Алтайский государственный технический университет
им.И.И. Ползунова, г. Барнаул, Россия

Разработка информационно-аналитической системы для исследования производительности и отказоустойчивости программных систем в зависимости от архитектурных решений и протоколов обмена сообщениями

Выполнил: Магистрант Факультета Информационных технологий,
группы 8ПИ-31, Алексенко Алексей Викторович,
e-mail: alexeyalexenko@yandex.ru

Научный руководитель: Профессор кафедры ПМ,
к.ф.-м.н. Крючкова Елена Николаевна,
e-mail: kruchkova_elena@mail.ru

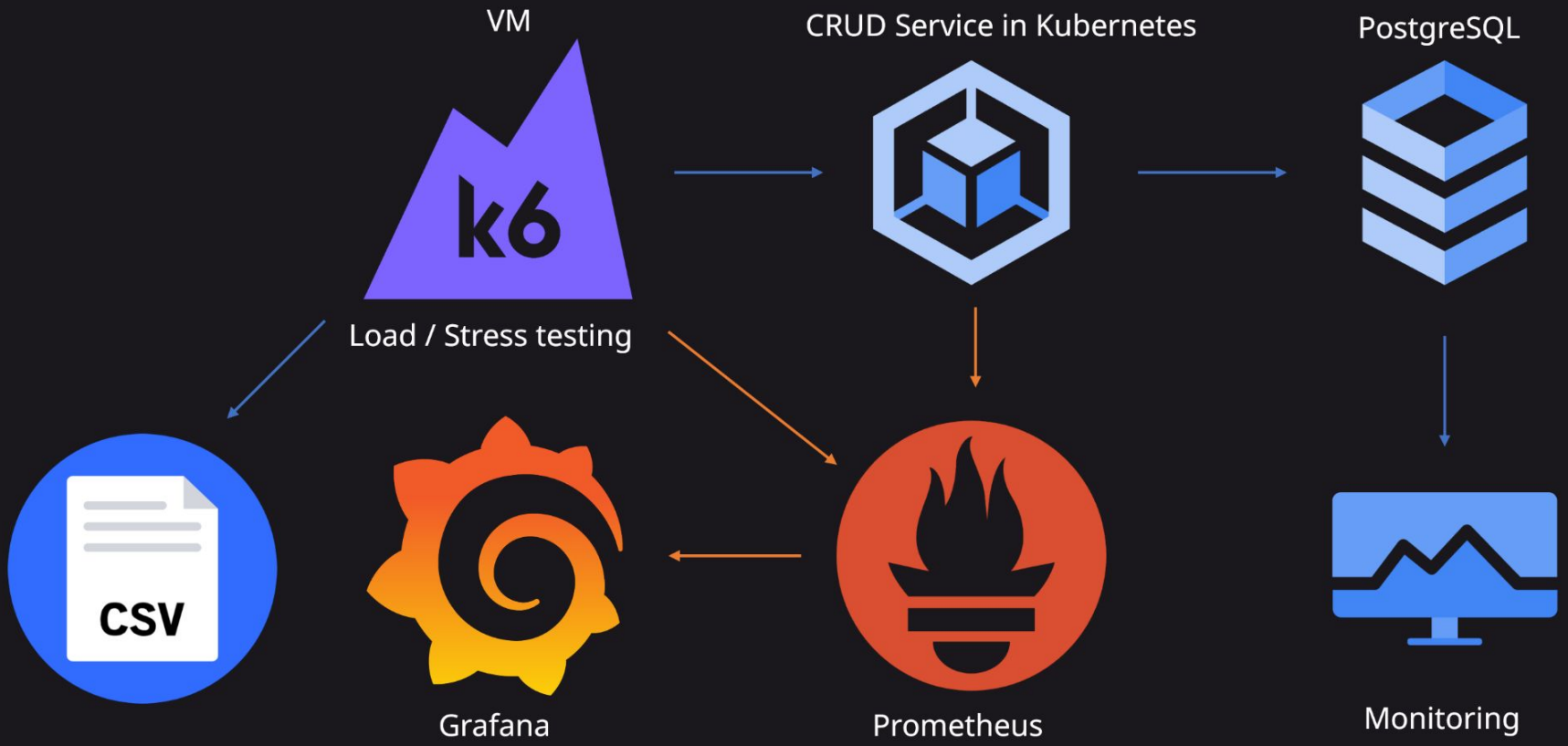
Актуальность

- Существование большого количества инструментов
- Отсутствие рекомендаций по их применению
- Отсутствие инструментария для сравнительного анализа эффективности решений

Задачи

- Разработать инструментарий для анализа эффективности комбинации различных решений
- На основе разработанного инструментария провести исследование имеющихся решений
- Выявить условия эффективной применимости исследованных решений

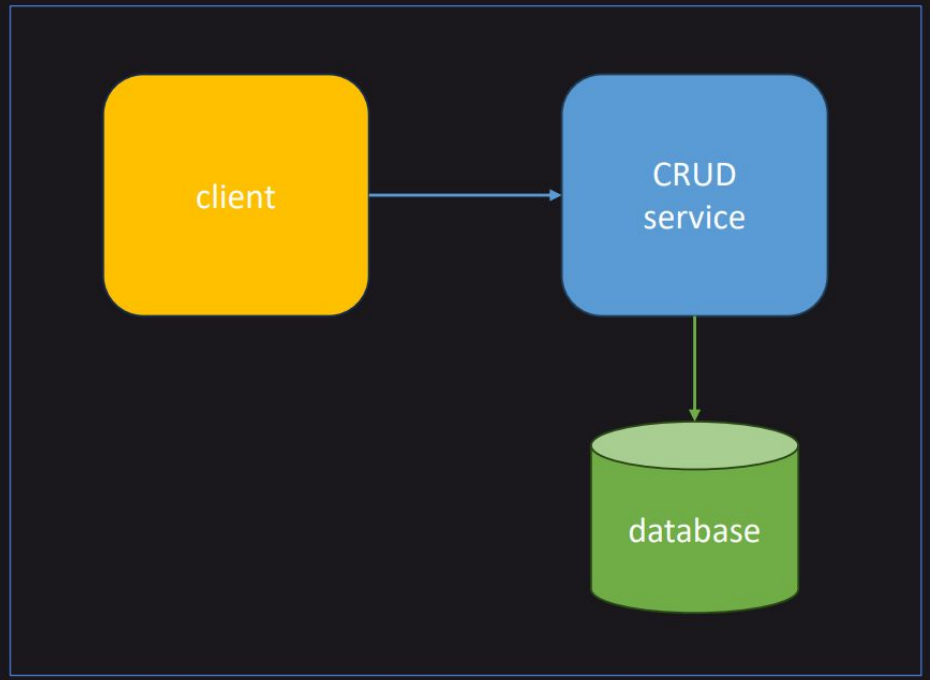
Тестовый стенд



Описание архитектуры сервиса

Способы реализации:

1. Blocking + Blocking: webmvc-jdbc
2. Reactive + Blocking: webflux-jdbc
3. Reactive + Reactive: webflux-r2dbc



Способы реализации

webmvc-jdbc

1. webmvc-jdbc
2. webmvc-jdbc-loom

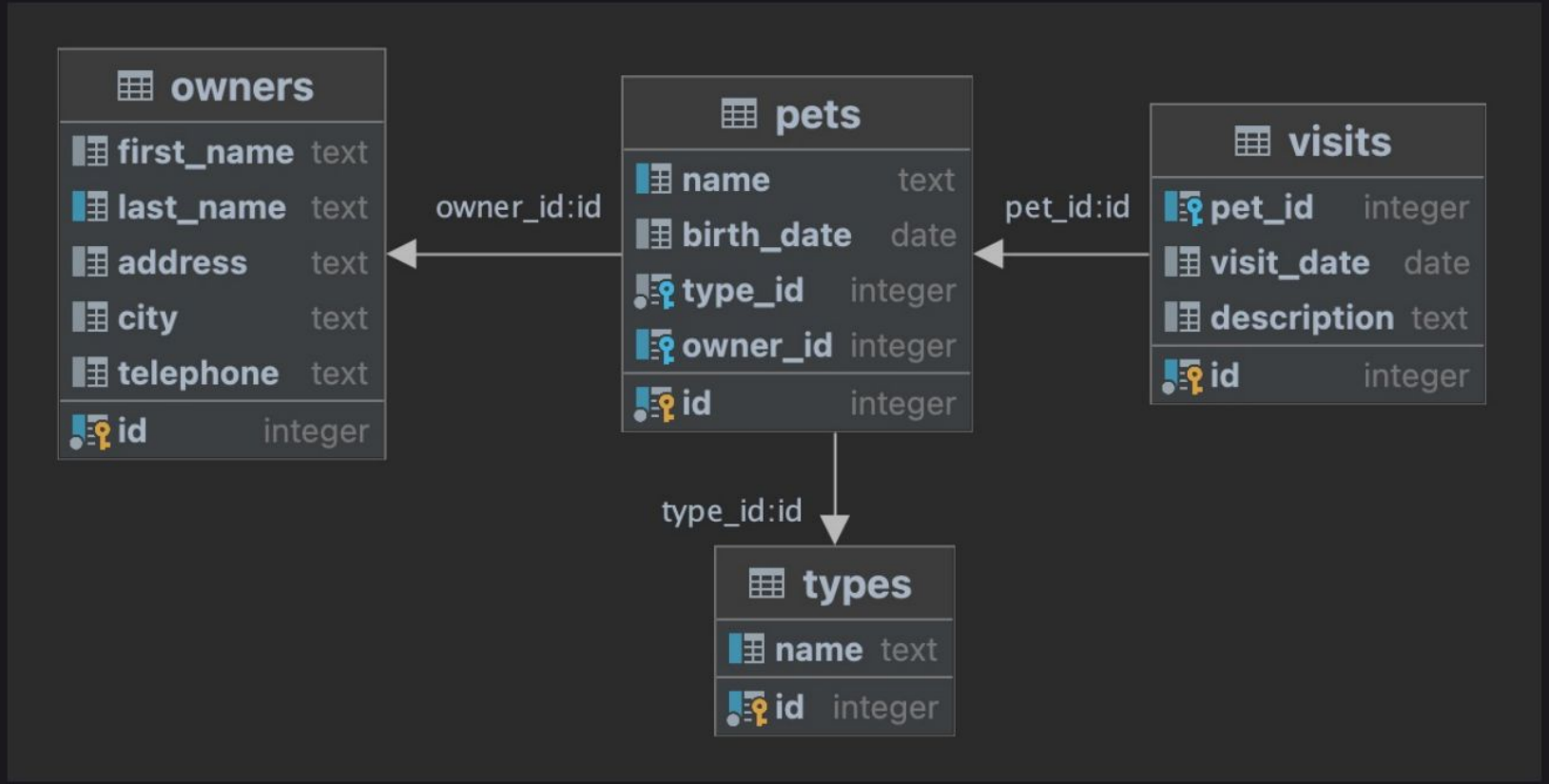
webfulx-jdbc

1. webfulx-jdbc
2. webfulx-jdbc-loom
3. webfulx-jdbc-coroutine

webflux-r2dbc

1. webflux-r2dbc
2. webflux-r2dbc-coroutine

Демонстрационная БД



Условия тестирования

- Идемпотентность
  Сложность постоянно повторяющейся тестируемой операции не должна зависеть от времени

- Равные внешние условия для всех тестов
  Кластер кубера, база данных, нагрузочный сценарий

- Тестируем именно наш сервис
  Важно не перегрузить базу данных

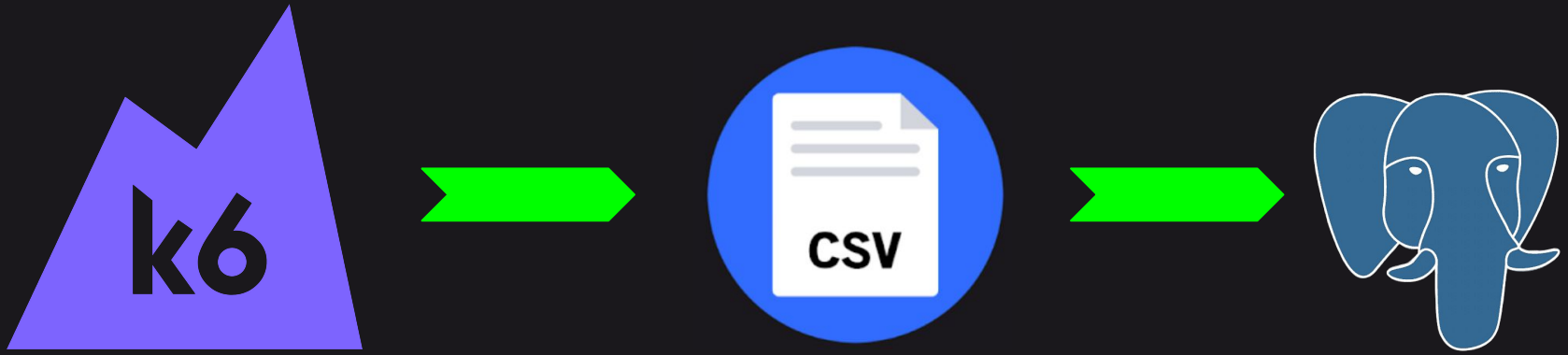
Тестовый сценарий №1

1. Create owner
2. Create visit
3. Create pet
4. Read owner
5. Read visit
6. Read pet
7. Find owners by last name
8. Update owner
9. Update visit
10. Update pet

Подготовка данных к анализу

12 минут тестирования дало CSV файл:

- Размер: 2.8 GB
- Объем: 24 млн. строк



Вариант 1: Ramping arrival rate

Ramping arrival rate. Позволяет понять, какой максимальный RPS система может обслужить до того, как закончатся ресурсы (CPU)

Протестированные реализации:

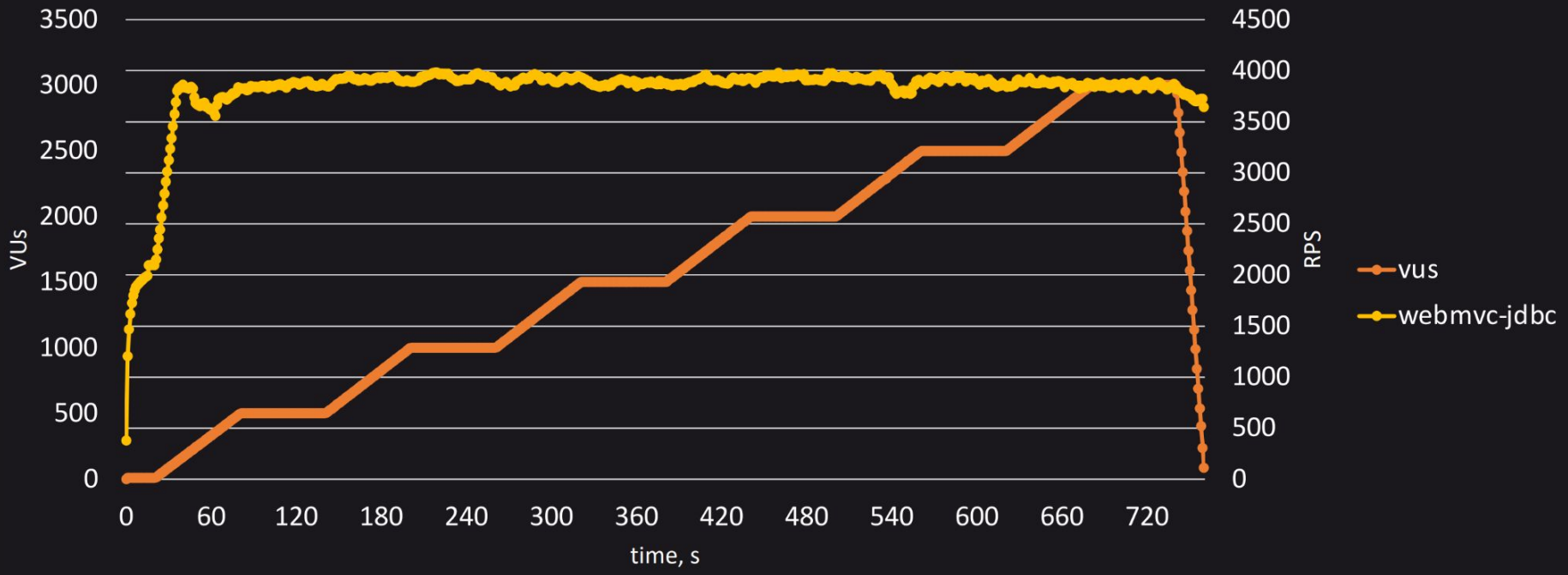
1. webmvc-jdbc
2. webmvc-jdbc-loom

3. webflux-jdbc-reactive, webflux-jdbc-coroutine
4. webflux-jdbc-loom

5. webflux-r2dbc-reactive, webflux-r2dbc-coroutine

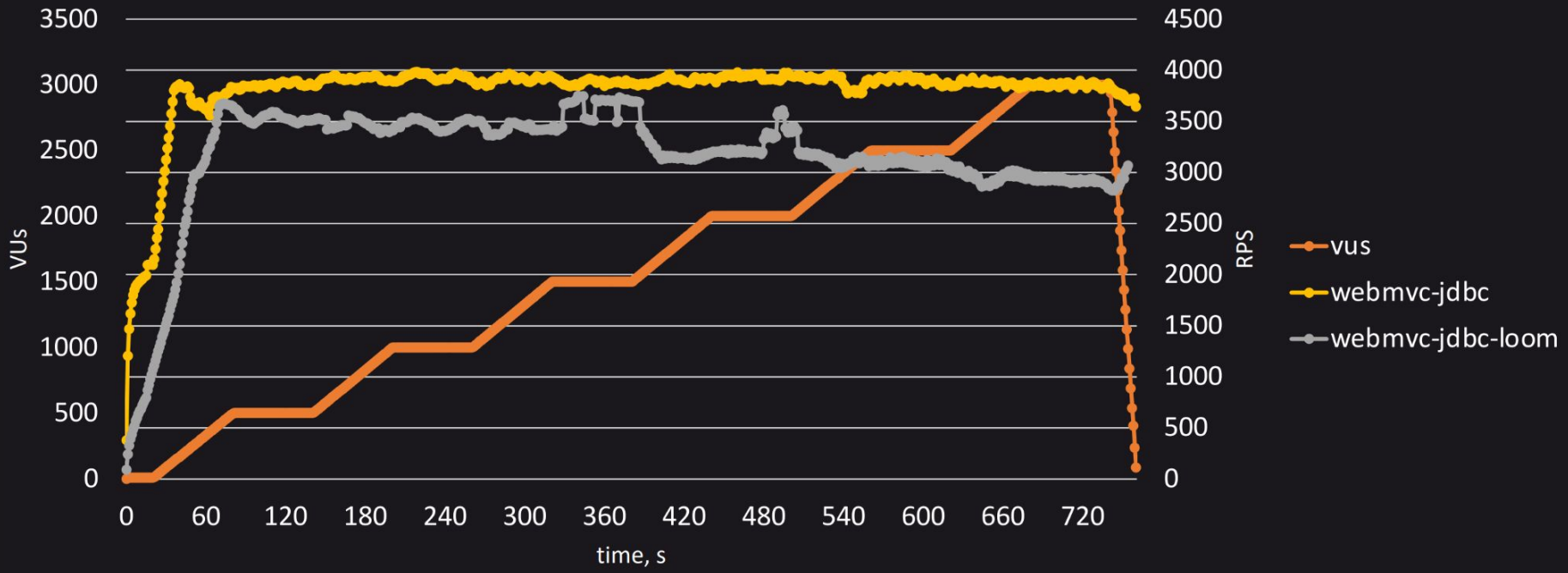
RPS

Рассматривается график webmvc-jdbc



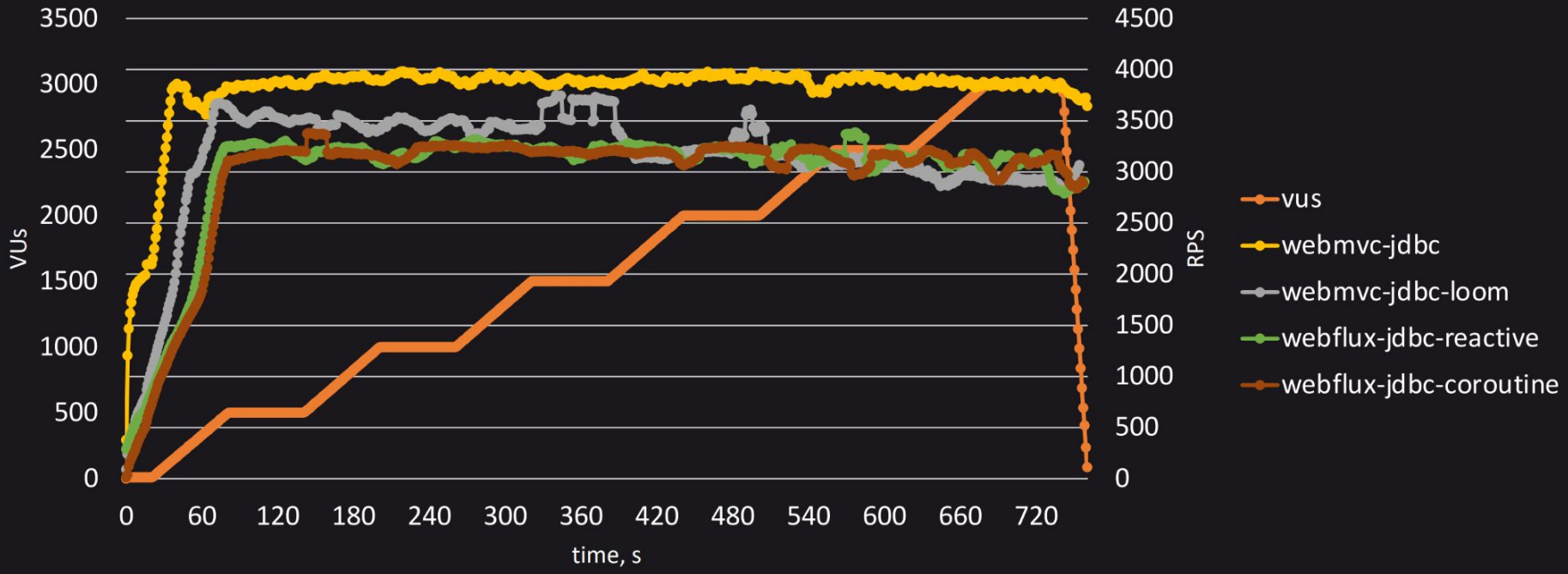
RPS

Рассматривается график webmvc-jdbc-loom



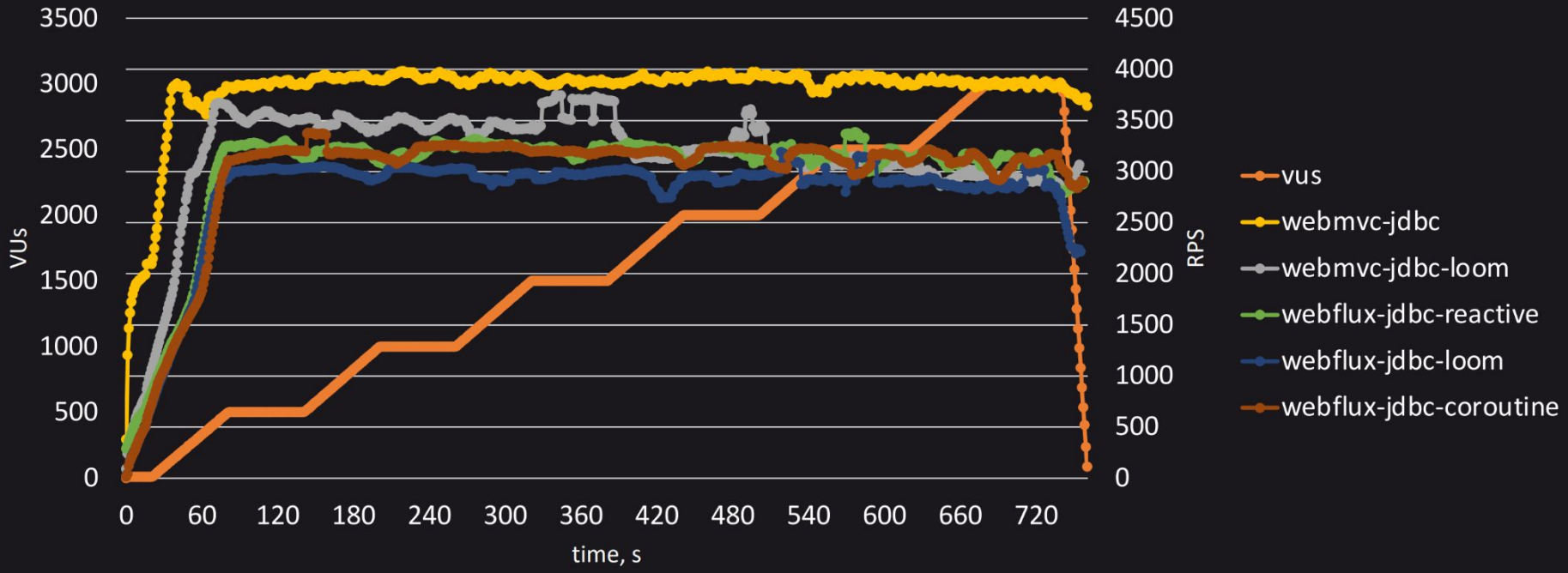
RPS

Рассматриваются графики webflux-jdbc-reactive, webflux-jdbc-coroutine



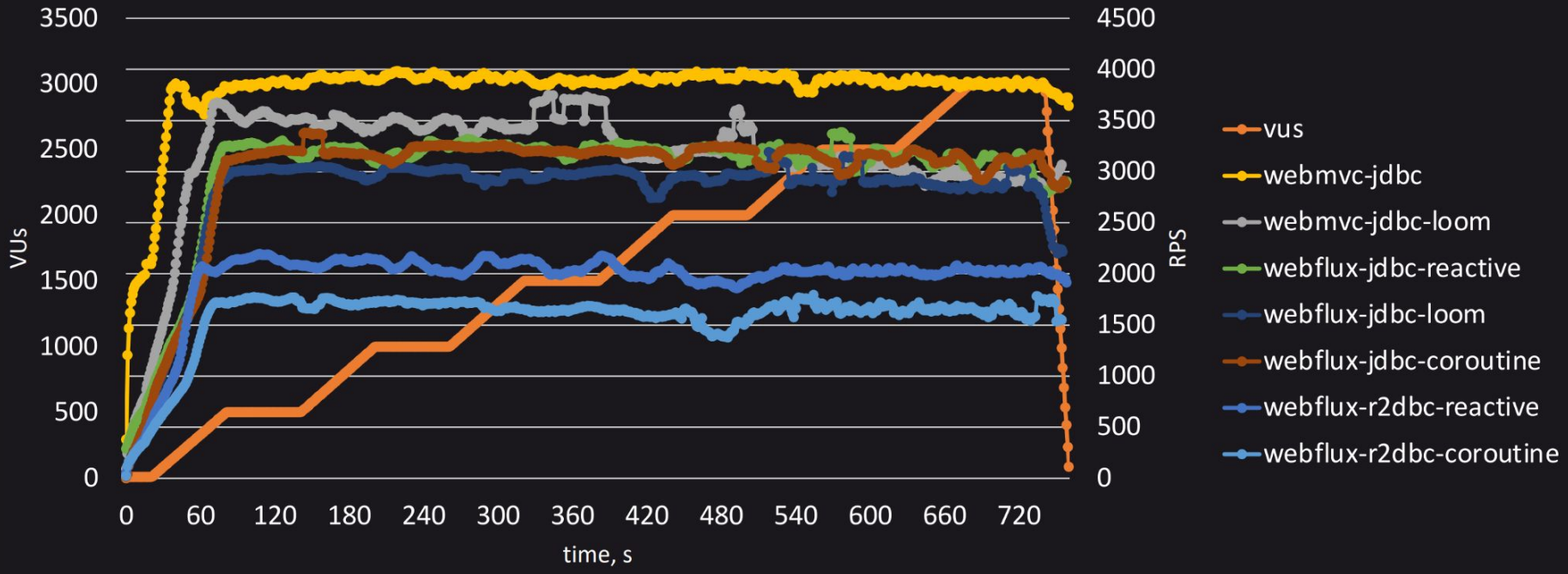
RPS

Рассматривается график webflux-jdbc-loom



RPS

Рассматриваются графики webflux-r2dbc-reactive, webflux-r2dbc-coroutine



Вариант 2: Ramping VUs

Ramping VUs - позволяет понять, какое максимальное количество VUs система одновременно может обслуживать при заданном таймауте на один запрос даже после того, как закончились ресурсы (CPU).

Протестированные реализации:

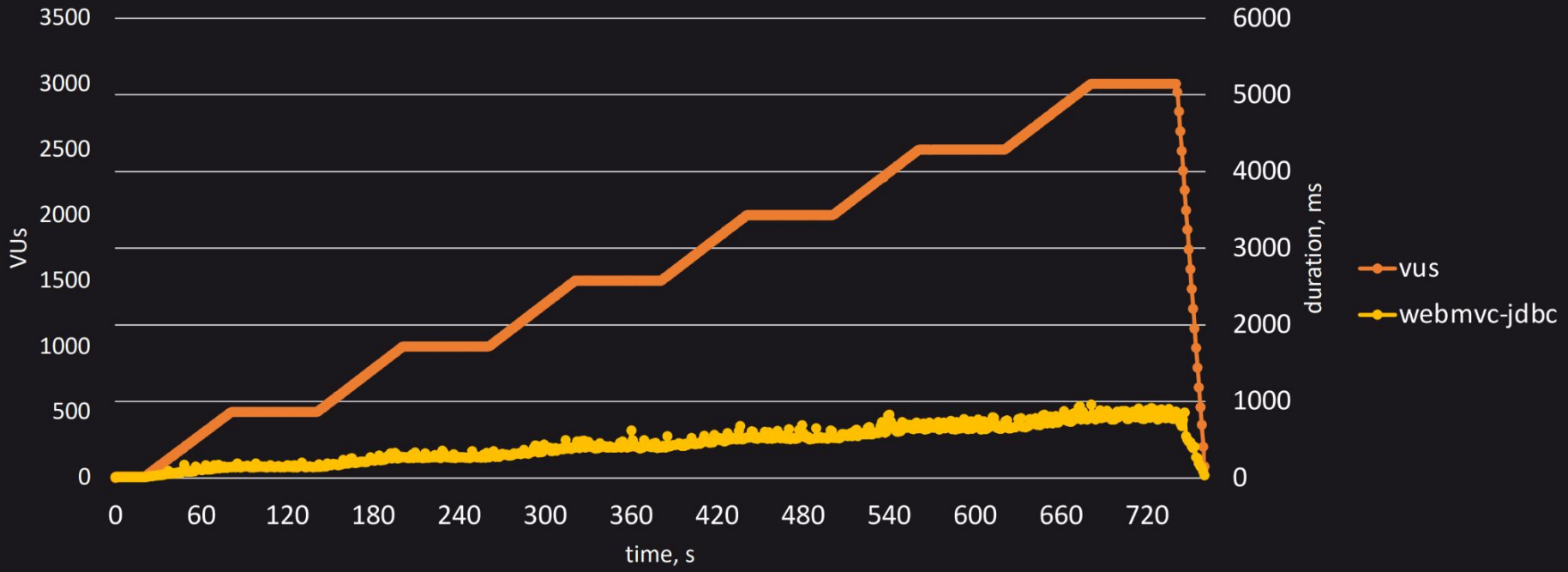
1. webmvc-jdbc
2. webmvc-jdbc-loom

3. webflux-jdbc-reactive, webflux-jdbc-coroutine
4. webflux-jdbc-loom

5. webflux-r2dbc-reactive, webflux-r2dbc-coroutine

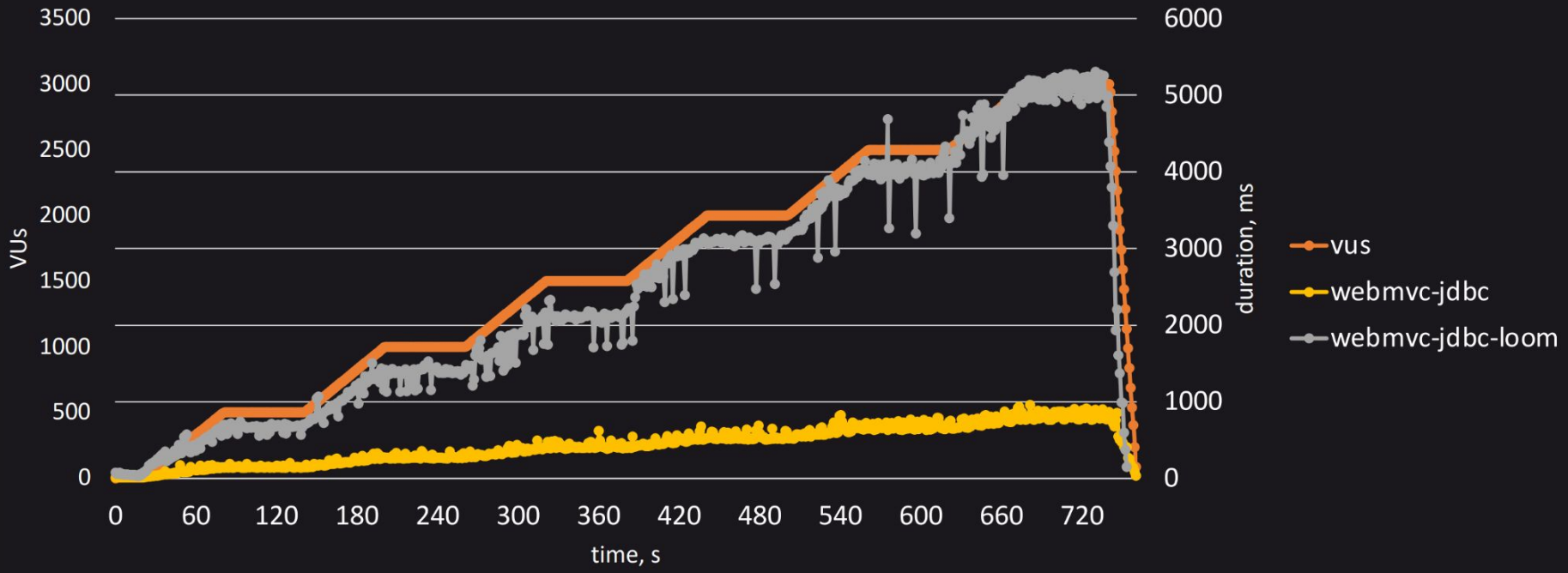
request duration: p(99)

Рассматривается график webmvc-jdbc



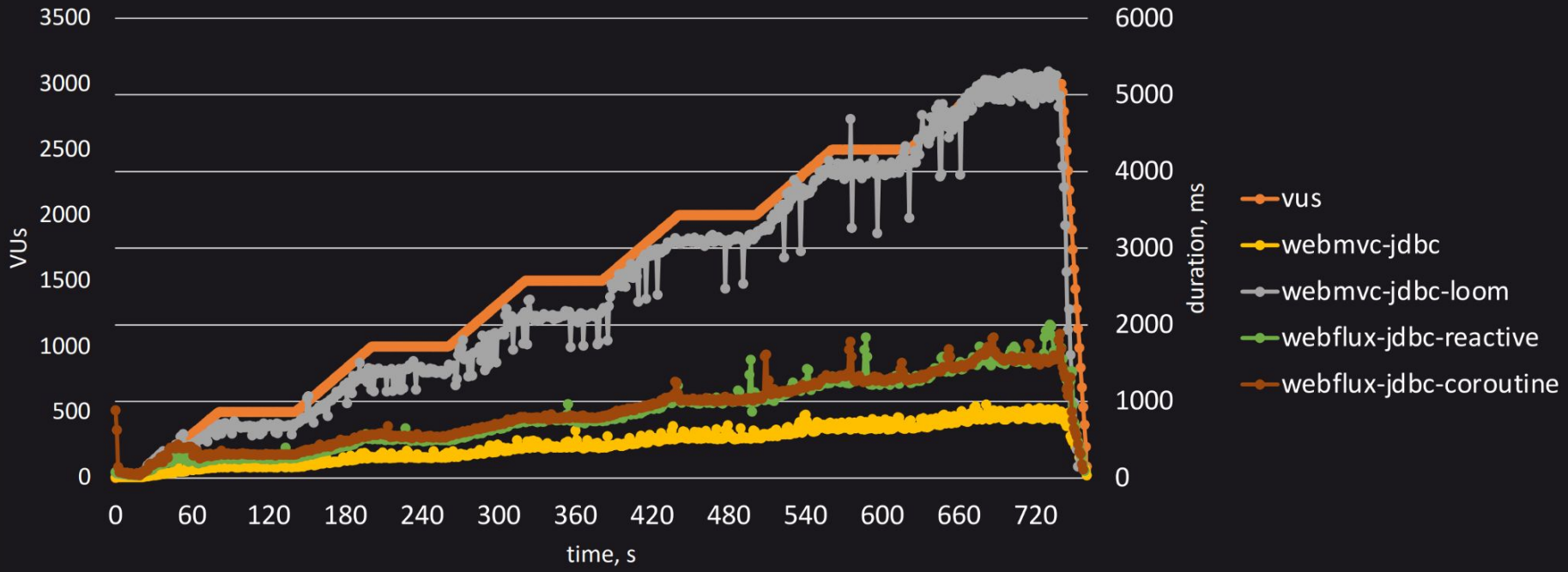
request duration: p(99)

Рассматривается график webmvc-jdbc-loom



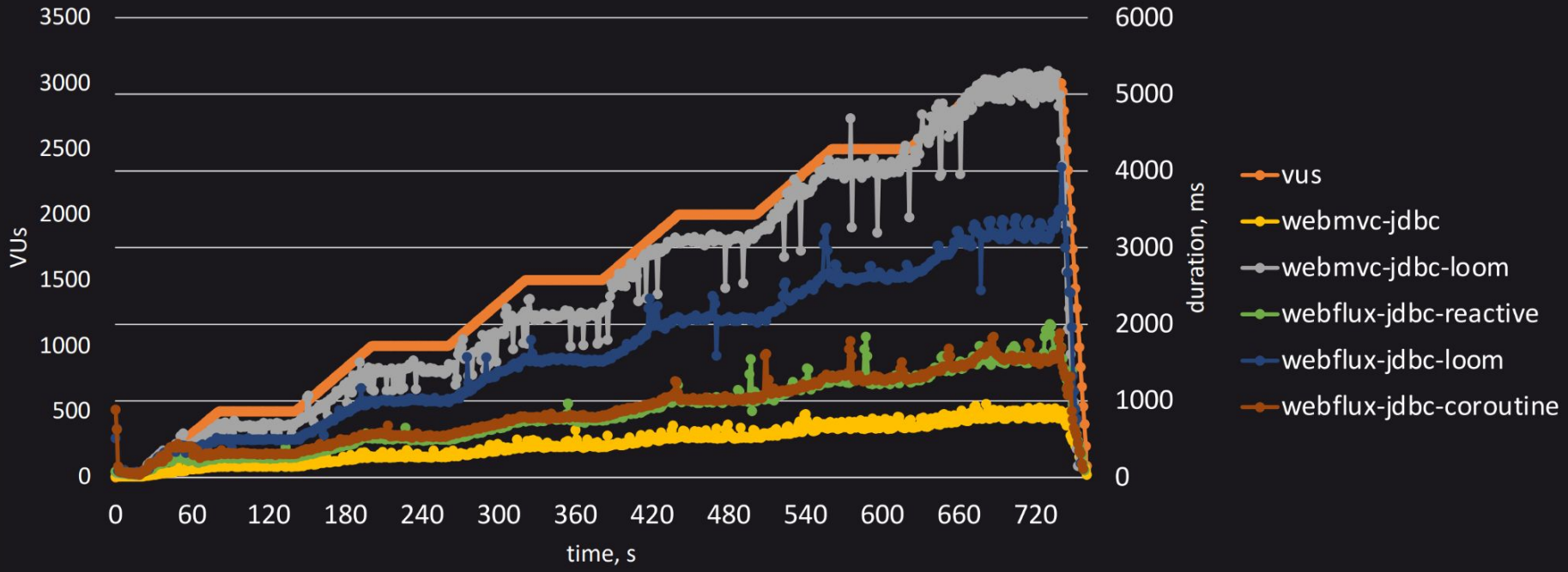
request duration: p(99)

Рассматриваются графики webflux-jdbc-reactive, webflux-jdbc-coroutine



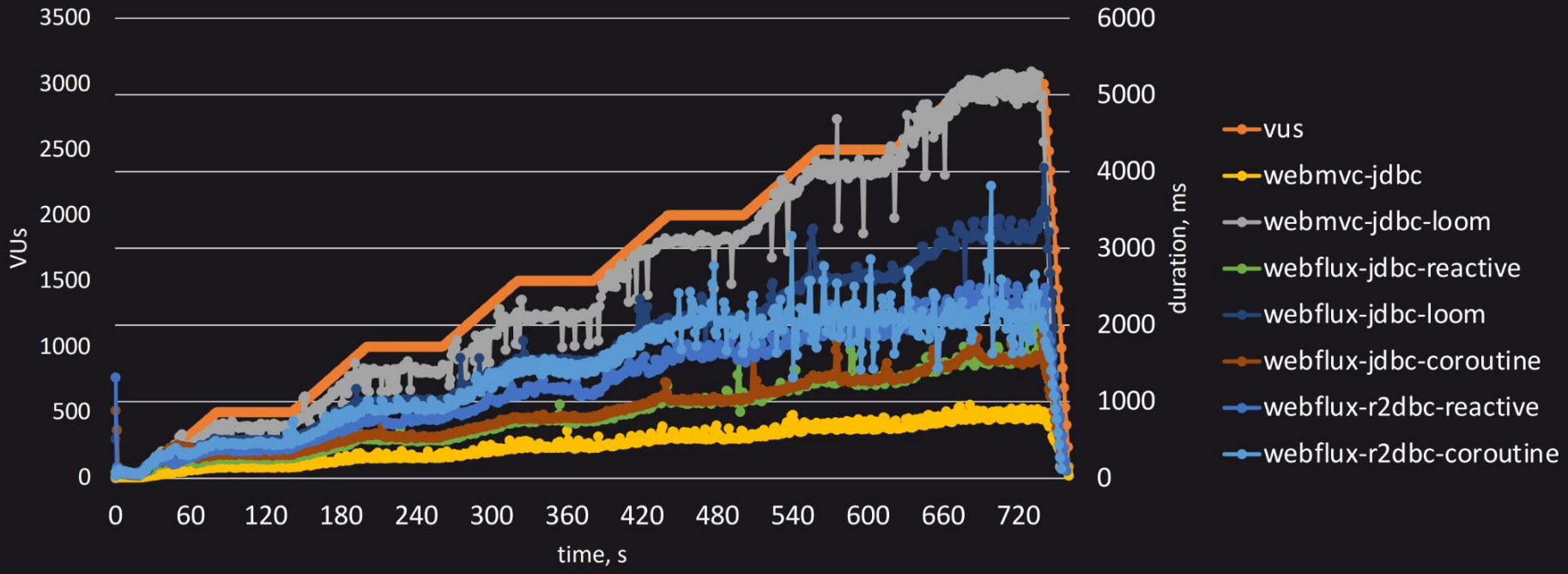
request duration: p(99)

Рассматривается график webflux-jdbc-loom



request duration: p(99)

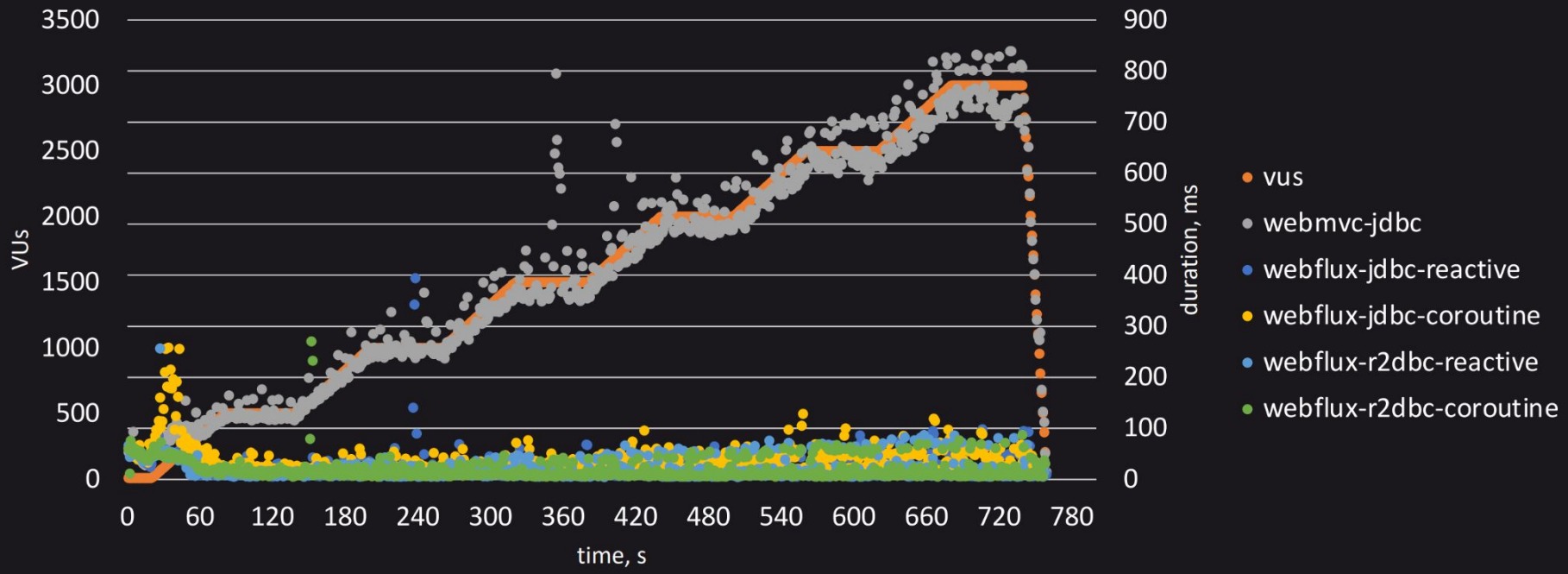
Рассматриваются графики webflux-r2dbc-reactive, webflux-r2dbc-coroutine



Тестовый сценарий №2

1. Create owner
2. Create visit
3. Create pet
4. Read owner
5. Read visit
6. Read pet
7. Find owners by last name
8. Update owner
9. Update visit
10. Update pet
11. Health check

Health check: request duration p(99)



Результаты тестирования

- При реальном CRUD-трафике ограничение переносится в СУБД: классический MVC + JDBC обеспечивает максимальный throughput, а реактивные или Loom-решения оправданы лишь при приоритетной задаче минимизации хвостовой задержки либо подтверждённом I/O-узком месте.

- Для лёгких запросов (health-check) решающим становится веб-слой; неблокирующий стек WebFlux + R2DBC снижает p(99) в 10 раз по сравнению со Spring MVC + JDBC и потому предпочтителен при высокой параллельности.

Выводы

- Разработан тестовый стенд
- Проведены эксперименты анализа производительности
- Сделаны выводы об эффективности исследуемых технологий
- Результаты работы опубликованы