Нагрузочные тесты backend CPS (Kotlin/Java)

# 1 Основные положения

## 1.1 Цели тестирования

Основной целью проведения нагрузочного тестирования новой версии IoT-платформы является оценка возможностей системы, сравнение с показателями предыдущей версии. Для определения производительности IoT-платформы планируется:
Определение максимальной производительности (количество запросов в секунду) системы, на существующей конфигурации.
Определение максимальной производительности (количество запросов в секунду) системы, на ограниченной и расширенной конфигрурации для оценки масштабируемости системы.
Проверка надежности IoT-платформы во время тестирования в течение длительного времени.
Выявление потенциально «узких» мест IoT-платформы.

## 2.1 Объект тестирования

Тестирование проводится на нескольких конфигурациях:

1) На одной виртуальной машине DigitalOcean в следующей конфигурации:

CPU: 4x2500 Mhz
RAM: 8 Gb
Storage: SSD 120 Gib
OS: CentOS 7.3

2) На кластере из 3 виртуальных машин аналогичной конфигурации
3) На кластере из 5 виртуальных машин аналогичной конфигурации

На каждой машине запущен только по 1 инстансу Бэкенда (в Docker-контейнере) и СУБД Cassandra. Приложения самостоятельно определяют необходимое количество потоков, необходимых для работы.
Для проведения тестов сформирована тестовая база данных на 1000 пользователей с несколькими устройствами и очередями в каждом пользовательском профиле.

## 2.2 Общие параметры и отступления от методики тестирования

Для проведения тестирования использовалось 2 профиля использования системы с несколькими сценариями поведения в каждом:

- Профиль устройства
- Профиль пользователя

Профиль устройства имитирует работу конечного устройства пользователя, все задачи, выполняемые в профиле ориентированы на запись данных в KV-хранилище

пользователя, либо в последовательности данных (временные ряды показателей снимаемых устройством). Данный профиль реализует запись целочисленных значений в базу CPS с различными интервалами 1-5-10 секунд.
Профиль пользователя имитирует работу пользователя системы в личном кабинете, большинство операций в данном профиле ориентировано на чтение данных из БД (просмотр графиков последовательностей по устройствам, просмотр триггеров и данных в базе и KV хранилище). Также в небольшом количестве пользователь может осуществлять операции записи в базу (создание и удаление триггеров и очередей). Сценарии использования внутри каждого профиля выбирается случайным образом.
Принципиальных отступлений от Методики тестирования не было. По аналогии с работой промышленной среды, запросы к IoT-платформе отправлялись без задержек с максимально возможной интенсивностью (ограничением интенсивности запросов была только производительность самой IoT-платформы).
В ходе тестирования было увеличено количество инстансов CPS-платформы, при увеличении были проведены дополнительные тестирования.
В среднем план тестирования имеет следующие показатели:
4,5% запросов приходится на чтение данных из БД
95,5% запросов за запись значений в базу (в т.ч. изменение данных)

### 2.3 Ограничения тестирования

Не было зафиксировано.

## 3 Выводы

### 3.1 Общие результаты

Максимальная производительность IoT-платформы на минимальной конфигурации (один сервер с приложением и БД) равна ~3000 обработанных запросов в секунду на кластер на текущей тестовой базе данных.
При увеличении количества серверов до 3, производительность системы выросла до ~4500 запросов в секунду (фактор репликации БД 3, т.е. на каждом сервере хранится целостная реплика БД).
При увеличении количества серверов до 5, производительность увеличилась до ~6500 запросов (фактор репликации также 3, при использовании более высокого фактора репликации возможно небольшое увеличение производительности и снижение нагрузки на сетевую инфраструктуру).
В среднем возможен прирост около 1500 запросов в секунду на каждые 2 дополнительные машины.
Критерием для определения максимальной производительности в данном случае является значительное увеличение времени отклика (свыше 1 секунды на запрос), являющимся недопустимым.
Необходимо отметить что при данной нагрузке также полностью утилизируются все аппаратные возможности серверов.
Длительный тест надежности не проводился, но по результатам краткосрочных тестов (1 час), можно ожидать стабильную работу системы при нагрузках близких к пиковым.

Также можно отметить, что тестирование проводилось на виртуальных серверах, что позволяет предположить несколько лучшую производительность при использовании серверов на “голом железе”, либо виртуальных серверов на собственной “железной базе”.

## 3.2 Производительность при 1 инстансе бэкенда

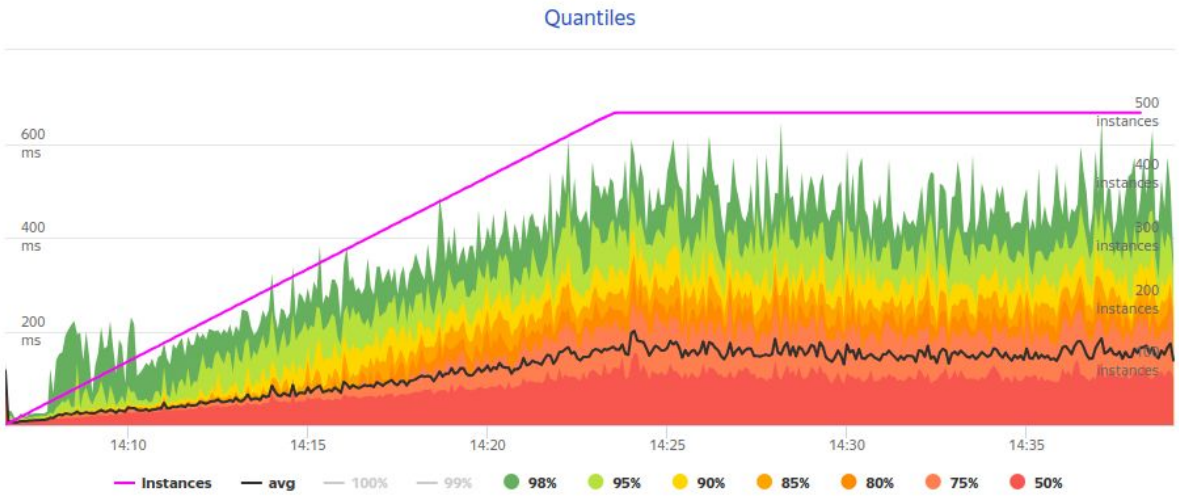


Рисунок 3.1 Процентили показателей задержки ответа на 1 инстансе при нагрузках близких к пиковым


Рисунок 3.2 Группированные задержки ответа на 1 инстансе при нагрузках близких к пиковым

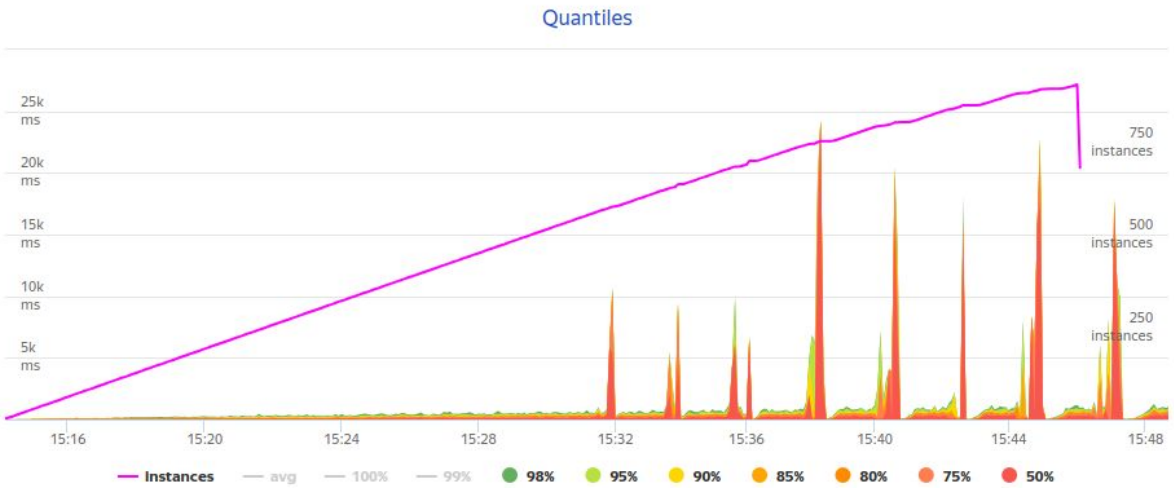


Рисунок 3.3 Процентили показателей задержки ответа на 1 инстансе при пиковых нагрузках

“Расческа” на графике - результат работы Garbage Collector’а сервера нагрузки. При увеличении количества потоков выше 500 на полную катушку включается G1GC который вешает процесс на периоды до 30с, отключить его не получится - память закончится моментально (к конце тестов утилизируется под 30Гб), толком скофигурировать его тоже никак. Если только запускать тесты на 10 Яве, в которой уже новый GC поумнее, но это реально долго.
Сам тестируемый сервер при этом отвечет снижением нагузки на процессор и увеличением времени задержки. Также возникают мистические сетевые ошибки, которые в принципе невозможны в данной ситуации - скорее всего артефакты ЯТанка, который в целом не задуман для работы с генератором нагрузки Jmeter, который мы используем, да еще и с такой нагрузкой.

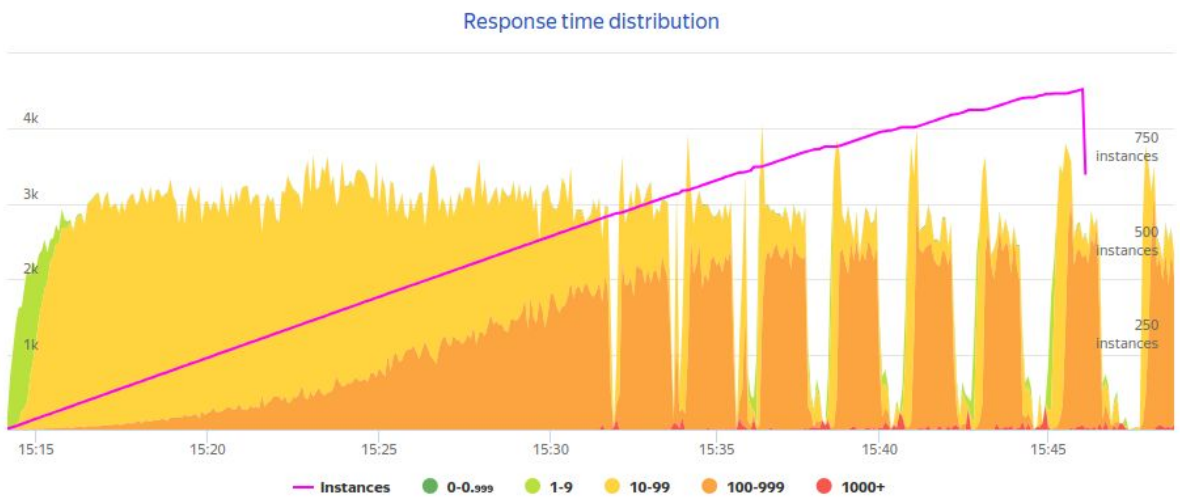


Рисунок 3.4 Группированное время задержки при пиковых нагрузках на 1 инстансе

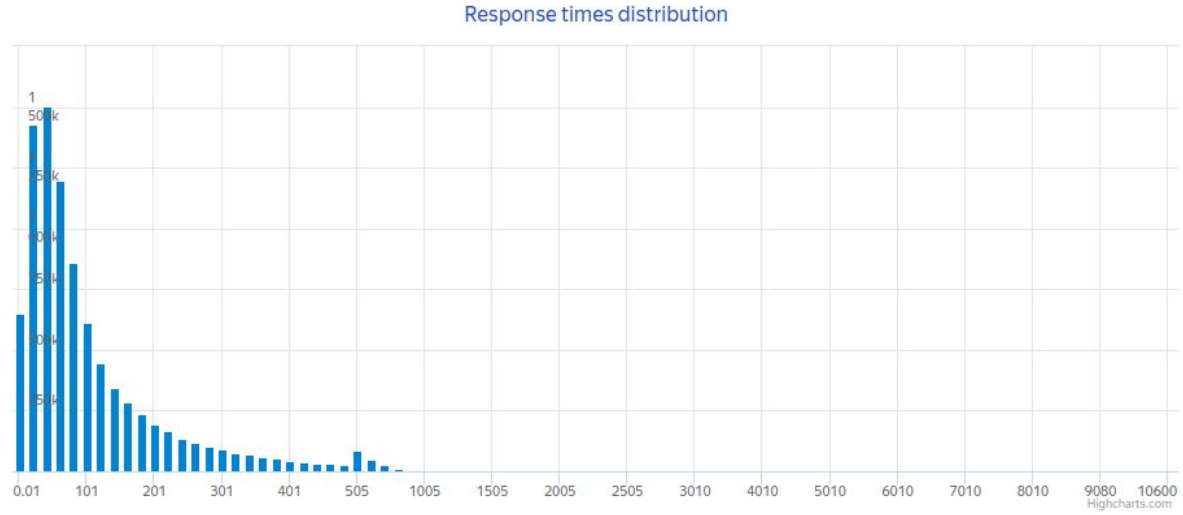


Рисунок 3.5 Распеделение времени отклика (наибольшее число результатов крутится вокруг значения 100мс)

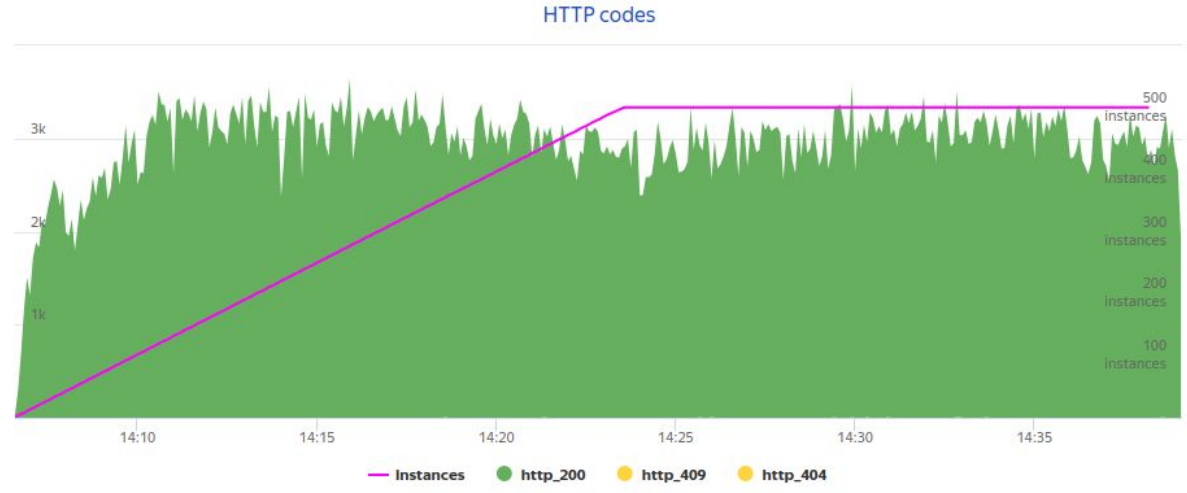


Рисунок 3.6 Количество запросов в секунду на 1 инстансе при нагрузках близких к пиковым

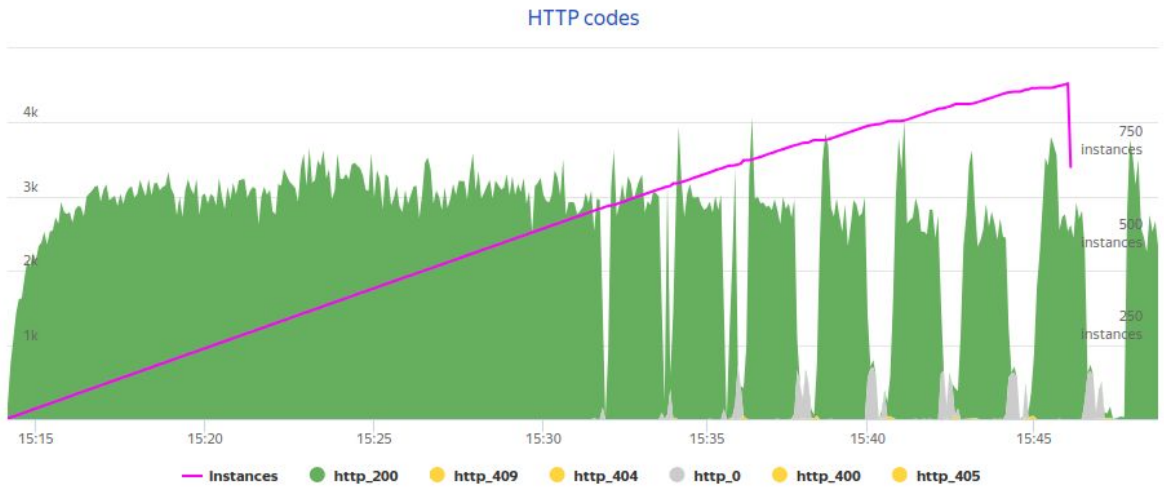


Рисунок 3.7  Количество запросов в секунду на 1 инстансе при пиковых нагрузках

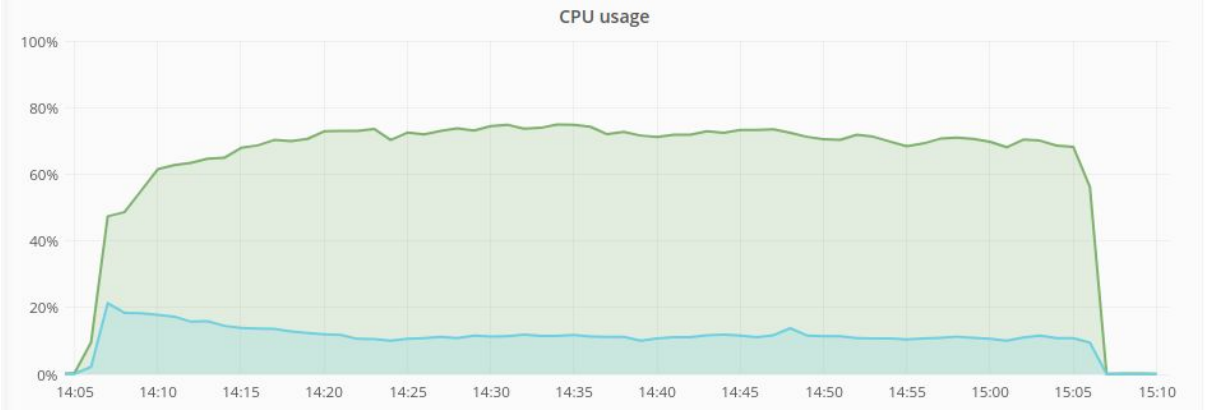


Рисунок 3.8 Нагрузка на процессор при нагрузках близких к пиковым зеленый-User (генерирует бекенд) синий-System (дисковый IO)

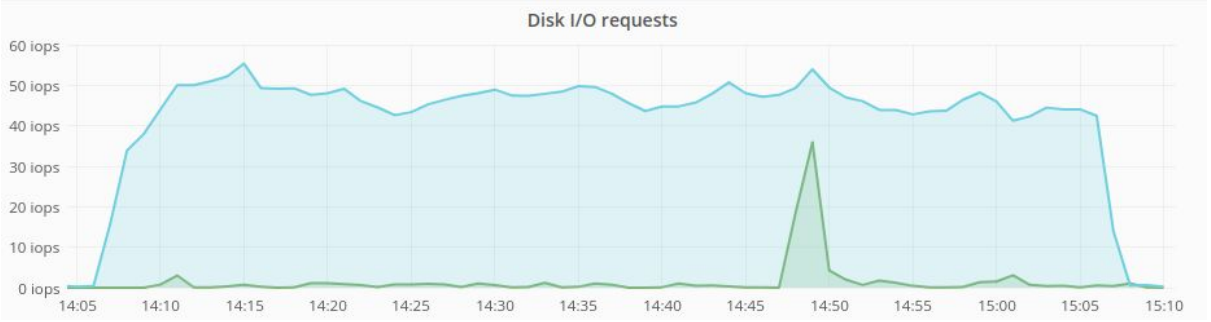


Рисунок 3.9 количество операций обращения к диску на нагрузках близких к пиковым

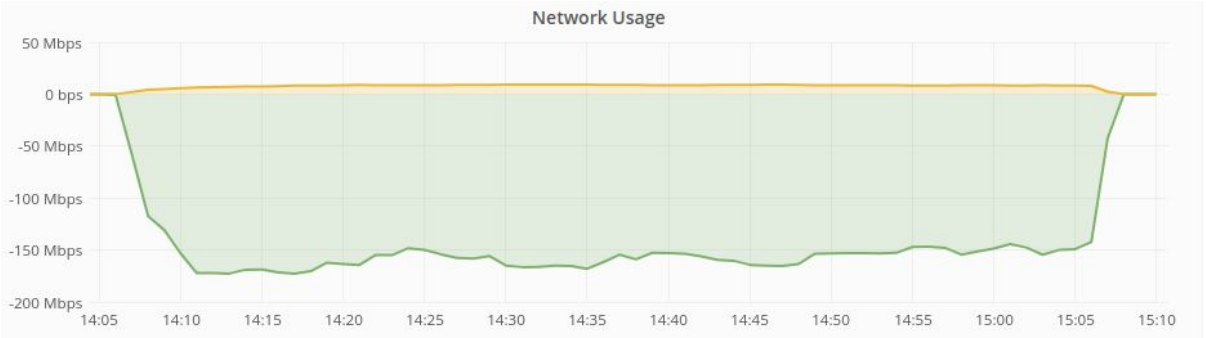


Рисунок 3.10 Нагрузка на сеть зеленый-in желтый-out

## 3.3 Производительность при 3 инстансах бэкенда

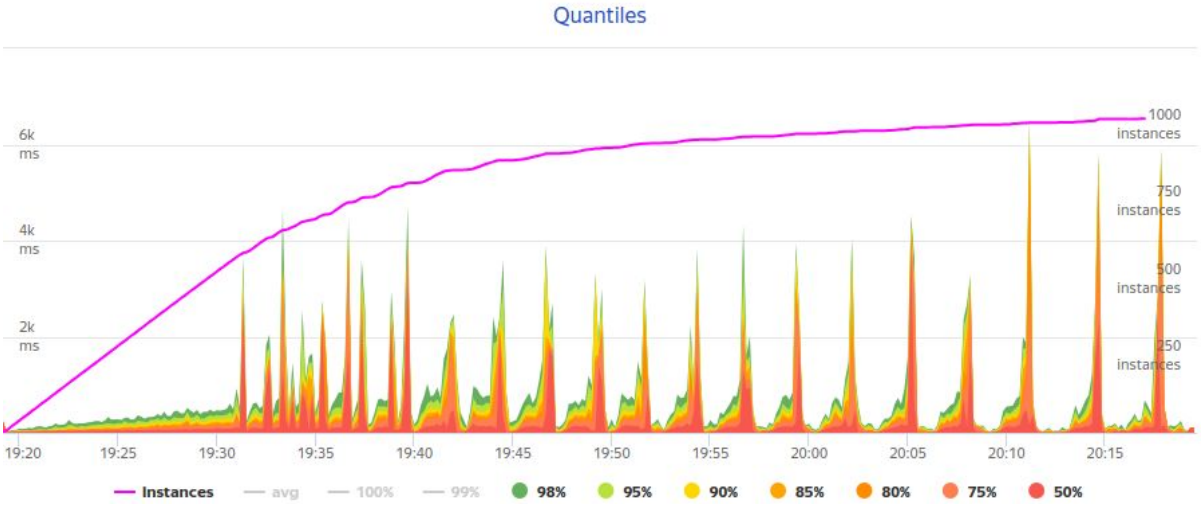


Рисунок 3.11 Процентили показателей задержки ответа


Рисунок 3.12 Группированное время задержки

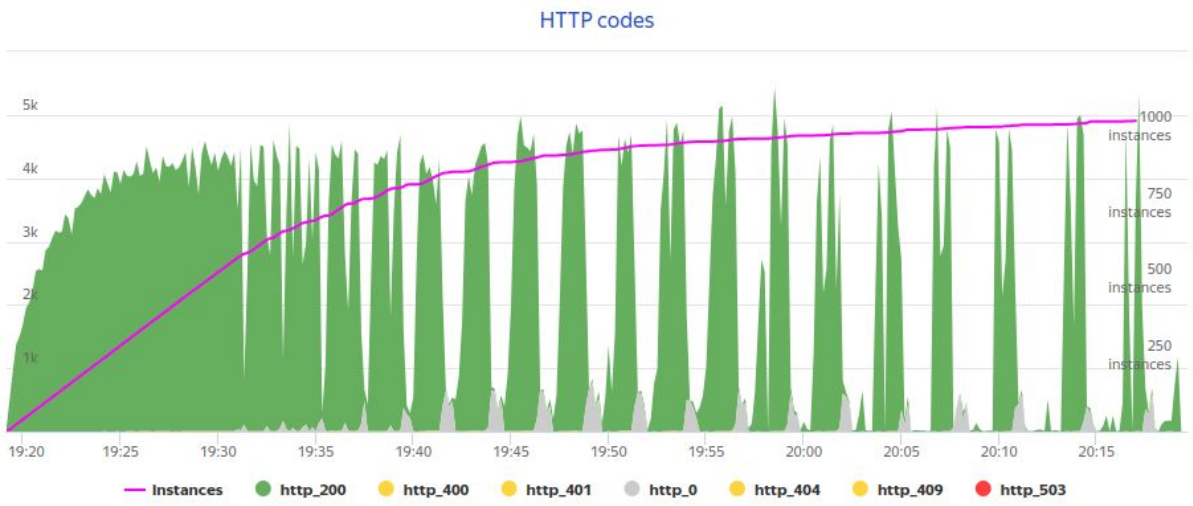


Рисунок 3.13 Коды ответов/Запросы в секунду http_0 - ошибки сети (следующий график)

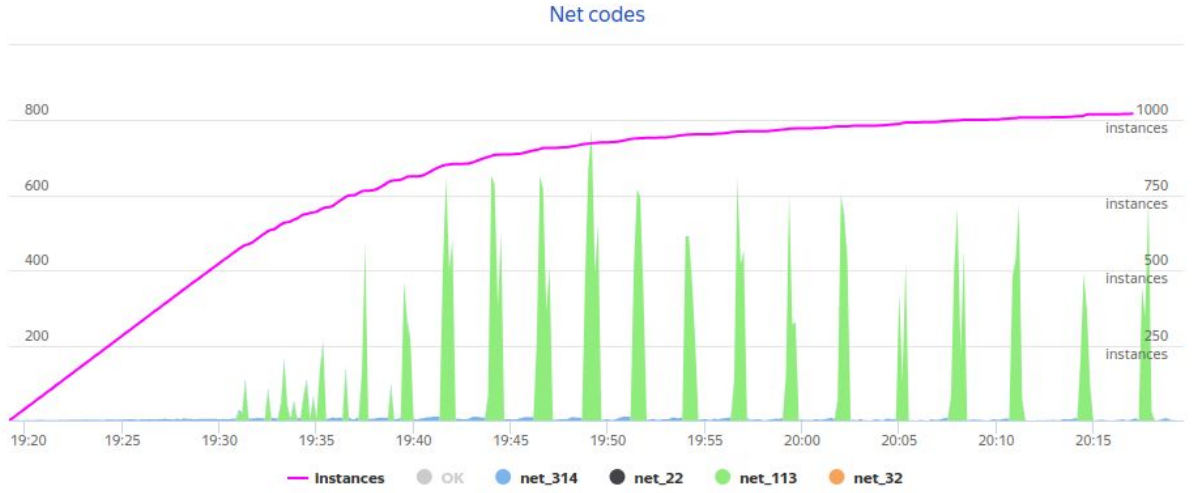


Рисунок 3.14 Коды ошибок на “расческе” в периоды провала нагрузки 113-No route to host 314-N/A (нет ответа)

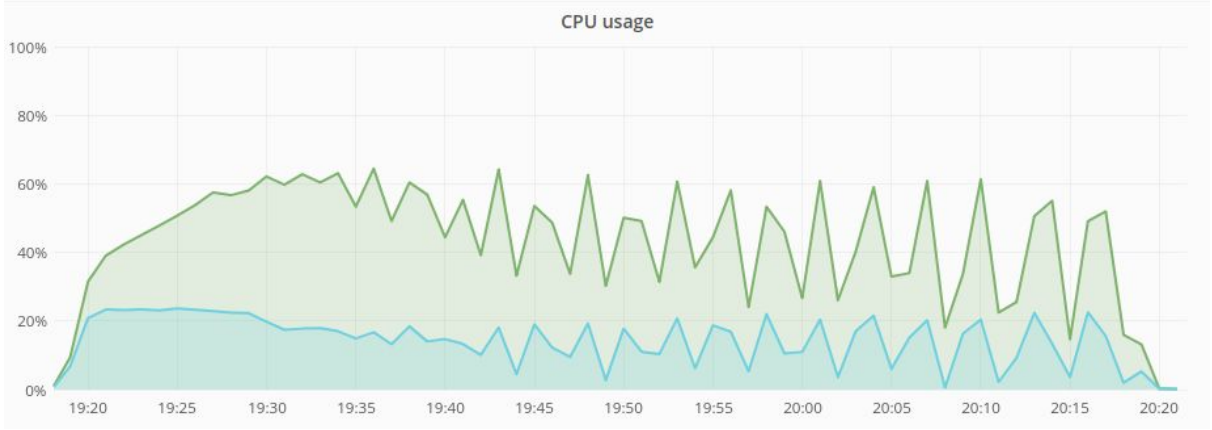


Рисунок 3.15 Нагрузка на процессор

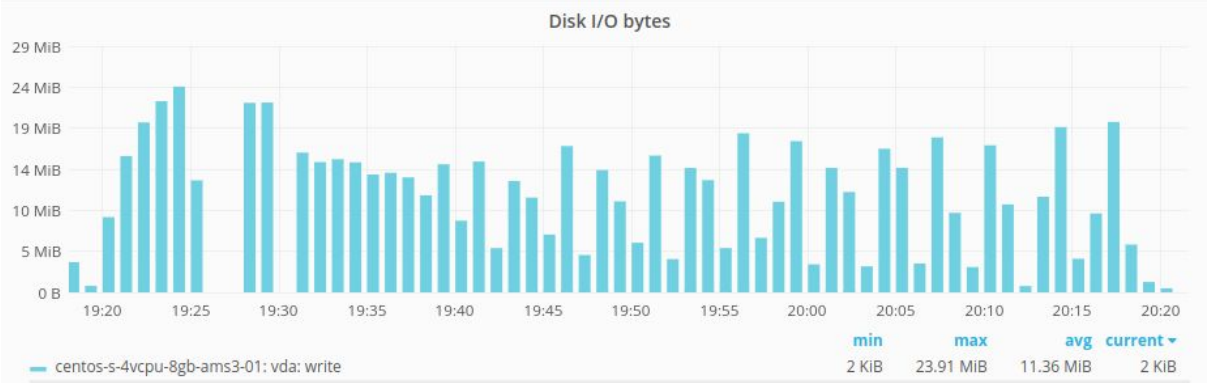


Рисунок 3.16 Запись на диск в Мб.

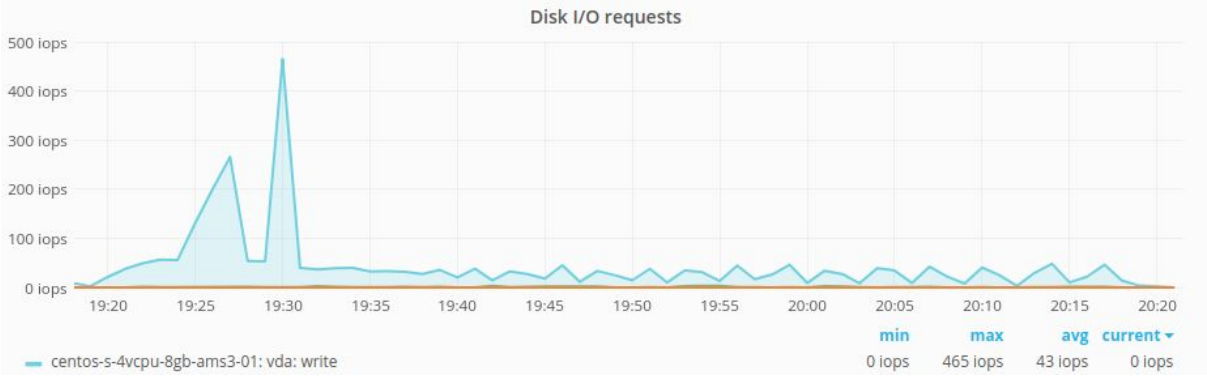


Рисунок 3.17 Запросы к диску

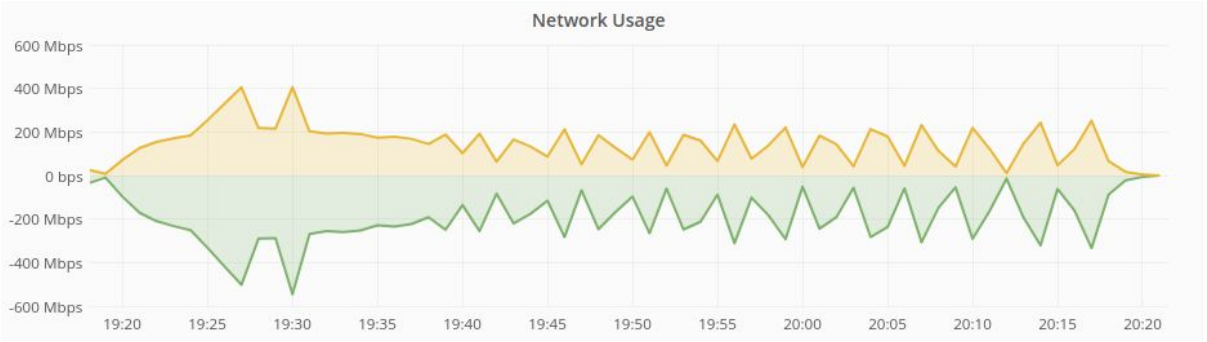


Рисунок 3.18 Нагрузка на сеть

## 3.4 Производительность при 5 инстансах бэкенда

Разница с результатами, полученными при работе 3х инстансов отсутствует, разве что несколько увеличенная производительность

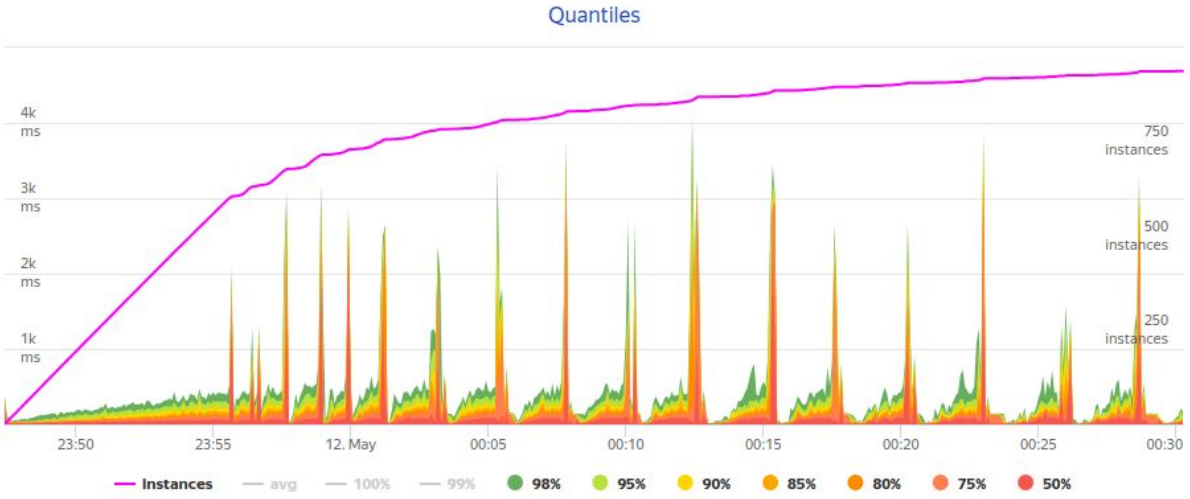


Рисунок 3.17 Процентили показателей задержки ответа

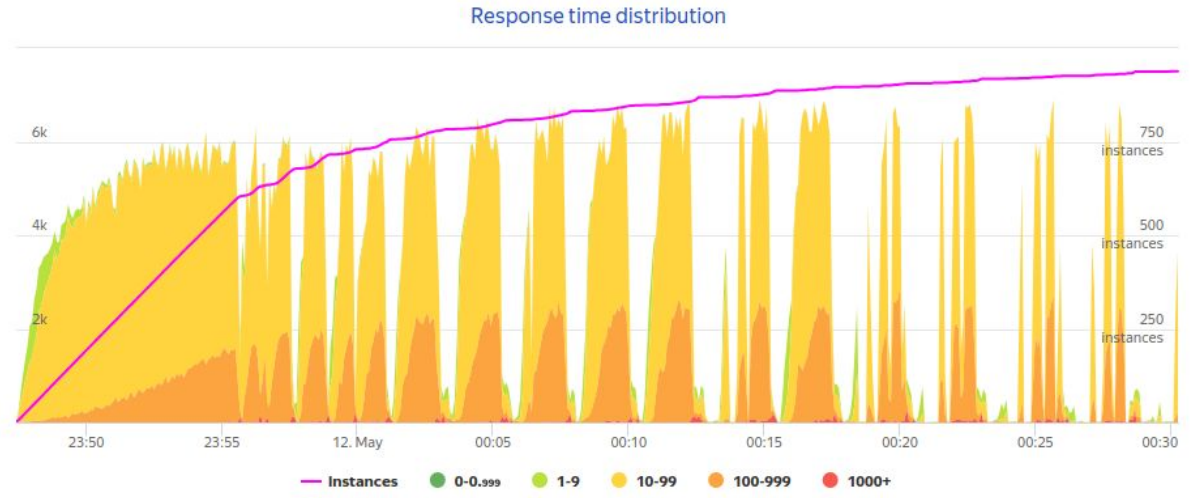


Рисунок 3.18 Группированное время задержки

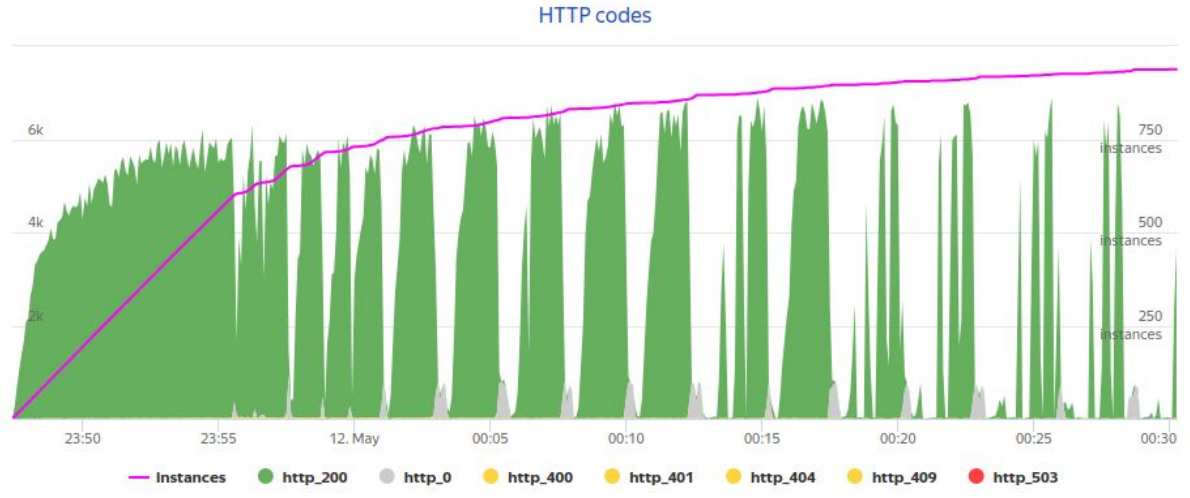


Рисунок 3.19  Коды ответов/Запросы в секунду

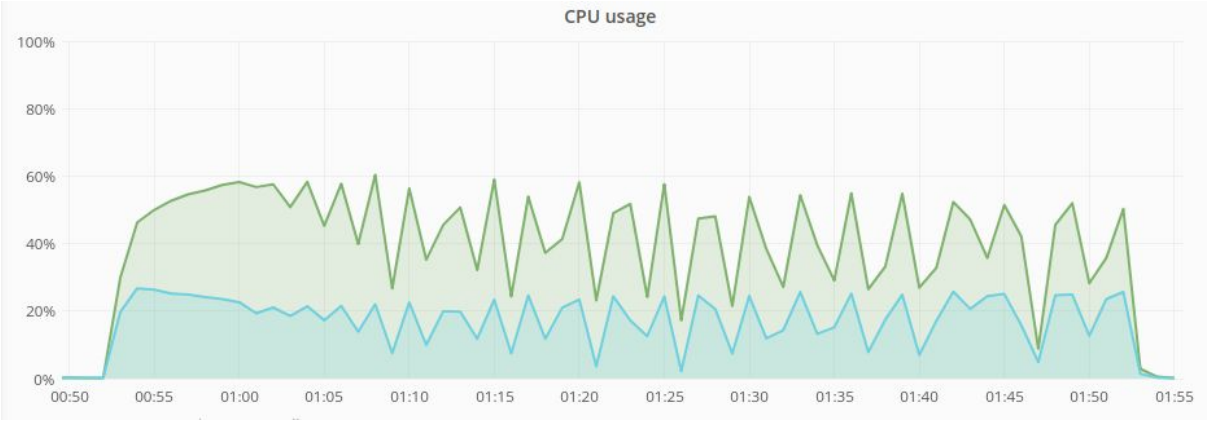


Рисунок 3.20 Нагрузка на процессор

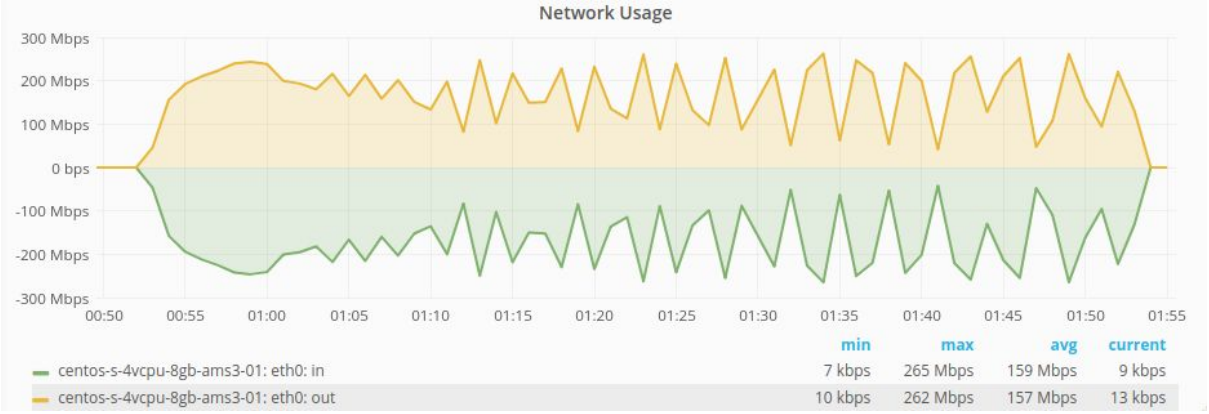


Рисунок 3.21 Нагрузка на сеть