1 минуту, строится на основе процессов, т. е. программ в стадии выполнения, стоящих в очереди ожидания ресурсов, выражается как отношение количества ожидающих процессов к общему количеству ядер;

— загрузка процессора узла в текущий момент времени, выраженная в процентах;

— использование памяти узла, выраженное в процентах относительно общего количества;

— географическое расстояние между узлами.

Алгоритм Dynamic Round Robin, выбирающий узлы по их текущей нагрузке, состоит из следующих шагов для каждого входящего запроса [1].

1) Установить переменную *target* на первый доступный узел.

2) Цикл по всем доступным узлам, кроме первого:

   — если нагрузка на текущий рассматриваемый узел меньше нагрузки узла *target*, то установить *target* на текущий узел.

3) Отправить запрос на узел *target*.

Особенности алгоритма Dynamic Round Robin:

— низкая точность прогнозирования, поскольку распределение запросов сильно зависит от внешних факторов;

— высокая отказоустойчивость, поскольку в алгоритме учитывается ситуация отказа узлов.

ЗАКЛЮЧЕНИЕ

В результате проделанной научно-исследовательской работы, была описана предметная область балансировки нагрузки.

Было описано, что задача балансировки состоит в минимизации времени обслуживания запросов, а основными параметрами алгоритмов являются: отказоустойчивость, точность прогнозирования и время обработки запроса. Были рассмотрены такие алгоритмы статической балансировки, как Round Robin и Weighted Round Robin. Также были приведены основные различия между статическими и динамическими алгоритмами, в качестве примера динамических алгоритмов был рассмотрен алгоритм Dynamic Round Robin.

В результате, была достигнута цель научно-исследовательской работы, а именно, описаны методы балансировки нагрузки в высоконагруженных системах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *А.В. Ш.* СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ БАЛАНСИРОВКИ НАГРУЗКИ В СРЕДЕ ОБЛАЧНЫХ ВЫЧИСЛЕНИЙ // Научный журнал. — 2021. — № 6.
2. Хеш-таблица. — Режим доступа: `https://neerc.ifmo.ru/wiki/` (дата обращения: 07.11.2023).
3. *Onay Dogan B.*, *Camdereli M.* Digital Communication Activities of Corporations in the Context of Corporate Communication and Governance // Online Journal of Communication and Media Technologies. — 2015. — Апр. — С. 61—77.
4. Digital 2023: Global Overview Report. — Режим доступа: `https://datareportal.com/reports/digital-2023-global-overview-report` (дата обращения: 07.11.2023).
5. *Бершадский А. М. Курилов Л. С. Ф. А. Г.* Исследование стратегий балансировки нагрузки в системах распределенной обработки данных // Известия вузов. Поволжский регион. Технические науки. — 2009. — № 4.
6. *Бершадский А. М. Курилов Л. С. Ф. А. Г.* Разработка системы балансировки нагрузки // Гаудеамус. — 2012. — № 20.
7. АЛГОРИТМ РАСПРЕДЕЛЕНИЯ НАГРУЗКИ В ПРОГРАММНОЙ СИСТЕМЕ, ПОСТРОЕННОЙ НА ОСНОВЕ ПРОТОКОЛА HDP. — Режим доступа: `https://cyberleninka.ru/article/n/algoritm-raspredeleniya-nagruzki-v-programmnoy-sisteme-postroennoy-na-osnove-protokola-hdp` (дата обращения: 05.10.2023).
8. *Макаров Д.А. Ш. А.* МАСШТАБИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЙ // Теория и практика современной науки. — 2021. — № 1.
9. *А. О. Гусев В. В. Костылева И. Б. Р.* Сравнение алгоритмов балансировки нагрузки // Инновационное развитие техники и технологий в промышленности (ИНТЕКС-2020). — 2020. — № 1.
10. *В. С. Д.* Управление трафиком в сети с высокой динамикой метрик сетевых маршрутов // Вестник евразийской науки. — 2016. — № 1.

11. *Кхаинг Мин Тху Л. С.* СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ОЦЕНКИ ПРОИЗВОДИТЕЛЬНОСТИ УЗЛОВ В РАСПРЕДЕЛЕННЫХ СИСТЕМАХ // International Journal of Open Information Technologies. — 2023. — № 6.

12. *Haroon M.* Dynamic Load balancing by Round Robin and Warshall Algorithm in Cloud Computing // International Journal of Innovative Technology and Exploring Engineering. — 2021. — № 62.