УДК 676.22.017

Д. В. Котович, магистрант;
Д. В. Шиман, доц., канд. техн. наук
(БГТУ, г. Минск)

АЛГОРИТМЫ И МЕТОДЫ РАСПРЕДЕЛНИЯ ЗАДАНИЙ МЕЖДУ НЕСКОЛЬКИМИ СЕРВЕРАМИ

Вопрос о планировании нагрузки следует решать в начале разработки любого веб-проекта. «Падение» сервера может привести к серьёзными последствиями. Первоначально проблемы недостаточной производительности сервера в связи с ростом нагрузок можно решать путем наращивания производительности сервера, или же оптимизацией кода. Но рано или поздно наступает момент, когда и эти меры оказываются недостаточными[1].

Приходится прибегать к кластеризации: несколько серверов объединяются в кластер и нагрузка между ними распределяется при помощи комплекса специальных методов, называемых балансировкой. Балансировка нагрузки может осуществляться при помощи как аппаратных, так и программных инструментов.

Процедура балансировки осуществляется при помощи целого комплекса алгоритмов и методов, соответствующим следующим уровням модели OSI:

– Сетевому. Балансировка на сетевом уровне предполагает решение следующей задачи: нужно сделать так, чтобы за один конкретный IP-адрес сервера отвечали разные физические машины(DNS-баллансировка: на одно доменное имя выделяется несколько IP-адресов. Сервер, на который будет направлен клиентский запрос, обычно определяется с помощью алгоритма Round Robin; балансировка по территориальному признаку: осуществляется путём размещения одинаковых сервисов с одинаковыми адресами в территориально различных регионах Интернета);

– Транспортному. Этот вид балансировки является самым простым: клиент обращается к балансировщику, тот перенаправляет запрос одному из серверов, который и будет его обрабатывать.

– Прикладному. При балансировке на прикладном уровне балансировщик работает в режиме «умного прокси». Он анализирует клиентские запросы и перенаправляет их на разные серверы в зависимости от характера запрашиваемого контента.

В числе целей, для достижения которых используется балансировка, нужно выделить следующие:

- справедливость: нужно гарантировать, чтобы на обработку каждого запроса выделялись системные ресурсы и не допустить возникновения ситуаций, когда один запрос обрабатывается, а все остальные ждут своей очереди;
- эффективность: все серверы, которые обрабатывают запросы, должны быть заняты на 100%; желательно не допускать ситуации, когда один из серверов простаивает в ожидании запросов на обработку (сразу же оговоримся, что в реальной практике эта цель достигается далеко не всегда);
- сокращение времени выполнения запроса: нужно обеспечить минимальное время между началом обработки запроса (или его постановкой в очередь на обработку) и его завершения;
- сокращение времени отклика: нужно минимизировать время ответа на запрос пользователя;
- предсказуемость: нужно четко понимать, в каких ситуация и при каких нгрузках алгоритм будет наиболее эффективным, чтобы решить необходимые задачи;
- равномерная загрузка ресурсов системы;
- масштабируемость: алгоритм должен позволять масштабировать кластеризованную систему в случае, когда будет добавлено несколько новых серверов для обработки заросов.

Алгоритм Round robin[2], или алгоритм кругового обслуживания, представляет собой перебор по круговому циклу: первый запрос передаётся одному серверу, затем следующий запрос передаётся другому и так до достижения последнего сервера, а затем всё начинается сначала.

Алгоритм Weignted round robin — усовершенствованная версия алгоритма round robin. Суть усовершенствований заключается в следующем: каждому серверу присваивается весовой коэффициент в соответствии с его производительностью и мощностью. Это помогает распределять нагрузку более гибко: серверы с большим весом обрабатывают больше запросов. Однако всех проблем с отказоустойчивостью это отнюдь не решает. Более эффективную балансировку обеспечивают другие методы, в которых при планировании и распределении нагрузки учитывается большее количество параметров.

Рассмотрим практический пример. Имеется два сервера — обозначим их условно как А и Б. К серверу А подключено меньше пользователей, чем к серверу Б. При этом сервер А оказывается более перегруженным. Как это возможно? Ответ достаточно прост: подключения к серверу А поддерживаются в течение более долгого времени

по сравнению с подключениями к серверу Б.

Описанную проблему можно решить с помощью алгоритма, известного под названием least connections[3]. Он учитывает количество подключений, поддерживаемых серверами в текущий момент времени. Каждый следующий вопрос передаётся серверу с наименьшим количеством активных подключений.

Также, как и для round robin существует усовершенствованный вариант алгоритма least connection, предназначенный в первую очередь для использования в кластерах, состоящих из серверов с разной производительностью. Он называется Weighted Least Connections и учитывает при распределении нагрузки не только количество активных подключений, но и весовой коэффициент серверов.

Sticky Sessions[4] — алгоритм распределения входящих запросов, при котором соединения передаются на один и тот же сервер группы. Сессии пользователя могут быть закреплены за конкретным сервером. С помощью этого метода запросы распределяются по серверам на основе IP-адреса клиента. Применение этого метода сопряжено с возникновением проблемы привязки 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.

ЛИТЕРАТУРА

1. Что такое балансировка нагрузки [Электронный ресурс] / 8host.com. – Режим доступа: https://www.8host.com/blog/chto-takoe-balansirovka-nagruzki/ – Дата доступа: 19.01.2020.

2. Round Robin Scheduling - Wikipedia [Электронный ресурс] / wikipedia.org. – Режим доступа: https://en.wikipedia.org/wiki/Round-robin_scheduling – Дата доступа: 20.01.2020.

3. Least connection is not least loaded [Электронный ресурс] / devcentral.f5.com – Режим доступа: https://devcentral.f5.com/s/articles/back-to-basics-least-connections-is-not-least-loaded – Дата доступа: 22.01.2020.

4. Sticky session [Электронный ресурс] / imperva.com. – Режим доступа: https://www.imperva.com/learn/availability/sticky-session-persistence-and-cookies/ – Дата доступа: 25.01.2020.