**Сравнительный анализ**

gRPC разработан для поддержки высокопроизводительных RPC с открытым исходным кодом на многих языках. На этой странице описаны инструменты для оценки производительности, сценарии, рассматриваемые в тестах, и инфраструктура тестирования.

Обзор

gRPC предназначен как для высокопроизводительных, так и для высокопродуктивных распределённых приложений. Непрерывное тестирование производительности является важной частью рабочего процесса разработки gRPC. Каждые несколько часов выполняются многоязычные тесты производительности для основной ветки, и эти показатели отображаются на панели управления для визуализации.

* Многоязычная панель управления производительностью @ master (последняя версия для разработчиков)
* Устаревшая панель мониторинга (те же данные, что и выше)

Дизайн тестирования производительности

В каждом языке есть модуль для тестирования производительности, который реализует WorkerService gRPC. Этот модуль управляет работой модуля в качестве клиента или сервера для фактического теста производительности, представленного как BenchmarkService. У этого модуля есть два метода:

* UnaryCall — однократный RPC-запрос с простым запросом, в котором указывается количество байтов, возвращаемых в ответе.
* StreamingCall — потоковый RPC, который позволяет многократно отправлять запросы и получать ответы, подобно UnaryCall.

Эти рабочие процессы управляются драйвером, который принимает на вход описание сценария (в формате JSON) и переменную среды, указывающую хост и порт каждого рабочего процесса.

Тестируемые языки

Следующие языки проходят непрерывное тестирование производительности как на клиентах, так и на серверах master:

* C++
* Java
* Вперед
* C#
* Node.js
* Питон
* Рубин

Помимо тестирования производительности как на стороне клиента, так и на стороне сервера, все языки тестируются в качестве клиентов на сервере C++ и в качестве серверов на клиенте C++. Этот тест предназначен для определения текущего верхнего предела производительности клиентской или серверной реализации конкретного языка без тестирования другой стороны.

Хотя PHP или мобильные среды не поддерживают сервер gRPC (который необходим для наших тестов производительности), их производительность на стороне клиента можно сравнить с помощью прокси-сервера WorkerService, написанного на другом языке. Этот код реализован для PHP, но пока не находится в режиме непрерывного тестирования.

Тестируемые сценарии

На панелях управления выше представлено несколько важных сценариев тестирования, в том числе следующие:

* Безконфликтная задержка — средняя и максимальная задержки отклика, наблюдаемые при отправке только одним клиентом одного сообщения за раз с помощью StreamingCall.
* QPS — количество сообщений в секунду при наличии 2 клиентов и в общей сложности 64 каналов, каждый из которых одновременно отправляет 100 сообщений с помощью StreamingCall.
* Масштабируемость (для выбранных языков) — количество сообщений в секунду на одно ядро сервера.

В большинстве тестов производительности используется безопасная связь и протоколы. В некоторых тестах C++ дополнительно используется небезопасная связь и универсальный (не основанный на протоколе) API для демонстрации максимальной производительности. В будущем могут быть добавлены дополнительные сценарии.

Инфраструктура тестирования

Все тесты производительности выполняются в нашем выделенном кластере GKE, где каждый работник теста (клиент или сервер) назначается на отдельный узел GKE (и каждый узел GKE представляет собой отдельную виртуальную машину GCE) в одном из наших рабочих пулов. Исходный код используемой нами системы тестирования доступен в репозитории test-infra на GitHub.

Большинство тестовых экземпляров представляют собой 8-ядерные системы, которые используются как для измерения задержки, так и для измерения QPS. Для C++ и Java мы дополнительно поддерживаем тестирование QPS на 32-ядерных системах. Во всех тестах QPS для каждого сервера используются 2 идентичные клиентские машины, чтобы убедиться, что измерение QPS не ограничено клиентом.

Последнее изменение 24 января 2022 года: обновите ссылку на панель управления на странице тестирования. (#930) (e7c6231)