Сетевые компоненты для распределенных систем

Современные веб-приложения зависят от критически важных сетевых компонентов, которые обеспечивают масштабируемость, отказоустойчивость и глобальную производительность. В этом материале рассматриваются основные компоненты архитектуры распределенных систем: балансировщики нагрузки, шлюзы API, прокси, CDN и компоненты вспомогательной инфраструктуры.

Балансировщики нагрузки

Балансировщики нагрузки распределяют входящий трафик между несколькими бэкэнд-серверами, обеспечивая горизонтальную масштабируемость, отказоустойчивость и сокращение времени отклика. Они предоставляют единую точку входа для клиентов и автоматически перенаправляют трафик с неработающих серверов.

Балансировка нагрузки эволюционировала от дорогостоящего аппаратного оборудования (F5, Citrix) к гибким программным решениям (HAProxy, NGINX) и современным облачным управляемым сервисам (AWS ALB, Google Cloud Load Balancing, контроллеры входящего трафика Kubernetes).

Типы балансировщиков нагрузки

Выбор подходящего типа балансировщика нагрузки зависит от ваших конкретных требований и того, где в сетевом стеке необходимо распределять трафик.

DNS-балансировщики нагрузки

DNS-балансировщики нагрузки, такие как AWS Route 53, работают на уровне разрешения доменов, возвращая разным пользователям разные IP-адреса серверов. Эти решения отлично подходят для географического распределения и простых сценариев отработки отказа, хотя кэширование DNS может задерживать обновления во время сбоев сервера. Основные случаи использования включают направление пользователей в оптимальные центры обработки данных.

Маршрутизаторы ECMP

Маршрутизаторы ECMP от таких поставщиков, как Juniper и Cisco, работают на уровне сети, распределяя трафик по нескольким путям с одинаковой стоимостью. Эти решения обеспечивают оптимальную производительность в средах центров обработки данных, где требуется распределение трафика по нескольким сетевым каналам без создания единичных точек отказа.

Балансировщики сетевой нагрузки (уровень 4)

Балансировщики сетевой нагрузки (уровень 4), такие как AWS NLB, сосредоточены исключительно на IP-адресах и портах, обеспечивая исключительную производительность с ультранизкой задержкой, превышающей миллионы запросов в секунду. Эти решения оптимальны для сценариев, требующих высокой производительности без сложной логики маршрутизации, таких как платформы высокочастотной торговли или бэкэнды для игр в реальном времени.

Балансировщики приложений (уровень 7)

Балансировщики приложений (уровень 7), такие как AWS ALB, проверяют содержимое HTTP-запросов, чтобы принимать интеллектуальные решения о маршрутизации, распределяя трафик по нескольким экземплярам приложений. Проверка содержимого HTTP позволяет использовать расширенные функции, включая привязку сеансов с помощью файлов cookie. Когда пользователи впервые получают доступ к приложениям, ALB могут вставлять файлы cookie, обеспечивая попадание последующих запросов на тот же бэкэнд-сервер. Эта функция необходима для приложений, которые хранят данные сеансов локально, а не в общих хранилищах сеансов. Хотя они требуют большей вычислительной мощности, чем балансировщики нагрузки уровня 4, они предоставляют необходимые функции для сохранения сеансов и маршрутизации с учетом содержимого.

Алгоритмы балансировки нагрузки

Алгоритмы распределения трафика балансировщика нагрузки значительно влияют на производительность приложений и пользовательский опыт.

Статические алгоритмы

Round Robin последовательно циклически переключается между серверами. **Weighted Round Robin** позволяет назначать веса серверам. **Алгоритмы на основе хеширования** обеспечивают согласованное сопоставление пользователей и серверов.

Динамические алгоритмы

Least Connections направляет запросы на серверы с наименьшим количеством активных соединений. **Least Response Time** учитывает скорость отклика сервера. **Least Loaded** анализирует использование ресурсов сервера.

Sticky sessions

Связь на основе файлов cookie работает с балансировщиками Layer 7. **Связь по исходному IP-адресу** маршрутизирует на основе IP-адресов пользователей и поддерживает балансировщики Layer 4 и 7.

Развертывание в производственной среде

Для развертывания балансировщиков нагрузки в производственной среде требуется несколько важных функций, обеспечивающих надежность и безопасность.

Проверки работоспособности и SSL-терминация

Проверки работоспособности

Проверки работоспособности обеспечивают раннее обнаружение сбоев посредством непрерывных HTTP-запросов или TCP-соединений для проверки отзывчивости сервера. Когда серверы не проходят проверки работоспособности, трафик автоматически перенаправляется на работоспособные серверы, как правило, в течение нескольких секунд после обнаружения проблемы.

Прерывание SSL

Прерывание SSL значительно улучшает производительность и управление за счет централизации обработки шифрования и дешифрования на уровне балансировщика нагрузки. Этот подход централизует управление сертификатами, устраняя необходимость обновления сертификатов на множестве серверов и позволяя серверам приложений сосредоточиться на бизнес-логике.

Защита и кэширование

Защита от DDoS-атак

Защита от DDoS-атак и ограничение скорости защищают бэкэнд-сервисы от злонамеренных атак и неожиданных всплесков трафика. Многие облачные балансировщики нагрузки интегрируются с межсетевыми экранами веб-приложений (WAF), чтобы обеспечить комплексную защиту без дополнительных сложностей настройки.

Возможности кэширования

Современные возможности кэширования балансировщиков нагрузки значительно снижают нагрузку на бэкэнд за счет хранения часто запрашиваемых ответов. Хотя кэширование балансировщиков нагрузки менее сложно, чем выделенные CDN, оно эффективно обрабатывает ответы API и динамический контент с низкой частотой изменений.

Облачные реализации

Облачные провайдеры упростили балансировку нагрузки за счет управляемых сервисов, которые обрабатывают операционные сложности.

AWS

AWS предоставляет три основных варианта: Application Load Balancer (ALB) для трафика HTTP/HTTPS со всеми ожидаемыми функциями уровня 7, Network Load Balancer (NLB) для высокопроизводительного трафика TCP/UDP и Classic Load Balancer для устаревших приложений (хотя вам, вероятно, следует отказаться от этого варианта).

Kubernetes

Kubernetes использует другой подход: службы обеспечивают внутреннюю балансировку нагрузки внутри кластера, а контроллеры Ingress обрабатывают внешний трафик с помощью расширенных функций маршрутизации. Преимущество Kubernetes заключается в том, что он может автоматически предоставлять балансировщики нагрузки облачного поставщика по мере необходимости, упрощая большую часть сложности с помощью простых конфигураций YAML.

Шлюзы API

В то время как балансировщики нагрузки отлично справляются с распределением запросов между идентичными серверами, шлюзы API решают проблему управления сложностью API в распределенных архитектурах. Хотя шлюзы API обычно ассоциируются с микросервисами, они также полезны в сервис-ориентированных архитектурах (SOA), гибридных облачных развертываниях и монолитных приложениях, предоставляющих несколько конечных точек API. Шлюзы API функционируют как централизованные системы маршрутизации запросов, которые направляют запросы к соответствующим службам.

В отличие от балансировщиков нагрузки, которые распределяют запросы между идентичными экземплярами служб, шлюзы API принимают интеллектуальные решения о маршрутизации на основе характеристик запросов. Запросы к /users/profile направляются к службам пользователей на определенных серверах, а запросы к /orders/history — к службам заказов на других серверах.

Что делает шлюзы API особенными

Шлюзы API превосходны в областях, выходящих за рамки распределения трафика. Они централизуют вопросы безопасности, включая потоки OAuth, проверку токенов JWT, управление ключами API и контроль доступа на основе ролей, устраняя необходимость в индивидуальной реализации сервисов. Это позволяет реализовать единую аутентификацию на уровне шлюза, а не дублировать ее во многих бэкэнд-сервисах.

Ограничение скорости и регулирование защищают бэкэнд-сервисы от злонамеренного использования и непреднамеренной перегрузки системы. Различные уровни API могут иметь разные ограничения — например, пользователи бесплатного уровня могут получать 300 запросов за 15 минут, а корпоративные клиенты — более высокие ограничения в зависимости от уровня подписки.

Преобразование и мониторинг

Преобразование запросов

Преобразование запросов и ответов демонстрирует значительную ценность шлюза API благодаря переводу протоколов (REST в GraphQL), преобразованию форматов данных и версионированию API без изменения кода бэкэнда. Эти возможности необходимы для обеспечения обратной совместимости и интеграции устаревших систем.

Мониторинг и аналитика

Мониторинг и аналитика предоставляют информацию о фактических моделях использования API, включая популярные конечные точки, частоту ошибок сервисов и узкие места в производительности, прежде чем они повлияют на пользователей.

Кэширование

Кэширование на уровне шлюза снижает нагрузку на бэкэнд-сервисы и сокращает время отклика для часто запрашиваемых данных.

Выбор правильного решения

Развитие продуктов стирает границы между API-шлюзами и балансировщиками нагрузки, поскольку решения включают в себя функции обеих категорий.

Гибридные решения

Гибридные решения, включая Kong, Ambassador и Istio Gateway, сочетают в себе расширенные функции управления API с высокопроизводительной балансировкой нагрузки. Эти решения предоставляют унифицированные инструменты для решения обеих задач, но могут потребовать более сложной настройки и эксплуатации.

Чистые шлюзы API

Чистые шлюзы API, такие как AWS API Gateway, Google Cloud Endpoints и Azure API Management, ориентированы на функции управления API. Эти полностью управляемые сервисы хорошо интегрируются с соответствующими облачными экосистемами, но могут иметь ограничения по производительности в сценариях с высокой пропускной способностью.

Чистые балансировщики нагрузки

Чистые балансировщики нагрузки, включая AWS NLB, базовый HAProxy и F5 BIG-IP, отличаются высокой производительностью распределения трафика, но требуют дополнительных инструментов для функций управления API.

Интеграция облачных технологий и Kubernetes

AWS API Gateway

AWS API Gateway предоставляет полностью управляемый хостинг API с интеграцией Lambda, поддерживая как REST, так и HTTP API (HTTP API обеспечивают более высокую производительность и более низкие затраты для простых случаев использования).

Kubernetes

В средах Kubernetes для базовой маршрутизации обычно используются NGINX Ingress Controllers и Traefik, а решения для service mesh, такие как Istio, предоставляют расширенные функции шлюза. Облачные решения, такие как Kong для Kubernetes, предоставляют функции управления API для предприятий непосредственно в кластерах.

Прокси

Прокси являются важными компонентами сетевой инфраструктуры, расположенными между клиентами и серверами для выполнения задач кэширования, безопасности и оптимизации, которые в противном случае создавали бы нагрузку на ваши приложения.

Прямые прокси: действуют от имени клиентов

Прямые прокси перехватывают запросы клиентов до того, как они достигнут места назначения. С точки зрения сервера, все запросы кажутся исходящими от прокси, а не от отдельных пользователей.

Прямые прокси широко используются в корпоративных средах для фильтрации доступа в Интернет, экономии пропускной способности за счет кэширования и анонимности пользователей. Squid является стандартным выбором для корпоративных развертываний, а разработчики используют Charles Proxy или Burp Suite для отладки API. Облачные решения, такие как Zscaler, стали популярными для распределенных рабочих групп.

Прямые прокси-серверы позволяют централизовать политики безопасности и оптимизировать пропускную способность, но могут вызывать задержки и создавать единичные точки отказа без надлежащей конструкции высокой доступности.

Обратные прокси-серверы: действуют от имени серверов

Обратные прокси работают в обратном направлении, перехватывая входящие запросы до того, как они достигнут бэкэнд-приложений. Клиенты, как правило, не подозревают, что общаются с прокси, а не с реальными серверами.

Обратные прокси отлично справляются с SSL-терминацией (обработкой всего шифрования/дешифрования), кэшированием статического контента для сокращения времени отклика, маршрутизацией запросов на соответствующие бэкэнд-серверы и сжатием ответов для уменьшения использования пропускной способности.

С точки зрения безопасности обратные прокси имеют неоценимое значение. Они скрывают детали вашей бэкэнд-инфраструктуры, обеспечивают защиту от DDoS-атак за счет ограничения скорости, могут включать функции веб-прикладного брандмауэра и централизовать регистрацию всего входящего трафика.

NGINX и Apache HTTP Server — популярные решения для самостоятельного хостинга, а Cloudflare предоставляет глобально распределенный сервис обратных прокси. Стоит отметить, что грань между обратными прокси-серверами и балансировщиками нагрузки становится все более размытой — многие современные обратные прокси-серверы, такие как NGINX Plus и HAProxy, включают в себя сложные функции балансировки нагрузки, а балансировщики нагрузки часто включают в себя функции обратного прокси-сервера, такие как SSL-терминация и кэширование. Ключ к успеху — тщательная настройка: плохо настроенный обратный прокси-сервер может легко стать узким местом, ограничивающим производительность всего приложения.

Сети доставки контента (CDN)

Доставка изображений высокого разрешения пользователям в Токио с серверов в Нью-Йорке требует прокладки тысяч километров оптоволоконного кабеля, что добавляет сотни миллисекунд задержки. CDN решают эту проблему, распределяя контент по географически разнесенным серверам, располагая его ближе к пользователям по всему миру.

Как CDN меняют пользовательский опыт

Когда пользователи в Токио запрашивают изображения, системы маршрутизации CDN направляют их на японские пограничные серверы, которые либо имеют кэшированный контент, либо могут быстро получить его с региональных серверов. Географическая близость обычно сокращает задержку на 50–80 %, что значительно выгодно для приложений с большим объемом мультимедиа и сайтов электронной коммерции, где улучшение на миллисекунды влияет на коэффициент конверсии. Статические ресурсы, включая изображения, CSS-файлы и пакеты JavaScript, кэшируются на длительный период из-за редких изменений. Динамический контент может кэшироваться на короткий срок или не кэшироваться вовсе, хотя современные CDN все чаще поддерживают интеллектуальное персонализированное кэширование контента на пограничных узлах.

Более чем простое кэширование

Современные CDN вышли за рамки простого кэширования контента и теперь включают возможности пограничных вычислений, которые позволяют выполнять бессерверные функции на пограничных узлах, обрабатывая запросы без обратных запросов к исходному серверу. AWS CloudFront Functions, Cloudflare Workers и Fastly Compute@Edge являются примерами этой тенденции, поддерживая персонализацию контента и простую обработку запросов API в пограничных точках.

Функции безопасности приобрели не меньшее значение в современных реализациях CDN. Современные CDN включают защиту веб-приложений, защиту от DDoS-атак, обнаружение и управление ботами, а также оптимизацию SSL/TLS с использованием современных протоколов безопасности. Многие атаки блокируются в пограничных точках, не доходя до исходных серверов.

Аналитика в реальном времени предоставляет подробную информацию о поведении пользователей, показателях производительности и угрозах безопасности во всех пограничных точках, включая скорость загрузки контента, географическое распределение пользователей и популярные модели доступа к контенту.

Выбор и внедрение CDN

Крупные поставщики, включая Cloudflare, AWS CloudFront, Google Cloud CDN, Fastly и Akamai, предлагают свои уникальные преимущества. Глобальные гипермасштабируемые провайдеры предлагают обширные сети с интегрированными облачными сервисами, а специализированные решения могут обеспечить превосходные обновления контента в реальном времени или удобные для разработчиков API.

Для успешного внедрения необходимо уделить особое внимание стратегиям очистки кэша для обновлений глобально кэшированного контента, настройке исходного сервера для эффективной обработки промахов кэша, комплексному мониторингу для отслеживания улучшений производительности и управлению затратами в связи с потенциальным резким увеличением использования CDN при росте трафика.

Дополнительные критически важные компоненты сети

Современные распределенные системы полагаются на специализированные компоненты сетевой инфраструктуры, выходящие за рамки основных элементов, для решения конкретных архитектурных задач.

Service Mesh: уровень коммуникации микросервисов

По мере того как организации разбивают монолитные приложения на множество микросервисов, управление коммуникацией между сервисами становится все более сложным. Service mesh, включая Istio, Linkerd и Consul Connect, решают эту проблему с помощью специальных инфраструктурных уровней для коммуникации между сервисами.

Реализация осуществляется с помощью прокси-серверов sidecar — небольших прокси-серверов, развернутых рядом с каждым сервисом для обработки всего сетевого трафика. Этот подход предоставляет такие функции, как взаимное TLS-шифрование, управление трафиком, балансировка нагрузки и подробная наблюдаемость без изменения кода приложения. Сервисы выполняют стандартные HTTP-вызовы, а service mesh обрабатывает всю сложность на нижнем уровне.

Брандмауэры веб-приложений: ваша первая линия защиты

Брандмауэры веб-приложений работают на уровне приложений, фильтруя и отслеживая HTTP-трафик на основе заранее определенных правил. Они обеспечивают защиту от распространенных атак, включая SQL-инъекции, межсайтовый скриптинг (XSS) и 10 наиболее уязвимых уязвимостей по версии OWASP.

Современные WAF, включая AWS WAF, Cloudflare WAF и F5 Advanced WAF, интегрируются с CDN, балансировщиками нагрузки и шлюзами API, обеспечивая комплексную защиту без задержек. Многие решения изучают шаблоны трафика, чтобы автоматически блокировать подозрительные запросы до того, как они достигнут приложений.

DNS и обнаружение служб: поиск служб в динамических средах

Традиционный DNS хорошо работает для статической инфраструктуры, но современные контейнерные среды требуют динамических решений. Системы обнаружения сервисов, включая Consul, etcd и Kubernetes DNS, позволяют автоматически находить сервисы и обмениваться данными по мере их масштабирования и миграции по инфраструктуре.

Современные DNS-решения вышли за рамки разрешения доменов и теперь обеспечивают интеллектуальную маршрутизацию трафика, отработку неисправностей на основе состояния и географическую балансировку нагрузки. Эти решения интегрируются с обнаружением сервисов для создания отказоустойчивых сетевых архитектур с самовосстановлением.

Системная интеграция

Сетевые компоненты работают как взаимосвязанные экосистемы, которые преобразуют сложные распределенные архитектуры в бесшовный пользовательский опыт, а не функционируют изолированно.

Запросы пользователей могут сначала попадать на пограничные серверы CDN, которые направляют трафик через WAF для проверки безопасности, а затем на балансировщики нагрузки, распределяющие трафик по шлюзам API. Эти шлюзы могут использовать инфраструктуру service mesh для связи бэкэнд-микросервисов, в то время как обратные прокси обрабатывают SSL-терминацию и кэширование на протяжении всего процесса.

Эффективный дизайн системы требует понимания как отдельных компонентов, так и взаимодополняющих отношений между ними. Хорошо спроектированные системы используют балансировщики нагрузки для масштабирования, шлюзы API для оркестрации микросервисов, прокси для обеспечения безопасности и оптимизации производительности, CDN для глобального охвата, а также специализированные компоненты, такие как service mesh и WAF, для обеспечения отказоустойчивости и защиты.

При проектировании распределенных систем эти компоненты следует рассматривать как специализированные инструменты, выбирая подходящие решения для конкретных задач и сохраняя гибкость для развития архитектуры по мере роста требований.