Нагрузочное тестирование с применением Tarantool

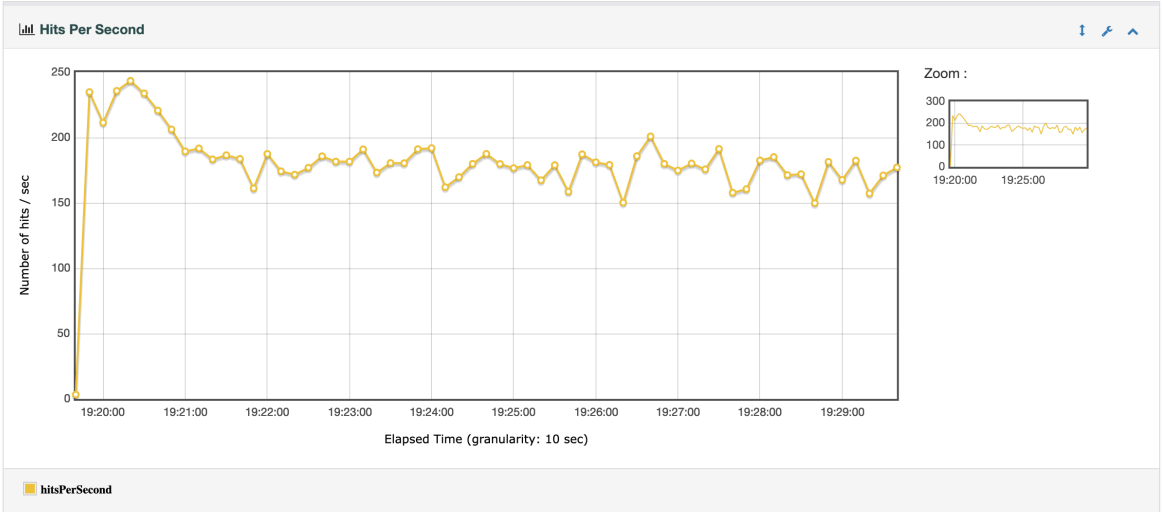
Подготовительный этап:

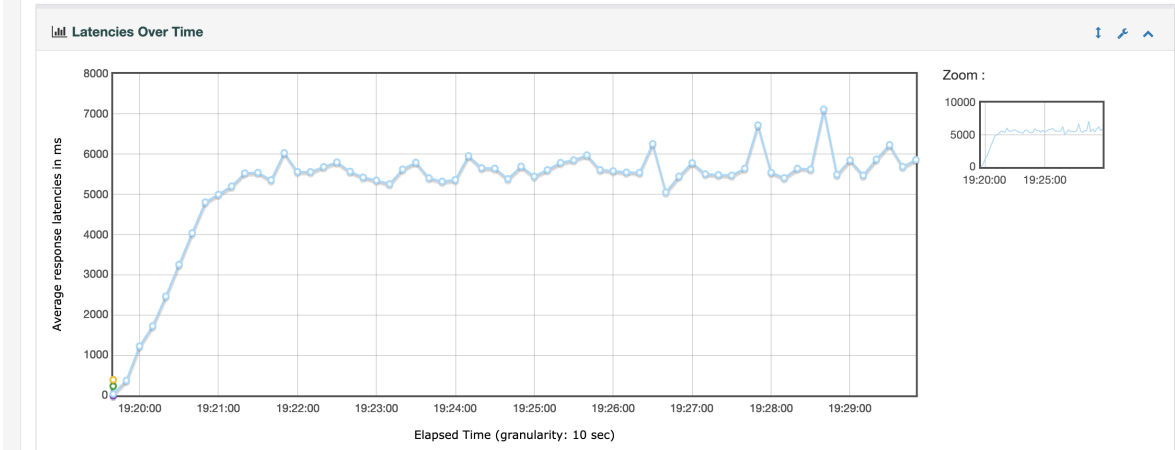
1. Запустим окружение согласно READMY.MD, которое включает инста приложения, инстанс Postgres, инстанс Tarantool и ряд других сервисов.
2. При старте инстанса Postgres будет выполнено наполнение таблицы users данными (6500 записей).
3. Аналогично развернутый инстанс Tarantool будет иметь спейс users, наполненный записями аналогично Postgres. Подробнее см. скрипты lua в каталоге perfomance-test-tarantool.

Нагрузочное тестирование ДО, сценарий:

1. Создание нового пользователя: в один поток отправляется запрос на сервер для регистрации нового пользователя.
2. Авторизация, получение токена: обращение к серверу по полученному в п. 1 идентификатору.
3. Поиск случайного пользователя из заданного набора данных в течении 10 минут: 1000 пользователей параллельно (по 10 на каждый запрос). Старт с 0 до 1000 в течении минуты. Верификация, что каждый ответ получил 200 Ок.
   - Поиск по имени и фамилии на эндпойнте **/user/search**
   - Все ответы приходят с кодом 200 Ок.

Нагрузочное тестирование ДО - результат:



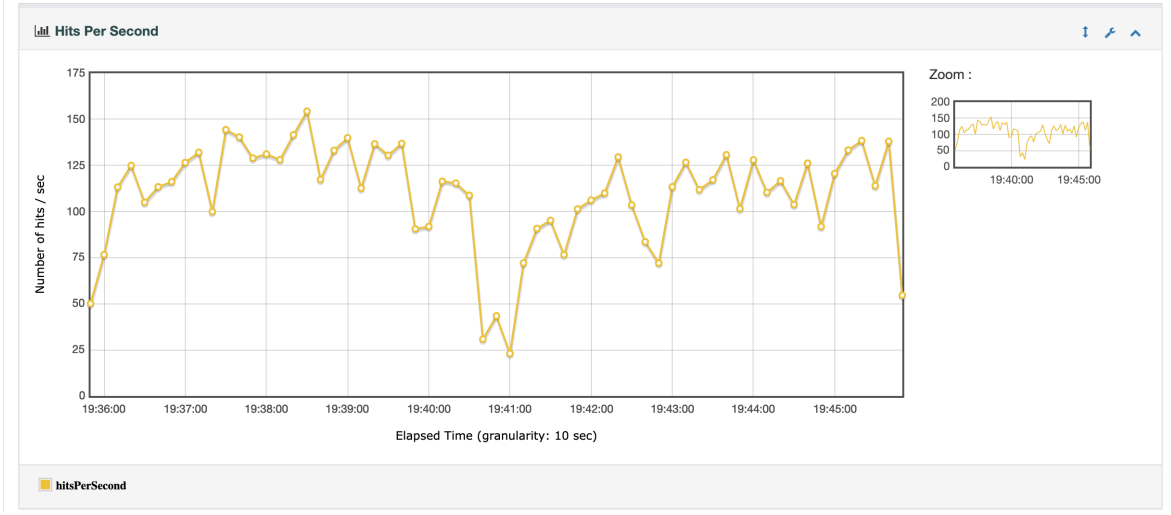


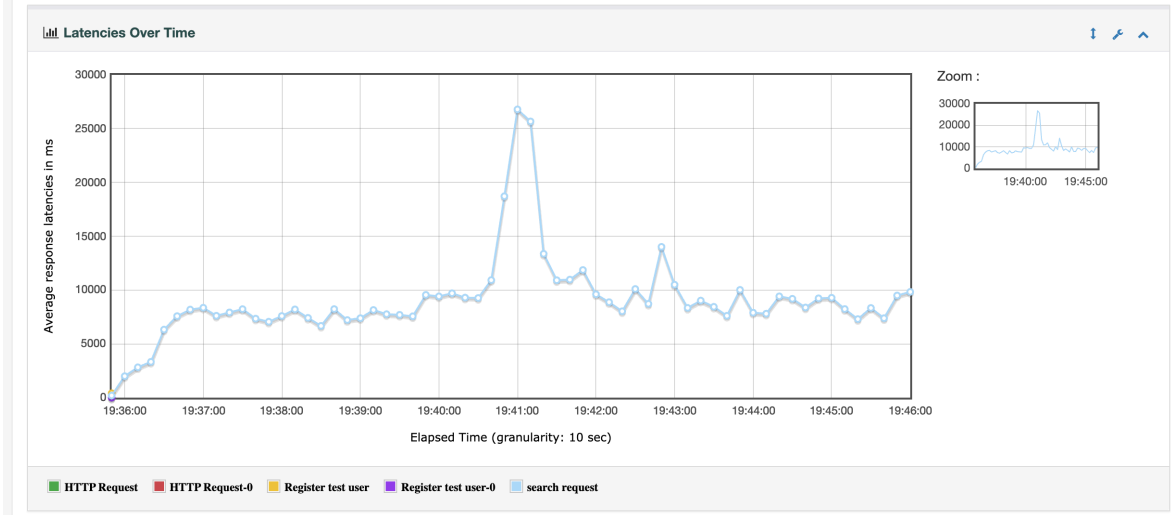
Вывод: среднее время отклика находилось в районе 5,5 секунд, среднее количество запросов - примерно 175.

Нагрузочное тестирование ПОСЛЕ, сценарий:

1. Создание нового пользователя: в один поток отправляется запрос на сервер для регистрации нового пользователя.
2. Авторизация, получение токена: обращение к серверу по полученному в п. 1 идентификатору.
3. Поиск случайного пользователя из заданного набора данных в течении 10 минут: 1000 пользователей параллельно (по 10 на каждый запрос). Старт с 0 до 1000 в течении минуты. Верификация, что каждый ответ получил 200 Ок.
   - Поиск по имени и фамилии на эндпойнте **/user/tarantool/search** - при вызове данного эндпойнта будет происходить обращение не к Postgres, а к Tarantool исключительно.
   - Все ответы приходят с кодом 200 Ок.

Нагрузочное тестирование ПОСЛЕ, результат:



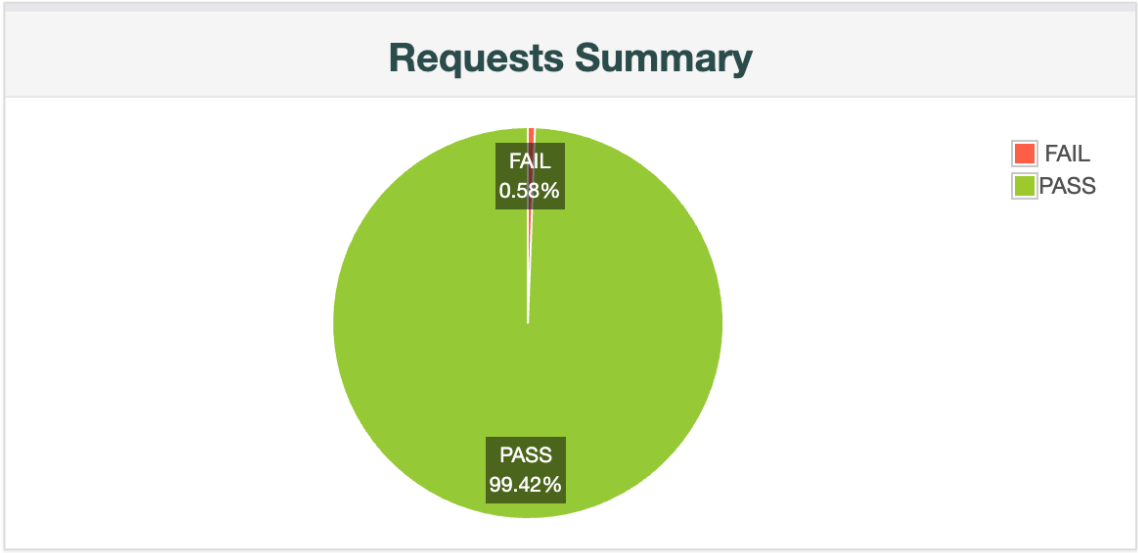


В ходе теста в логах появлялись ошибки:

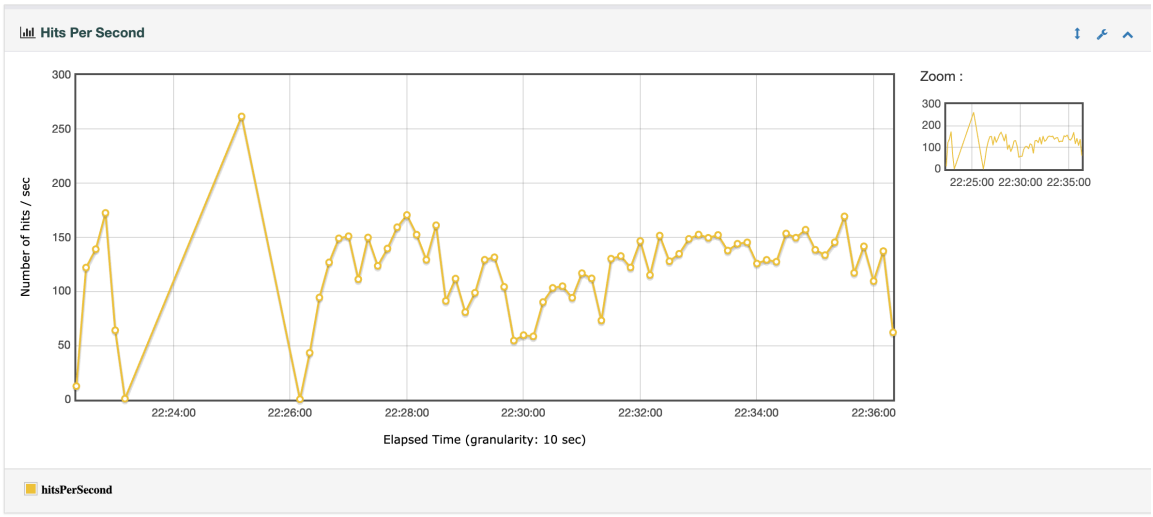
2023-07-25 19:42:54 2023-07-25 16:42:54.062 ERROR 1 --- [o-8080-exec-700] o.a.c.c.C.[.[.[/].[dispatcherServlet] : Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed; nested exception is **org.springframework.dao.DataRetrievalFailureException**: **java.util.concurrent.TimeoutException: Failed to get response for request id: 45407 within 5000 ms;** nested exception is java.util.concurrent.TimeoutException: Failed to get response for request id: 45407 within 5000 ms] with root cause
2023-07-25 19:42:54

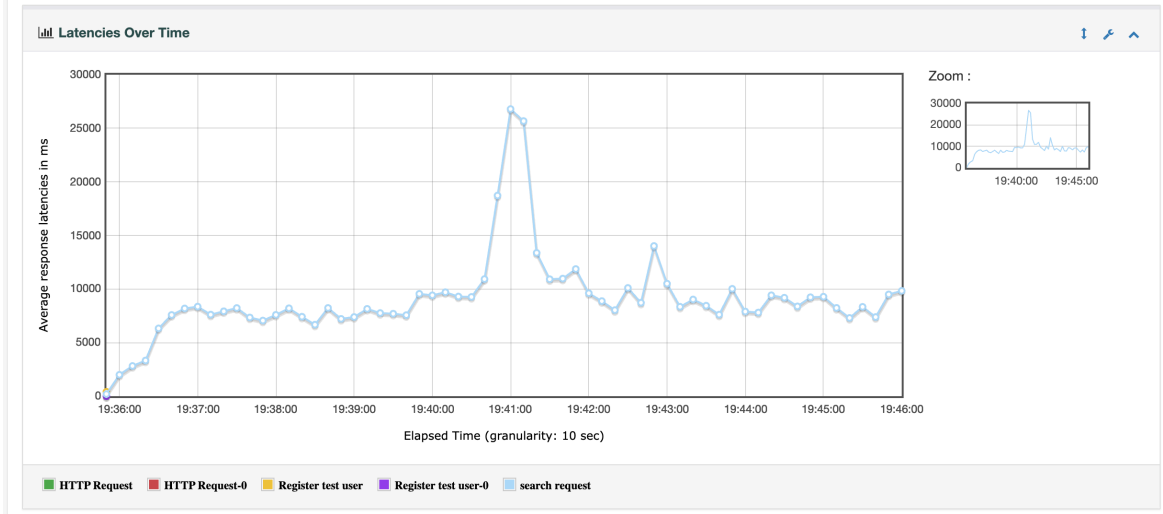
2023-07-25 19:42:54 java.util.concurrent.TimeoutException: Failed to get response for request id: 45407 within 5000 ms
2023-07-25 19:42:54 at io.tarantool.driver.core.RequestFutureManager.lambda$submitRequest$1(RequestFutureManager.java:68) ~[cartridge-driver-0.12.0.jar!/:na]
2023-07-25 19:42:54 at java.base/java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:539) ~[na:na]
2023-07-25 19:42:54 at java.base/java.util.concurrent.FutureTask.run(FutureTask.java:264) ~[na:na]
2023-07-25 19:42:54 at java.base/java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:304) ~[na:na]
2023-07-25 19:42:54 at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1136) ~[na:na]
2023-07-25 19:42:54 at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:635) ~[na:na]
2023-07-25 19:42:54 at java.base/java.lang.Thread.run(Thread.java:833) ~[na:na]
2023-07-25 19:42:54
2023-07-25 19:42:54 2023-07-25 16:42:54.028 INFO 1 --- [ntLoopGroup-5-1] i.t.d.handlers.TarantoolResponseHandler : Request 45374 is not registered in this client instance

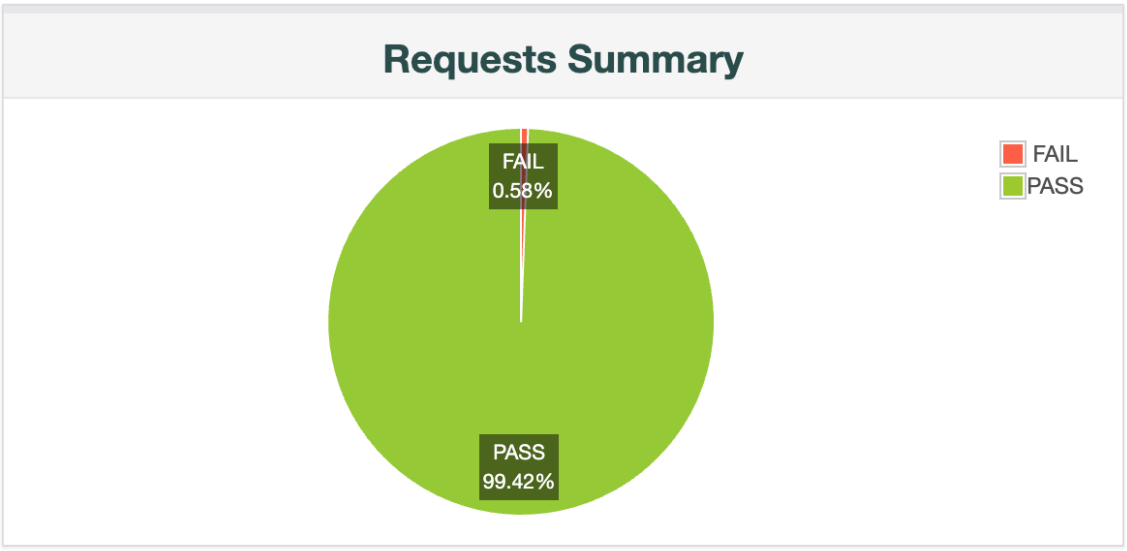
То есть часть запросов - менее 1% падали по таймауту.



После замены библиотеки интеграции с tarntool (spring-data-tarantool -> cartridge driver) ситуация только ухудшилась:







Вывод: нагрузочное тестирование показало, что Postgres показывает лучший результат по сравнению с Tarantool в режиме standalone. Дальнейшая отладка тайм-аутов и развертывание tarantool в кластером режиме может показать улучшение производительности, но это выходит за рамки данного теста.