Сценарий нагрузки

В БД 100к записей, для нагрузки используется 2 типа пользователей (каждый использует только один метод):

- 10 запросов из 100 — делают поиск по случайному (т. е. Скорее всего отсутствующему) слову в трех строковых полях проекта (название, описание, отдел магазина).

90 запросов из 100 — выбирают из массива констант слово, которое точно встечается в характеристиках проекта (название, описание, отдел магазина) и вызывают метод поиска с этим словом.

Профиль нагрузки

В нагрузочном использовался следующий сценарий нагрузки:

- ступенчатый профиль — добавлять по 25 пользователей каждые 15 секунд

Выполнялся следующей командой:

locust -f locustfile.py --headless -u 100000 -r 10 --run-time 10m --host http://arch.homework --step-load --step-users 25 --step-time 15s

Для версии приложения с и без кэширование использовался одинаковый сценарий нагрузки, описанный выше.

Версия с отключенным кэшированием

График нагрузки ресурсов:

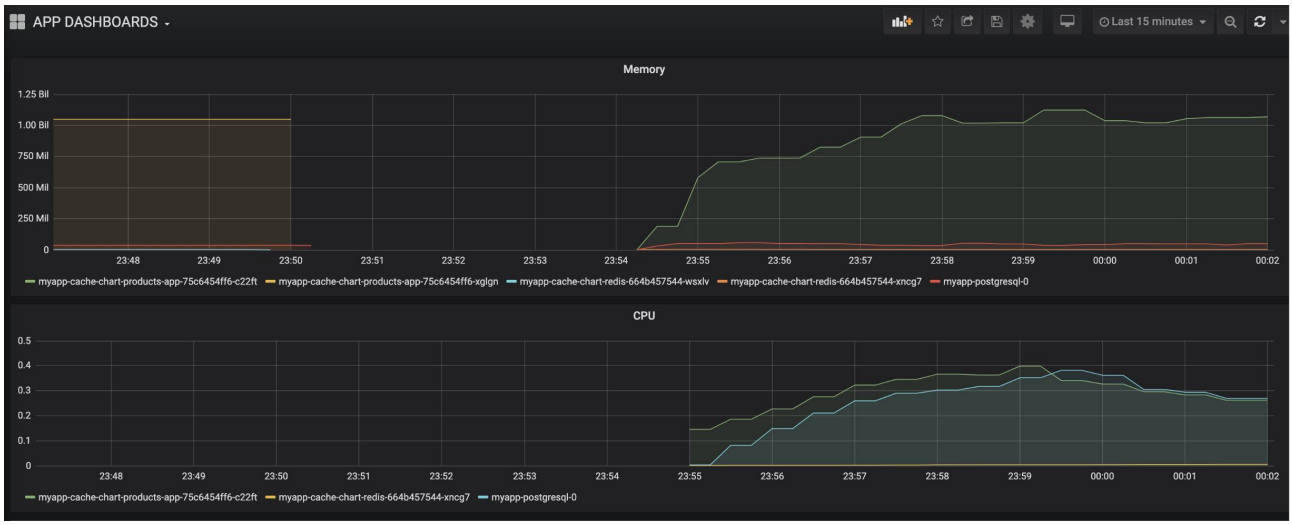
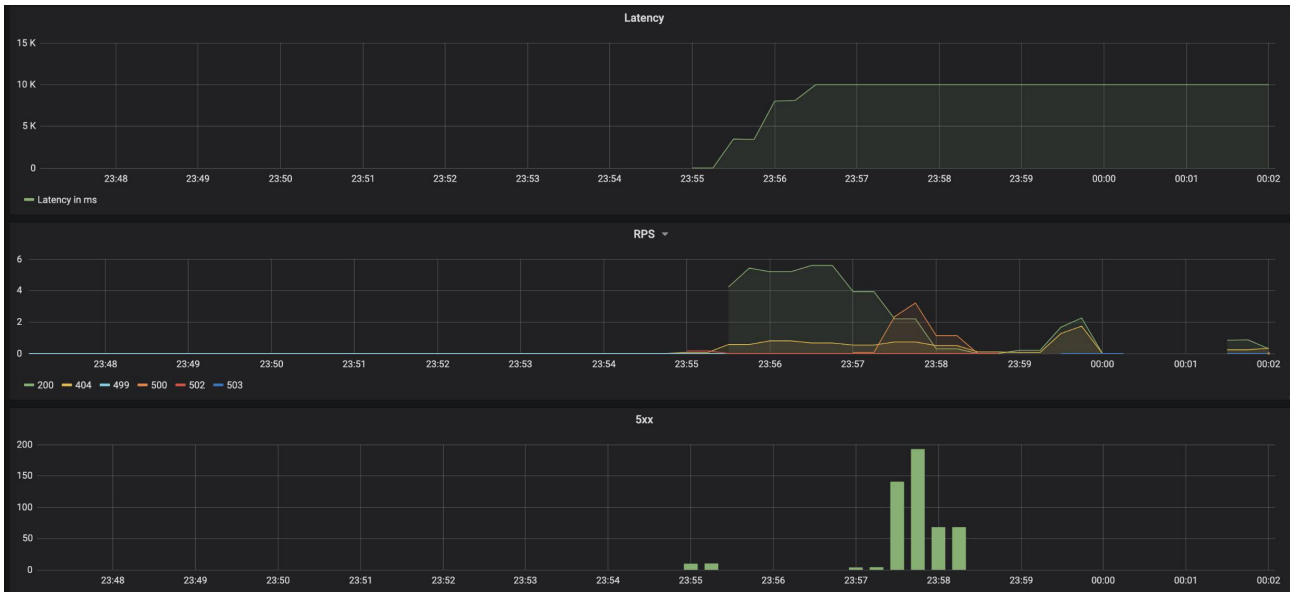


График загрузки приложения:



Выводы по приложению без кэша

Утилизация ресурсов достигла максимума через 4 минут
Точка деградации — секунд 10
Точка отказа — 1 минута (чуть меньше)
Ошибки появились после 2.5 минут работы
Приложение полностью **развалилось** через 2 минуты

Приложение вышло на плато RPS по 200-м запросам через 1 минуту
Каждую минуту нагрузчик добавляет по 100 пользователей (25 каждые 15 секунд: 25 * 4 = 100), т. е.
Пиковая нагрузка достигнута после 1 минуты: 100 пользователей

Рабочая нагрузка
Рабочая нагрузка : пиковая * 0.8 = 100 * 0.8 = **80 пользователей** с заданным профилем

Версия с задействованным кэшированием

График нагрузки ресурсов:

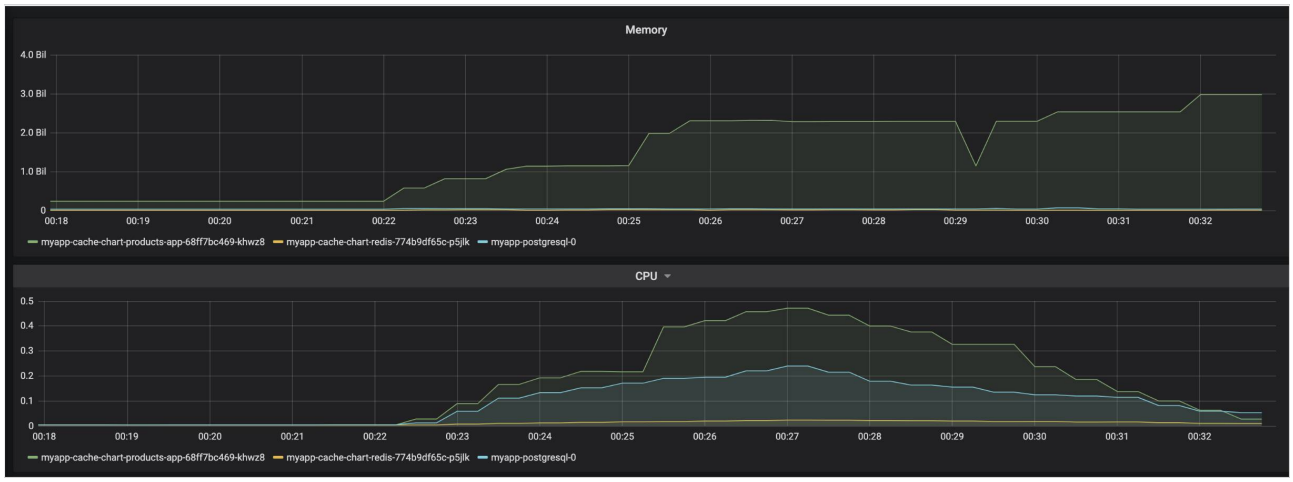
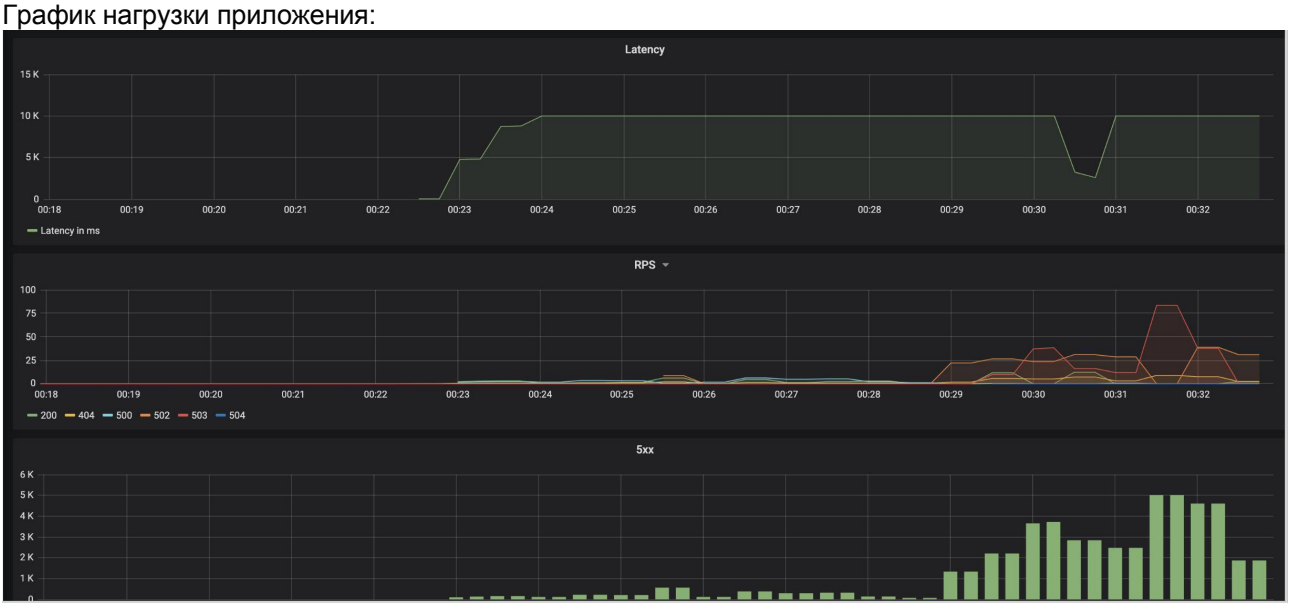


График нагрузки приложения:



Утилизация ресурсов достигла максимума через 5 минут
Точка деградации приложения — 4.5 минуты (в 00:26 с половиной минут)
Точка отказа — 6 минут (00:26)
Существенные **ошибки** появились через 6 минут работы

Приложение вышло на плато RPS по 200-м запросам с 00:26 по 00:28, это с 3й по 5ю минуту с начала нагрузочного тестирования.
Каждую минуту нагрузчик добавляет по 100 пользователей (25 каждые 15 секунд: 25 * 4 = 100), т. е.
Пиковая нагрузка после 3х минут: 3 минуты * 100 пользователей = 300 пользователей

Рабочая нагрузка
Рабочая нагрузка : пиковая * 0.8 = 300 * 0.8 = **240 пользователей** с заданным профилем

Общие выводы

Приложение без кэширование ожидаемо выдерживает меньшую нагрузку, чем с кэшированием, и деградирует раньше по времени.
Приложение без кэширование существенно больше потребляем памяти, чем PostgreSQL. Ожидаемо, т.к. приложение на Java.
Приложение с кэшированием выходит на плато по памяти и больше не растет.