**Продолжай Жить**

Как использовать HTTP/2 PING для поддержания работоспособности в gRPC.

Обзор

Поддерживающие соединения HTTP/2 на основе PING — это способ поддерживать соединение HTTP/2 активным, даже если данные не передаются. Для этого периодически отправляется кадр PING на другой конец соединения. Поддерживающие соединения HTTP/2 могут повысить производительность и надёжность соединений HTTP/2, но важно правильно настроить интервал поддерживающего соединения.

Примечание

Существует связанная с этим, но отдельная проблема, которая называется [проверкой работоспособности]. Проверка работоспособности позволяет серверу сообщать о том, что *сервис* работает, в то время как keepalive касается только *соединения*.

Предыстория

TCP keepalive — это хорошо известный метод поддержания соединений и обнаружения разорванных соединений. Когда функция TCP keepalive включена, обе стороны соединения могут отправлять дополнительные пакеты. Если другая сторона подтвердит получение пакета, соединение будет считаться активным. Если после повторных попыток подтверждения не поступает, соединение считается разорванным.

В отличие от TCP keepalive, gRPC использует HTTP/2, который предоставляет обязательный кадр PING, который можно использовать для оценки времени прохождения сигнала в обе стороны, произведения пропускной способности на задержку или для проверки соединения. Интервал и повторные попытки в TCP keepalive не совсем применимы к PING, поскольку транспортный протокол является надёжным, поэтому в реализации keepalive на основе PING в gRPC они заменены на тайм-аут (эквивалентный интервалу \* повторным попыткам).

Примечание

Владельцам сервисов не требуется поддерживать keepalive. **Авторы клиентов должны согласовать с владельцами сервисов,** приемлемы ли конкретные настройки на стороне клиента. Владельцы сервисов сами решают, что они готовы поддерживать, в том числе, готовы ли они вообще получать keepalives (если сервис не поддерживает keepalive, первые несколько пингов keepalive будут проигнорированы, и сервер в конечном итоге отправит GOAWAY сообщение с отладочными данными, равными ASCII-коду для too\_many\_pings).

How configuring keepalive affects a call

Keepalive is less likely to be triggered for unary RPCs with quick replies. Keepalive is primarily triggered when there is a long-lived RPC, which will fail if the keepalive check fails and the connection is closed.

For streaming RPCs, if the connection is closed, any in-progress RPCs will fail. If a call is streaming data, the stream will also be closed and any data that has not yet been sent will be lost.

Warning

To avoid DDoSing, it’s important to take caution when setting the keepalive configurations. Thus, it is recommended to avoid enabling keepalive without calls and for clients to avoid configuring their keepalive much below one minute.

Common situations where keepalives can be useful

gRPC HTTP/2 keepalives can be useful in a variety of situations, including but not limited to:

* When sending data over a long-lived connection which might be considered as idle by proxy or load balancers.
* When the network is less reliable (For example, mobile applications).
* When using a connection after a long period of inactivity.

Keepalive configuration specification

OptionsAvailabilityDescriptionClient DefaultServer DefaultKEEPALIVE\_TIMEClient and ServerThe interval in milliseconds between PING frames.INT\_MAX (Disabled)7200000 (2 hours)KEEPALIVE\_TIMEOUTКлиент и СерверТайм-аут в миллисекундах для подтверждения получения кадра PING. Если отправитель не получит подтверждение в течение этого времени, он закроет соединение.20000 (20 секунд)20000 (20 секунд)KEEPALIVE\_WITHOUT\_CALLSКлиентДопустимо ли отправлять запросы на подтверждение соединения с клиента без открытых потоков.0 (ложь)N/APERMIT\_KEEPALIVE\_WITHOUT\_CALLSСерверДопустимо ли отправлять запросы на подтверждение соединения с клиента без открытых потоков.N/A0 (ложь)PERMIT\_KEEPALIVE\_TIMEСерверМинимально допустимое время между получением сервером последовательных кадров ping без отправки каких-либо данных/заголовков.N/A300000 (5 минут)MAX\_CONNECTION\_IDLEСерверМаксимальное время, в течение которого канал может не обрабатывать запросы, после чего сервер закроет соединение.N/AINT\_MAX (Бесконечный)MAX\_CONNECTION\_AGEСерверМаксимальное время, в течение которого может существовать канал.N/AINT\_MAX (Бесконечный)MAX\_CONNECTION\_AGE\_GRACEСерверЛьготный период после достижения каналом максимального возраста.N/AINT\_MAX (Бесконечный)

Примечание

В некоторых языках могут быть доступны дополнительные опции. Пожалуйста, ознакомьтесь с примерами на разных языках и дополнительными ресурсами для получения более подробной информации.

Языковые руководства и примеры

ЯзыкПримерДокументацияC++Пример C ++Документация по C ++ВпередПерейти к ПримеруПерейти к документацииJavaПример JavaДокументация JavaПитонПример PythonДокументация по Python

Дополнительные ресурсы

* gRFC для клиентского Keepalive
* gRFC для управления подключениями на стороне сервера
* Использование gRPC для долговременных и потоковых RPC

Последнее изменение 9 октября 2024 года: добавлена ссылка на видео gRPC Keepalive на YouTube (#1370) (79b65fd)