**Запрос на Хеджирование**

Объясняет, что такое хеджирование запросов и как вы можете его настроить.

Обзор

Хеджирование — это одна из двух настраиваемых политик повторных попыток, поддерживаемых gRPC. При хеджировании клиент gRPC отправляет несколько копий одного и того же запроса на разные серверы и использует первый полученный ответ. Затем клиент отменяет все невыполненные запросы и отправляет ответ приложению.

Варианты использования

Хеджирование — это метод сокращения задержки в конце очереди в крупномасштабных распределённых системах. Хотя наивные реализации могут значительно увеличить нагрузку на серверы, можно добиться большей части эффекта сокращения задержки, лишь незначительно увеличив нагрузку.

Подробное обсуждение задержек в работе «хвоста» можно найти в основополагающей статье «Хвост в масштабе» Джеффа Дина и Луиса Андре Баррозу.

Настройка хеджирования в gRPC

Хеджирование настраивается с помощью конфигурации службы gRPC для каждого метода в отдельности. Конфигурация содержит следующие параметры:

"hedgingPolicy": {

"maxAttempts": INTEGER,

"hedgingDelay": JSON proto3 Duration type,

"nonFatalStatusCodes": JSON array of grpc status codes (int or string)

}

* maxAttempts: максимальное количество запросов в процессе ожидания успешного ответа. Это обязательное поле, которое необходимо указать. Если указанное значение больше 5, gRPC использует значение 5.
* hedgingDelay: количество времени, которое должно пройти, прежде чем клиент отправит следующий запрос в ожидании успешного ответа. Это поле необязательно, и если оно не указано, то maxAttempts количество запросов будет отправлено одновременно.
* nonFatalStatusCodes: an optional list of grpc status codes. If any of hedged requests fails with a status code that is not present in this list, all outstanding requests are canceled and the response is returned to the application.

Hedging policy

When the application makes an RPC call that contains a hedgingPolicy configuration in the Service Config, the original RPC is sent immediately, as with a standard non-hedged call. After hedgingDelay has elapsed without a successful response, the second RPC will be issued. If neither RPC has received a response after hedgingDelay has elapsed again, a third RPC is sent, and so on, up to maxAttempts. gRPC call deadlines apply to the entire chain of hedged requests. Once the deadline has passed, the operation fails regardless of in-flight RPCS, and regardless of the hedging configuration.

When a successful response is received (in response to any of the hedged requests), all outstanding hedged requests are canceled and the response is returned to the client application layer.

If an error response with a non-fatal status code (controlled by the nonFatalStatusCodes field) is received from a hedged request, then the next hedged request in line is sent immediately, shortcutting its hedging delay. If any other status code is received, all outstanding RPCs are canceled and the error is returned to the client application layer.

If all instances of a hedged RPC fail, there are no additional retry attempts. Essentially, hedging can be seen as retrying the original RPC before a failure is even received.

If server pushback that specifies not to retry is received in response to a hedged request, no further hedged requests should be issued for the call.

Throttling Hedged RPCs

gRPC provides a way to throttle hedged RPCs to prevent server overload. Throttling can be configured via the Service Config as well using the RetryThrottlingPolicy message. The throttling configuration contains the following:

"retryThrottling": {

"maxTokens": 10,

"tokenRatio": 0.1

}

Для каждого имени сервера клиент gRPC поддерживает token\_count значение, которое изначально равно max\_tokens. Каждое исходящее RPC-вызов (независимо от службы или вызываемого метода) изменяет token\_count следующим образом:

* Каждый сбойный RPC-процесс будет уменьшать значение token\_count на 1.
* Каждый успешный RPC будет увеличивать значение token\_count на token\_ratio.

При хеджировании первый запрос отправляется всегда, но последующие хеджированные запросы отправляются только в том случае, если token\_count больше порогового значения (определяемого как max\_tokens / 2). Если token\_count меньше или равно пороговому значению, хеджированные запросы не блокируются. Вместо этого они отменяются, и если нет других уже отправленных хеджированных RPC, то клиентскому приложению возвращается сообщение об ошибке.

В соответствии с политикой ограничения скорости в качестве неудачных запросов учитываются только те, которые завершаются с кодом состояния, не являющимся критическим, или получают ответ с отказом от повторной попытки. Это позволяет избежать путаницы между сбоями сервера и ответами на некорректные запросы (например, с кодом состояния INVALID\_ARGUMENT).

Откат сервера

Серверы могут явно отклонять запросы, устанавливая метаданные в ответе клиенту. Если в отклонении запроса указано, что повторная попытка не требуется, дальнейшие запросы с хеджированием не будут отправляться. Если в отклонении запроса указано, что повторная попытка будет предпринята после заданной задержки, следующий запрос с хеджированием (если он есть) будет отправлен после истечения заданной задержки.

Откат сервера задаётся с помощью ключа метаданных grpc-retry-pushback-ms. Значение представляет собой 32-битное целое число со знаком в кодировке ASCII без лишних ведущих нулей, которое указывает, сколько миллисекунд нужно подождать перед отправкой следующего запроса с хеджированием. Если значение для отката отрицательное или не поддаётся анализу, то это будет воспринято как просьба сервера к клиенту вообще не повторять попытку.

Ресурсы

* Хвост В масштабе
* Конфигурация службы gRPC
* Конструкция Повторной попытки gRPC

Языковая Поддержка

ЯзыкПримерJavaПример JavaC++Пока недоступноВпередПока не поддерживается

Последнее изменение 3 октября 2023 г.: добавлено руководство по хеджированию запросов (#1199) (d357308)