В компьютерах балансировка нагрузки распределяет нагрузку между несколькими вычислительными ресурсами, такими как компьютеры, компьютерные кластеры, сети, центральные процессоры или диски. Цель балансировки нагрузки — оптимизация использования ресурсов, максимизация пропускной способности, уменьшение времени отклика и предотвращение перегрузки какого-либо одного ресурса. Использование нескольких компонентов балансировки нагрузки вместо одного может повысить надежность и доступность за счет резервирования. Балансировка нагрузки предполагает обычно наличие специального программного обеспечения или аппаратных средств, таких как многоуровневый коммутатор или система доменных имен, как серверный процесс.

Вышеприведенное определение применяется ко всем аспектам вычислений, а не только к сетям. Операционные системы используют балансировку нагрузки для планирования задач по физическим процессорам. например, в оркестраторах контейнеров, таких как Kubernetes, используется балансировка нагрузки для планирования задач в вычислительном кластере, а сетевые балансировки нагрузки используют балансировку нагрузки для планирования сетевых задач через доступные бэкенды. Остальная часть поста будет охватывать только балансировку сетевой нагрузки.

Обзор балансировки сетевой нагрузки

Рисунок показывает высокоуровневый обзор балансировки сетевой нагрузки. Некоторое количество клиентов запрашивает ресурсы из некоторого количества бэкендов. Балансировщик нагрузки находится между клиентами и внутренними компонентами и на высоком уровне выполняет несколько важных задач:

Обнаружение служб: Какие бэкенды доступны в системе? Каковы их адреса (например, как должен работать балансировщик нагрузки)?

Проверка работоспособности: Какие бэкенды в настоящее время здоровы и доступны для приема запросов?

Балансировка нагрузки: Какой алгоритм следует использовать для балансировки отдельных запросов по здоровым бэкендам?

Правильное использование балансировки нагрузки в распределенной системе дает несколько преимуществ:

Именование абстракции: Нет необходимости в том чтобы каждый клиент знал о каждом бэкенде (service discovery). Клиент может обращаться к балансировщику нагрузки через предопределенный механизм, а затем действие разрешения имен можно делегировать в балансировщик нагрузки. Предопределенные механизмы включают встроенные библиотеки и хорошо известные DNS/IP/port и будут обсуждаться более подробно ниже.

Отказоустойчивость: Благодаря проверке работоспособности и различным алгоритмическим методам балансировщик нагрузки может эффективно маршрутизировать вокруг неработоспособного или перегруженного бэкенда. Это означает, что администратор может спокойно, без спешки устранить проблему на нездоровом узле.

Стоимость и производительность: Распределенные системные сети редко бывают однородными. Наиболее вероятно, что система будет охватывать несколько сетевых зон и регионов. Интеллектуальная балансировка нагрузки может максимально увеличить трафик запросов в зонах, что увеличивает производительность (меньше латентности) и снижает общую стоимость системы (меньше полосы пропускания и прокладки оптоволокна, требуемого между зонами).

Балансировка нагрузки vs прокси-сервер

Когда речь идет о балансировщиках сетевой нагрузки, термины балансировки нагрузки и прокси используются примерно одинаково. Этот пост также будет рассматривать термины как эквивалентные. (Если быть точными, не все прокси-серверы являются балансировщиками нагрузки, но подавляющее большинство прокси-серверов выполняют балансировку нагрузки в качестве основной функции).

Есть мнение, что когда балансировка нагрузки выполняется как часть встроенной клиентской библиотеки, балансировщик нагрузки на самом деле не является прокси-сервером. Тем не менее, я бы сказал, что различие добавляет сложности уже запутанной теме. Типы топологий балансировки нагрузки подробно обсуждаются ниже, но этот пост рассматривает встроенную топологию балансировки нагрузки как просто особый случай проксирования; приложение проксирует через встроенную библиотеку, которая предлагает все те же абстракции, что и балансировщик нагрузки, находящийся вне процесса приложения.

L4 (connection/session) балансировка нагрузки

В масштабах отрасли сегодня решения по балансировке нагрузки часто разделяются на две категории: L4 и L7. Они относятся к уровням 4 и 7 модели OSI. Модель OSI представляет очень плохое приближение к сложности решений балансировки нагрузки, которые включают традиционные протоколы уровня 4, такие как TCP и UDP, но часто заканчиваются включением битов и частей протоколов на разных уровнях OSI. Возникает вопрос: если балансировщик нагрузки L4 TCP также поддерживает терминирование TLS, является ли он теперь балансировщиком нагрузки L7?

Базовая балансировка нагрузки TCP L4

Рисунок показывает традиционный балансировщик нагрузки L4 TCP. В этом случае клиент устанавливает TCP-соединение с балансировщиком нагрузки. Балансировщик нагрузки завершает соединение (т.е. отвечает непосредственно на SYN), выбирает бэкенд и устанавливает новое TCP-соединение с бэкендом (т.е. отправляет новый SYN). Детали диаграммы не важны и будут подробно обсуждаться ниже — в разделе, посвященном балансировке нагрузки L4.

Основной вывод этого раздела: заключается в том, что балансировщик нагрузки L4 обычно работает только на уровне соединения/сеанса L4 TCP/UDP. Таким образом, балансировщик нагрузки грубо перетасовывает байты взад и вперед и гарантирует, что байты с того же сеанса завершаются на одном и том же бэкенде. Балансировщик L4 не знает о каких-либо деталях байтов приложения, которые перетасовывает. Байтами могут быть HTTP, Redis, MongoDB или любой другой протокол приложения.

L7 (приложение) балансировка нагрузки

Балансировка нагрузки L4 проста и по-прежнему широко используется. Каковы недостатки балансировки нагрузки L4, которые гарантируют инвестиции в балансировку нагрузки L7 (приложение)? В качестве примера возьмем следующий конкретный случай L4:

Два клиента gRPC/HTTP2 хотят поговорить с бэкендом, поэтому подключаются через балансировщик нагрузки L4.

Балансировщик нагрузки L4 делает одно исходящее TCP-соединение для каждого входящего TCP-соединения, что приводит к двум входящим и двум исходящим соединениям.

Однако клиент A отправляет по 1 запрос в минуту (RPM), а клиент B отправляет 50 запросов в секунду (RPS) по его соединению.

В предыдущем сценарии бэкенд, выбранный для обработки клиента A, будет обрабатывать нагрузку примерно в 3000x раз меньше, чем бэкенд, выбранный для обработки клиента B! Эта серьезная проблема, как правило, в первую очередь нарушает цель балансировки нагрузки. Также обратите внимание, что эта проблема возникает для любого протокола мультиплексирования. (Мультиплексирование означает отправку одновременных запросов приложений по одному соединению L4, а keep-alive означает команду не закрывать соединение, когда нет активных запросов). Все современные протоколы развиваются как для мультиплексирования, так и для поддержания жизнеспособности по соображениям эффективности (как правило, соединения, которые шифруются с использованием TLS, создавать дорого). Поэтому с течением времени сопротивление балансировки нагрузки L4 становится более выраженным. Эта проблема устраняется балансировщиком нагрузки L7.

HTTP/2 балансировка нагрузки L7

Рисунок показывает балансировщик нагрузки L7 HTTP/2. В этом случае клиент делает одно соединение HTTP/2 TCP с балансировщиком нагрузки. Затем балансировщик выполняет два внутренних подключения. Когда клиент отправляет два потока HTTP/2 в балансировщик нагрузки, поток 1 отправляется на бэкенд 1, в то время как поток 2 отправляется на бэкенд 2. Таким образом, даже мультиплексирование клиентов, которые имеют значительно разные нагрузки на запросы, будет эффективно балансироваться по внутренним компонентам. Вот почему балансировка нагрузки L7 так важна для современных протоколов. (Балансировка нагрузки L7 дает также множество дополнительных преимуществ из-за возможности проверки трафика приложений, но более подробно это будет рассмотрено ниже).

Балансировка нагрузки L7 и модель OSI

Как я сказал выше в разделе о балансировке нагрузки L4, использование модели OSI для описания функций балансировки нагрузки проблематично. Причина в том, что L7 уже включает несколько дискретных уровней абстракции балансировки нагрузки (по крайней мере, как описано в модели OSI). Например, для HTTP-трафика рассмотрим следующие подуровни:

Дополнительная безопасность транспортного уровня (TLS). Обратите внимание: люди до сих пор спорят о том, в какой OSI-уровень входит TLS. Ради этой дискуссии мы рассмотрим TLS L7.

Физический протокол HTTP (HTTP/1 или HTTP/2).

Логический HTTP-протокол (headers, body data, и trailers).

Протокол обмена сообщениями (gRPC, REST, и т.д.).

Усовершенствованный балансировщик нагрузки L7 может предлагать функции, связанные с каждым из вышеперечисленных подслоев. Еще один балансировщик L7 может иметь только небольшой набор функций, которые помещают его в категорию L7. Короче говоря, ландшафт балансировки нагрузки L7 значительно сложнее с точки зрения сравнения функций, чем категория L4. (И, конечно, этот раздел только что затронул HTTP: Redis, Kafka, MongoDB и т.д. — всё это примеры протоколов приложений L7, которые выигрывают от балансировки нагрузки L7).

Функции балансировки нагрузки

В этом разделе я кратко расскажу о функциях высокого уровня, которые обеспечивают балансировки нагрузки. Не все балансировочные устройства обеспечивают все функции.

Обнаружение службы

Обнаружение служб — это процесс, с помощью которого балансировщик нагрузки определяет набор доступных бэкендов. Методы довольно разнообразны, и некоторые примеры включают:

Статичный файл конфигурации.

DNS.

Zookeeper, Etcd, Consul, и т.д.

Envoy’s универсальный data plane API.

Проверка работоспособности

Проверка работоспособности — это процесс, с помощью которого балансировщик нагрузки определяет, доступен ли бэкенд для обслуживания трафика. Проверка работоспособности обычно подразделяется на две категории:

Активная: Балансировщик нагрузки периодически отправляет ping (например, HTTP-запрос конечной точке /healthcheck) на бэкенд и использует это для оценки работоспособности.

Пассивная: Балансировщик нагрузки определяет состояние здоровья из основного потока данных. Например, балансировщик нагрузки L4 может решить, что бэкенд нездоровый, если в строке было три ошибки соединения. Балансировщик нагрузки L7 может решить, что бэкенд нездоровый, если в строке было три кода ответа HTTP 503.

Балансировка нагрузки

Да, балансировщики нагрузки должны фактически балансировать нагрузку! Учитывая набор здоровых бэкендов, как выбирается бэкенд, который будет обслуживать соединение или запрос? Алгоритмы балансировки нагрузки являются активной областью исследований и варьируются от упрощенных, таких как случайный выбор и циклический алгоритм, до более сложных алгоритмов, учитывающих переменную задержку и нагрузку на бэкенд. Один из самых популярных (учитывая его производительность и простоту) алгоритмов балансировки нагрузки известен как [power of 2 least request] (https://brooker.co.za/blog/2012/01/17/two-random.html).

Sticky sessions

В некоторых приложениях важно, чтобы запросы на один и тот же сеанс достигли одного и того же бэкенда. Это может быть связано с кешированием, временным сложным состоянием и т.д. Определение сеанса варьируется и может включать HTTP-файлы cookie, свойства клиентского соединения или какой-либо другой атрибут. У многих балансировщиков нагрузки L7 есть определенная поддержка липких сеансов. Отмечу, что липкость сеанса по своей сути является хрупкой (бэкенд-хостинг сеанса может умереть), поэтому будьте осторожны при разработке системы, которая опирается на них.

TLS-терминирование

Тема TLS и ее роль достойны отдельного поста. С учетом сказанного многие балансировочные устройства L7 выполняют обработку TLS, которая включает в себя терминирование, проверку сертификатов, фиксацию и т.д.

Возможность наблюдения

Как мне нравится говорить: «Наблюдение, наблюдение, наблюдение». Сети по своей сути ненадежны, и балансировщик нагрузки часто несет ответственность за экспорт статистики, следов и журналов, которые помогают операторам выяснить, что не так, чтобы они могли решить проблему. Балансировщики нагрузки сильно различаются по показателям наблюдаемости. Самые продвинутые балансировщики предлагают богатые возможности, которые включают числовые характеристики, распределенную трассировку и настраиваемое логирование. Я укажу, что усиленная наблюдаемость не является бесплатной; балансировщик нагрузки должен выполнить для этого дополнительную работу. Однако преимущества данных значительно перевешивают незначительные последствия для производительности сети.

Безопасность и предотвращение DoS

В частности, в топологии развертывания краев (см. ниже) балансировочные устройства часто реализуют различные функции безопасности, включая ограничение скорости, аутентификацию и смягчение DoS (например, маркирование и идентификацию IP-адреса, tarpitting и т.д.).

Конфигурация и плоскость управления

Балансировщики нагрузки нуждаются в настройке. В крупных развертываниях это может стать существенной задачей. Система, которая настраивает балансировщики нагрузки, известна как «плоскость управления» и широко варьируется в реализации. Для получения дополнительной информации по этой теме, пожалуйста, см. мое [сообщение о плоскости данных сетки обслуживания и плоскости управления] (https://medium.com/@mattklein123/service-mesh-data-plane-vs-control-plane-2774e720f7fc).

И еще многое другое

Этот раздел просто поцарапал поверхность типов функциональных возможностей, которые обеспечивают балансировщики нагрузки. Дополнительную информацию можно найти в разделе о балансировщиках L7 ниже.

Типы топологий балансировки нагрузки

Теперь, когда я предоставил высокоуровневое описание балансировки нагрузки, разницу между балансировщиками нагрузки L4 и L7 и краткое описание функций балансировки нагрузки, перейду к различным топологиям распределенных систем, в которых развернуты балансировщики нагрузки. (Каждая из следующих топологий применима как для балансировщиков нагрузки L4, так и для L7).

Прокси по середине

Топология балансировки нагрузки прокси по середине

Топология прокси по середине, показанная на рисунке выше, вероятно, является наиболее известным способом получения балансировки нагрузки для большинства читателей. Эта категория включает аппаратные устройства от Cisco, Juniper, F5 и т.д.; облачные программные решения, такие как Amazon ALB и NLB и Сloud Load Balancer; и чистые программные решения, такие как HAProxy, NGINX и Envoy. Достоинства подобной топологии — простота применения для пользователей. В общем случае пользователи подключаются к балансировщику нагрузки через DNS и не должны беспокоиться ни о чем другом. Соглашением решения прокси по середине является тот факт, что прокси (даже в случае кластеризации) является единственной точкой отказа, а также узким местом масштабирования. Прокси по середине также часто является черным ящиком, что затрудняет работу. Является ли наблюдаемая проблема у клиента? В физической сети? В прокси по середине? В бэкенде? Ответить на эти вопросы может быть очень сложно.

Краевой прокси

Топология балансировки нагрузки краевого прокси

Топология краевого прокси, показанная на рисунке, на самом деле является лишь вариантом топологии прокси по середине, в которой балансировщик нагрузки доступен через Интернет. В этом случае балансировщик нагрузки обычно должен предоставлять дополнительные функции «шлюза API», такие как терминирование TLS, ограничение скорости, аутентификация и сложная маршрутизация трафика. Все плюсы и минусы краевого прокси-сервера такие же, как и прокси по середине. Клиентам обычно требуется доступ к системе через DNS с использованием произвольных сетевых библиотек, которые владелец сервиса не контролирует (создание встроенной клиентской библиотеки или топологий прокси-сервера sidecar, описанных в следующих разделах, нецелесообразно запускать непосредственно на клиенте). Кроме того, по соображениям безопасности желательно иметь один шлюз, через который весь интернет-трафик проходит через систему.

Встроенная клиентская библиотека

Балансировка нагрузки через встроенную клиентскую библиотеку

Чтобы избежать присущих топологии сервера прокси по середине проблем отказа и масштабирования, более сложные инфраструктуры переключились на то, чтобы встроить балансировщик нагрузки непосредственно в службы через библиотеку, как показано на рисунке. Библиотеки сильно различаются по поддерживаемым функциям, но некоторые из наиболее известных и многофункциональных в этой категории — Finagle, [Eureka/Ribbon/Hystrix](https: //netflix.github.io/) и gRPC (на основе внутренней системы Google, называемой Stubby). Основным достоинством решения на базе библиотеки является то, что он полностью распределяет всю функциональность балансировщика нагрузки каждому клиенту, тем самым устраняя описанные ранее проблемы отказа и масштабирования. Первичным решением библиотечного решения является то, что библиотека должна быть реализована на всех языках, которые использует организация. Распределенные архитектуры становятся все более «полиглотными» (многоязычными). В этой среде стоимость переоснащения чрезвычайно сложной сетевой библиотеки на разных языках может стать непомерно высокой. Наконец, развертывание обновлений библиотек в большой архитектуре обслуживания может быть чрезвычайно болезненным.

Учитывая сказанное, упомянутые выше библиотеки были успешными для компаний, которые смогли ограничить распространение разных языков программирования и библиотек, тем самым преодолев проблемы обновления.

Прокси-сервер Sidecar

Балансировка нагрузки через прокси-сервер sidecar

Вариантом топологии балансировки нагрузки встроенной клиентской библиотеки является топология прокси-сервера sidecar, показанная на рисунке. В последние годы она была популяризирована как «служебная сетка». Идея прокси-сервера sidecar заключается в том, что ценой незначительного штрафа за латентность переходом на другой процесс все преимущества встроенного библиотечного подхода могут быть получены без программирования. Наиболее популярными балансировщиками нагрузки прокси-сервера sidecar на данный момент являются Envoy, NGINX, [HAProxy](https: //www.haproxy.com/) и Linkerd. Для более детального изучения подхода прокси-сервера sidecar см. мой пост в блоге о Envoy, а также мой пост [the service mesh data plane vs. control plane] (https://medium.com/@mattklein123/service-mesh-data-plane-vs-control-plane-2774e720f7fc).

Резюме и плюсы/минусы различных топологий балансировки нагрузки

Топология прокси по середине — это, как правило, самая легкая топология балансировки нагрузки. Топология не соответствует одной точке отказа, ограничениям масштабирования и работе черного ящика.

Топология краевого прокси-сервера похожа на сервер прокси по середине.

Топология встроенной клиентской библиотеки обеспечивает лучшую производительность и масштабируемость, но обладает минусом — необходимостью реализовать библиотеку на разных языках и обновлять библиотеки для всех служб.

Топология прокси-сервера sidecar не работает так же, как и встроенная топология клиентской библиотеки, но не страдает от каких-либо ограничений.

Думаю, топология прокси-сервера sidecar (служебная сеть) постепенно заменяет все другие топологии для связи между сервисами. Точка топологии краевого прокси-сервера всегда будет необходима до того, как трафик войдет в служебную сетку.

Современное состояние в балансировке нагрузки L4

Балансировщики нагрузки L4 по-прежнему актуальны?

В этом сообщении уже обсуждалось, насколько значимы балансировщики L7 для современных протоколов. Теперь более подробно перейдем к функциям балансировки нагрузки L7. Означает ли это, что балансировщики нагрузки L4 больше не актуальны? Нет! Хотя, на мой взгляд, балансировщики нагрузки L7 в конечном счете полностью заменяют балансировщики нагрузки L4 для связи между сервисами, балансировщики нагрузки L4 по-прежнему крайне важны, потому что почти все современные крупные распределенные архитектуры используют двухуровневую архитектуру балансировки нагрузки L4 / L7 для интернет-трафика. Преимущества размещения выделенных балансировщиков нагрузки L4 до балансировщика нагрузки L7 при развертывании на краю:

Поскольку балансировочные устройства L7 выполняют значительно более сложный анализ, преобразование и маршрутизацию трафика приложения, они могут обрабатывать меньшую часть нагрузки (измеренной в пакетах в секунду и байтах в секунду), чем оптимизированный балансировщик нагрузки L4. Этот факт, как правило, делает балансировщики нагрузки L4 лучшим местом для обработки определенных типов DoS-атак (например, SYN-флуда, общих атак на передачу пакетов и т.д.).

Балансировщики нагрузки L7, как правило, более активно развиваются, развертываются чаще и имеют больше багов, чем балансировщики нагрузки L4. Наличие спереди балансировщика нагрузки L4, который может выполнять проверку работоспособности и слив во время развертывания балансировщика нагрузки L7, значительно проще, чем применение механизмов развертывания с современными балансирами нагрузки L4, которые обычно используют BGP и ECMP (подробнее об этом ниже). И, наконец, поскольку балансировщики L7 с большей вероятностью имеют ошибки из-за сложности их функциональности, наличие балансировщика нагрузки L4, который может маршрутизировать ошибки и аномалии, приводит к более стабильной общей системе.

В следующих разделах я опишу несколько различных конструкций для балансировщика нагрузки L4 для прокси по середине и краевого прокси. Следующие проекты, как правило, неприменимы к клиентской библиотеке и топологиям прокси-сервера sidecar.

TCP/UDP балансировщик нагрузки

L4 балансировщик нагрузки

Первым типом балансировщика нагрузки L4, который все еще используется, является балансировка нагрузки на выключение, показанная на рисунке. Это тот же самый балансировщик нагрузки, который мы видели в введении к балансировке нагрузки L4 выше. В этом типе используются два дискретных TCP-соединения: один между клиентом и балансировщиком нагрузки и один между балансировщиком нагрузки и бэкендом.

Балансировщики нагрузки L4 по-прежнему используются по двум причинам:

Их относительно просто реализовать.

Прекращение соединения в непосредственной близости (низкая латентность) к клиенту имеет существенные последствия для производительности. В частности, если конечный балансировщик нагрузки можно разместить рядом с клиентами, которые используют сеть с потерями (например, сотовую), повторные передачи, скорее всего, будут происходить быстрее до того, как данные будут перемещены в надежный транзит по маршруту до конечного местоположения. Другими словами, этот тип балансировщика нагрузки может использоваться в сценарии Point of Presence (POP) для прерывания TCP-соединения.

Отказоустойчивость через пары высокой доступности

L4-отказоустойчивость через пары HA и отслеживание соединений

До сих пор мы рассматривали дизайн балансировщика нагрузки L4 изолированно. Для балансировки нагрузки Passthrough и DSR требуется некоторое отслеживание соединения и состояния в самом балансировщике нагрузки. Что делать, если балансировщик нагрузки умирает? Если один экземпляр балансировщика умирает, все проходящие через него соединения будут разорваны. В зависимости от приложения это может существенно повлиять на производительность.

Исторически балансировщики нагрузки L4 были аппаратными устройствами, приобретенными у типичных поставщиков (Cisco, Juniper, F5 и т.д.). Эти устройства чрезвычайно дороги и обрабатывают большой объем трафика. Чтобы избежать отказа одного балансировщика нагрузки, разрушающего все соединения и приводящего к существенному отключению приложений, балансировщики обычно развертывались в парах с высокой доступностью, как показано на рисунке. Типичная установка балансировки нагрузки HA имеет следующий дизайн:

Пара пограничных маршрутизаторов HA обслуживает некоторое количество виртуальных IP-адресов (VIP). Эти пограничные маршрутизаторы объявляют VIP-клиентов, используя протокол пограничных шлюзов (BGP). Основной пограничный маршрутизатор имеет более высокий вес BGP, чем резервный, поэтому в рабочем состоянии он обслуживает весь трафик. (BGP — чрезвычайно сложный протокол, для целей этой статьи просто представляйте BGP как механизм, с помощью которого сетевые устройства сообщают, что они доступны для приема трафика с других сетевых устройств, и учитывайте, что каждая ссылка может иметь вес, который определяет приоритетность трафика).

Аналогично, основной балансировщик нагрузки L4 объявляет себя граничным маршрутизатором с более высоким значением BGP, чем резервный, поэтому в рабочем состоянии он обслуживает весь трафик.

Первичный балансировщик нагрузки перекрестно подключен к резерву и разделяет состояние отслеживания соединений. Таким образом, если первичный балансировщик умирает, резервный может взять на себя обработку всех активных подключений.

Два пограничных маршрутизатора и два балансировщика нагрузки связаны друг с другом. Это означает, что если один из пограничных маршрутизаторов или один из балансировщиков нагрузки умирает или имеет собственные объявления маршрутов BGP по какой-либо другой причине, резервный может взять на себя обслуживание всего трафика.

Вышеприведенная настройка показывает, насколько много трафика по-прежнему обслуживается. Однако у такого подхода есть существенные недостатки:

VIP должны быть правильно распределены по парам балансировки нагрузки HA с учетом использования мощности.

Использование ресурсов системы невелико. 50% мощности находится в режиме ожидания. Учитывая, что аппаратные балансировщики нагрузки чрезвычайно дороги, это приводит к существенному количеству «простаивающих» денег.

Современному дизайну распределенной системы нужна большая отказоустойчивость, чем могут обеспечить активный/резервный балансировщики. Например, система должна выдерживать множественные одновременные сбои и продолжать работать. Пара балансировщиков нагрузки HA подвержена общему сбою, если одновременно активизируются как активный, так и резервный балансировщик нагрузки.

Собственные крупные аппаратные устройства от поставщиков чрезвычайно дороги и приводят к привязке к поставщику. Как правило, желательно заменить эти аппаратные устройства горизонтально масштабируемыми программными решениями, построенными с использованием обычных серверов.

Отказоустойчивость и масштабирование с помощью кластеров с распределенным консистентным хешированием

L4-отказоустойчивость и масштабирование с помощью кластерных балансировщиков нагрузки и консистентного хеширования

В предыдущем разделе была представлена ошибка отказоустойчивости балансировки нагрузки L4 через пары HA, а также указаны присущие этой конструкции проблемы. Начиная с середины 2000-х годов крупные интернет-инфраструктуры начали разрабатывать и развертывать новые массивно-параллельные системы балансировки нагрузки L4, как показано на рисунке. Целями этих систем являются:

Снижение всех недостатков конструкции пары HA, описанных в предыдущем разделе.

Уход от проприетарных балансировщиков оборудования в сторону программных решений, построенных с использованием обычных серверов и сетевых адаптеров.

Этот дизайн балансировки нагрузки L4 чаще всего называют отказоустойчивостью и масштабированием посредством кластеризации и распределенного согласованного хеширования. Он работает следующим образом:

N пограничных маршрутизаторов объявляют все VIP Anycast с одинаковым весом BGP. Равнозначный multi-path-роутинг (ECMP) используется для обеспечения того, чтобы в общем случае все пакеты из одного потока поступали на один и тот же пограничный маршрутизатор. Поток, как правило, представляет собой кортеж источника IP/порта и IP-адреса назначения. (Короче говоря, ECMP — это способ распространения пакетов по набору одинаково взвешенных сетевых ссылок с использованием консистентного хеширования). Хотя сами пограничные маршрутизаторы не особо заботятся о том, какие пакеты поступают в поток, предпочтительно, чтобы все пакеты из потока пересекали один и тот же набор ссылок, чтобы избежать сбоев пакетов, которые ухудшают производительность.

N машин для балансировки нагрузки L4 объявляют все VIP с одинаковым весом BGP к пограничным маршрутизаторам. Опять же, используя ECMP, пограничные маршрутизаторы обычно выбирают одну и ту же машину балансировки нагрузки для потока.

Каждая машина балансировки нагрузки L4 обычно выполняет частичное отслеживание соединения, а затем использует консистентное хеширование, чтобы выбрать бэкенд для потока. GRE используется для инкапсуляции пакетов, отправленных из балансировщика нагрузки в бэкенд.

Затем DSR используется для отправки пакетов непосредственно из бэкенда клиенту через пограничные маршрутизаторы.

Консистентный алгоритм хеширования, используемый балансировщиком нагрузки L4, является областью активных исследований. Существуют компромиссы в первую очередь вокруг выравнивания нагрузки, минимизации задержки, минимизации сбоев во время изменений бэкенда и минимизации издержек памяти. Полное обсуждение этой темы выходит за рамки этой статьи.

Давайте посмотрим, как приведенный выше пример смягчает все недостатки подхода пары HA:

По мере необходимости могут добавляться новые пограничные маршрутизаторы и устройства балансировки нагрузки. Консистентное хеширование используется на каждом уровне для максимального уменьшения количества затронутых потоков при добавлении новых машин.

Использование ресурсов системы может выполняться на как можно более высоком уровне при сохранении достаточной отказоустойчивости.

Как пограничные маршрутизаторы, так и балансировщики нагрузки теперь могут быть построены с использованием обычного оборудования с небольшой долей стоимости традиционных аппаратных балансировщиков (подробнее об этом ниже).

Вопросы, которые обычно задают об этой конструкции: — «почему пограничные маршрутизаторы не разговаривают напрямую с серверами через ECMP?», «зачем нам нужен балансировщик нагрузки?». Причины в первую очередь связаны с уменьшением эффективности DoS и операционной системой. Без балансировки нагрузки каждый бэкенд должен был бы участвовать в BGP и потребовалось бы значительно большее время для развертывания.

Все современные системы балансировки нагрузки L4 двигаются к этой конструкции (или к ее варианту). Двумя наиболее известными примерами являются Maglev из Google и [Балансировщик сетевой нагрузки (NLB)](http: //docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html) от Amazon. В настоящее время нет балансировщика нагрузки OSS, который реализует этот проект, однако есть компания, которая планирует выпуск OSS в 2018 году. Я очень взволнован этим выпуском, поскольку современный балансировщик нагрузки L4 является решающим элементом отсутствия OSS в сетевом пространстве.

Текущее состояние балансировки нагрузки L7

The proxy wars in tech currently is \*quite literally\* the proxy wars.

Or the “war of the proxies”.

Nginx plus, HAProxy, linkerd, Envoy all quite literally killing it.

And proxy-as-a-service/routing-as-a-service SaaS vendors raising the bar as well. Very interesting times!

— Cindy Sridharan (@copyconstruct) November 28, 2017

Да, в самом деле. За последние несколько лет наблюдается возрождение в балансировщиках нагрузки/прокси L7. Это очень хорошо сочетается с продолжающимся стремлением к микросервисной архитектуре в распределенных системах. Кроме того, рост автомасштабирования, планировщики контейнеров и т.д. означает, что дни предоставления статических IP-адресов в статических файлах давно прошли. Системы не только используют сеть активнее, они становятся значительно более динамичными, требуя новых функциональных возможностей в балансировщиках нагрузки.

Поддержка протокола

Современные балансировщики нагрузки L7 добавляют явную поддержку для многих разных протоколов. Чем большее отношение балансировщик нагрузки имеет к трафику приложения, тем более сложные вещи он может делать в отношении вывода наблюдаемости, расширенной балансировки нагрузки, маршрутизации и т.д. Например, на момент написания этой статьи Envoy явно поддерживает парсинг и маршрутизацию протокола L7 для HTTP/1, HTTP/2, gRPC, Redis, MongoDB и DynamoDB. В будущем, вероятно, появятся новые протоколы, включая MySQL и Kafka.

Динамическая конфигурация

Как описано выше, все более динамичный характер распределенных систем требует параллельных инвестиций в создание динамических и реактивных систем управления. Istio — один из примеров такой системы. Для получения дополнительной информации по этой теме см. мой пост service mesh data plane vs. control plane.

Расширенная балансировка нагрузки

Балансировщики L7 обычно имеют встроенную поддержку расширенных функций балансировки нагрузки, таких как тайм-ауты, повторные попытки, ограничение скорости, прерывание цепи, теневое копирование, буферизация, маршрутизация на основе контента и т.д.

Наблюдаемость

Как описано выше в разделе об общих характеристиках балансировки нагрузки, все более динамичные системы становится все труднее отлаживать. Надежная выходная характеристика протокола — это, пожалуй, самая важная особенность, обеспечиваемая современными балансировщиками L7. Вывод числовой статистики, распределенных трассировок и настраиваемого ведения журнала теперь практически необходим для любого решения балансировки нагрузки L7.

Расширяемость

Пользователи современных балансировщиков L7 часто хотят легко расширять их, чтобы добавить пользовательские функции. Это можно сделать путем записи подключаемых фильтров, которые загружаются в балансировщик нагрузки. Многие балансировки нагрузки также поддерживают сценарии, как правило, через Lua.

Отказоустойчивость

Немного выше я написал о L4-отказоустойчивости балансировки нагрузки. А что насчет отказоустойчивости балансировки нагрузки L7? Мы рассматриваем балансировщики нагрузки L7 как расходный материал и stateless. Использование программного обеспечения для балансировки позволяет L7 легко масштабироваться горизонтально. Кроме того, обработка и отслеживание состояния, которые выполняют балансировщики нагрузки L7, существенно сложнее, чем L4. Попытка построить HA-пару балансировщика нагрузки L7 технически возможна, но это будет тяжело реализовать.

Как в доменах балансировки нагрузки L4 и L7, отрасль в целом отходит от HA-пар к горизонтально масштабируемым системам, которые взаимодействуют через консистентное хеширование.

И еще

Балансировщики нагрузки L7 развиваются ошеломляющими темпами. Пример того, что предлагает Envoy, см. в разделе Обзор архитектуры.

Глобальная балансировка нагрузки и централизованная плоскость управления

Глобальная балансировка нагрузки

Будущее балансировки нагрузки будет все чаще рассматривать отдельные балансировщики как товар. На мой взгляд, настоящие инновационные и коммерческие возможности лежат в плоскости управления. Рисунок показывает пример глобальной системы балансировки нагрузки. В этом примере происходит несколько разных вещей:

Каждый прокси-сервер sidecar связывается с серверами в трех разных зонах (A, B и C).

Как показано, 90% трафика отправляется в зону C, а 5% трафика отправляется в обе зоны A и B.

Прокси-сервер sidecar и внутренние серверы сообщают о периодическом состоянии глобального балансировщика нагрузки. Это позволяет ему принимать решения, учитывающие задержки, затраты, нагрузку, текущие сбои и т.д.

Глобальный балансировщик нагрузки периодически настраивает каждый прокси-сервер sidecar с текущей информацией о маршрутизации.

Глобальный балансировщик нагрузки будет в состоянии делать сложные вещи, которые ни один балансировщик нагрузки не может сделать самостоятельно. Например:

Автоматическое обнаружение и маршрутизация вокруг зонального отказа.

Применение глобальных политик безопасности и маршрутизации.

Обнаружение и уменьшение аномалий трафика, включая атаки DDoS, с использованием машинного обучения и нейронных сетей.

Обеспечение централизованного пользовательского интерфейса и визуализации, которые позволяют инженерам понимать всю распределенную систему в совокупности и управлять ей.

Чтобы сделать глобальную балансировку нагрузки возможной, балансировщик нагрузки, используемый в качестве data plane, должен обладать сложными возможностями динамической конфигурации. Для получения дополнительной информации по этой теме, пожалуйста, см. мой пост Envoy’s universal data plane API, а также service mesh data plane vs. control plane..

Эволюция от аппаратного до программного обеспечения

Заключение и будущее балансировки нагрузки

Подводя итог, приведем ключевые выводы

Балансировщики нагрузки являются ключевым компонентом в современных распределенных системах.

Существуют два общих класса балансиров нагрузки: L4 и L7.

Оба балансировщика нагрузки L4 и L7 актуальны в современных архитектурах.

L4-балансировщики двигаются в направлении горизонтально масштабируемых распределенных консистентных решений хеширования.

В связи с распространением динамических архитектур микросервиса сейчас активно инвестируются балансировщики L7.

Глобальная балансировка нагрузки и разделение между control и data plane — это будущее балансировки нагрузки с инновационными и коммерческими возможностями.

Индустрия агрессивно продвигается к аппаратным и программным продуктам OSS для сетевых решений. Я считаю, что традиционные поставщики балансировщиков нагрузки, такие как F5, будут сначала заменены программным обеспечением OSS и облачными поставщиками.