**Крайние сроки**

Объясняет, как можно использовать крайние сроки для эффективной работы с ненадежными бэкендами.

Обзор

Крайний срок используется для указания момента времени, после которого клиент не желает ждать ответа от сервера. Эта простая идея очень важна при создании надёжных распределённых систем. Клиенты, которые не ждут без необходимости, и серверы, которые знают, когда нужно прекратить обработку запросов, повысят эффективность использования ресурсов и сократят задержки в вашей системе.

Обратите внимание, что в то время как в некоторых языковых API-интерфейсах есть понятие **срока выполнения**, в других используется понятие **таймаута**. Когда API запрашивает срок выполнения, вы указываете момент времени, до которого вызов не должен выполняться. Тайм-аут — это максимальная продолжительность вызова. Тайм-аут можно преобразовать в срок выполнения, добавив его к текущему времени, когда приложение начинает вызов. Для простоты в этом документе мы будем использовать только термин «срок выполнения».

Сроки выполнения зависят от Клиента

По умолчанию gRPC не устанавливает крайний срок, а это значит, что клиент может ждать ответа практически бесконечно. Чтобы этого избежать, вы всегда должны явно устанавливать реалистичный крайний срок для своих клиентов. Чтобы определить подходящий крайний срок, в идеале нужно начать с обоснованного предположения, основанного на том, что вы знаете о своей системе (задержка в сети, время обработки на сервере и т. д.), и подтвердить его с помощью нагрузочного тестирования.

Если сервер не уложился в срок при обработке запроса, клиент прекратит работу и завершит RPC со статусом DEADLINE\_EXCEEDED

Крайние сроки на Сервере

Сервер может получать запросы RPC от клиента с нереалистично коротким сроком выполнения, из-за чего у сервера не будет достаточно времени, чтобы вовремя ответить. Это приведёт к тому, что сервер просто будет тратить ценные ресурсы, а в худшем случае приведёт к сбою сервера. Сервер gRPC справляется с этой ситуацией, автоматически отменяя вызов (CANCELLED статус) по истечении заданного клиентом срока выполнения.

Обратите внимание, что серверное приложение отвечает за остановку любых процессов, запущенных для обслуживания RPC. Если ваше приложение запускает длительный процесс, вам следует периодически проверять, не был ли отменен инициировавший его RPC, и если да, то останавливать обработку.

Распространение Крайнего срока

Вашему серверу может потребоваться вызвать другой сервер для получения ответа. В таких случаях, когда ваш сервер также выступает в качестве клиента, вы захотите соблюдать крайний срок, установленный исходным клиентом. Некоторые реализации gRPC поддерживают автоматическое распространение крайнего срока от входящего RPC к исходящему. В некоторых языках это поведение необходимо включать явно (например, C++), а в других оно включено по умолчанию (например, Java и Go). Использование этой функции позволяет избежать ошибок при ручном указании срока действия для каждого исходящего запроса RPC.

Поскольку крайний срок устанавливается в определённый момент времени, его передача на сервер в неизменном виде может быть проблематичной, так как часы на двух серверах могут быть не синхронизированы. Чтобы решить эту проблему, gRPC преобразует крайний срок в тайм-аут, из которого вычитается уже прошедшее время. Это защищает вашу систему от любых проблем, связанных с расходом времени.

ClientUser ServerBilling ServerRequest at 13:00:00Should complete in 2sGetUserProfile(deadline: 13:00:02)0.5s spent beforecalling billing serverGetTransactionHistory(timeout: 1.5s)Retrieve transactionsIt's 13:00:02Time's up!Stop waiting for serverStop waiting for serverDEADLINE\_EXCEEDEDStop waiting for serverCancelCancelClean up resources(after noticing that thecall was cancelled)

Языковая Поддержка

ЯзыкПримерJavaПример JavaВпередПерейти к примеруC++ПитонПример Python

Другие Ресурсы

* Крайние сроки публикации в блоге

Последнее изменение 26 июля 2024 г.: docs/guides/deadlines: замените «запрос» на «RPC» (#1336) (63ad871)