



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Выпускная квалификационная работа бакалавра

«Разработка программного обеспечения для сравнения производительности многопоточной и многопоточно- асинхронной серверных архитектур»

Выполнил:

Студент группы М30-419Бк-19

Колодяжный М.А.

Руководитель:

к.т.н., доцент каф.304

Дмитриева Е.А.

Цель

Целью разработки данного программного обеспечения является создание платформы для сравнения производительности и анализа преимуществ и недостатков относительно друг друга для многопоточной и многопоточно-асинхронной серверных архитектур.

Задачи

- Изучение библиотеки BOOST
- Реализация многопоточной и многопоточно-асинхронной архитектур серверов
- Реализация клиентского приложения и имитатора клиентской нагрузки
- Проведение тестирования с логгированием ключевых параметров
- Анализ полученных данных с графическим отображением результатов

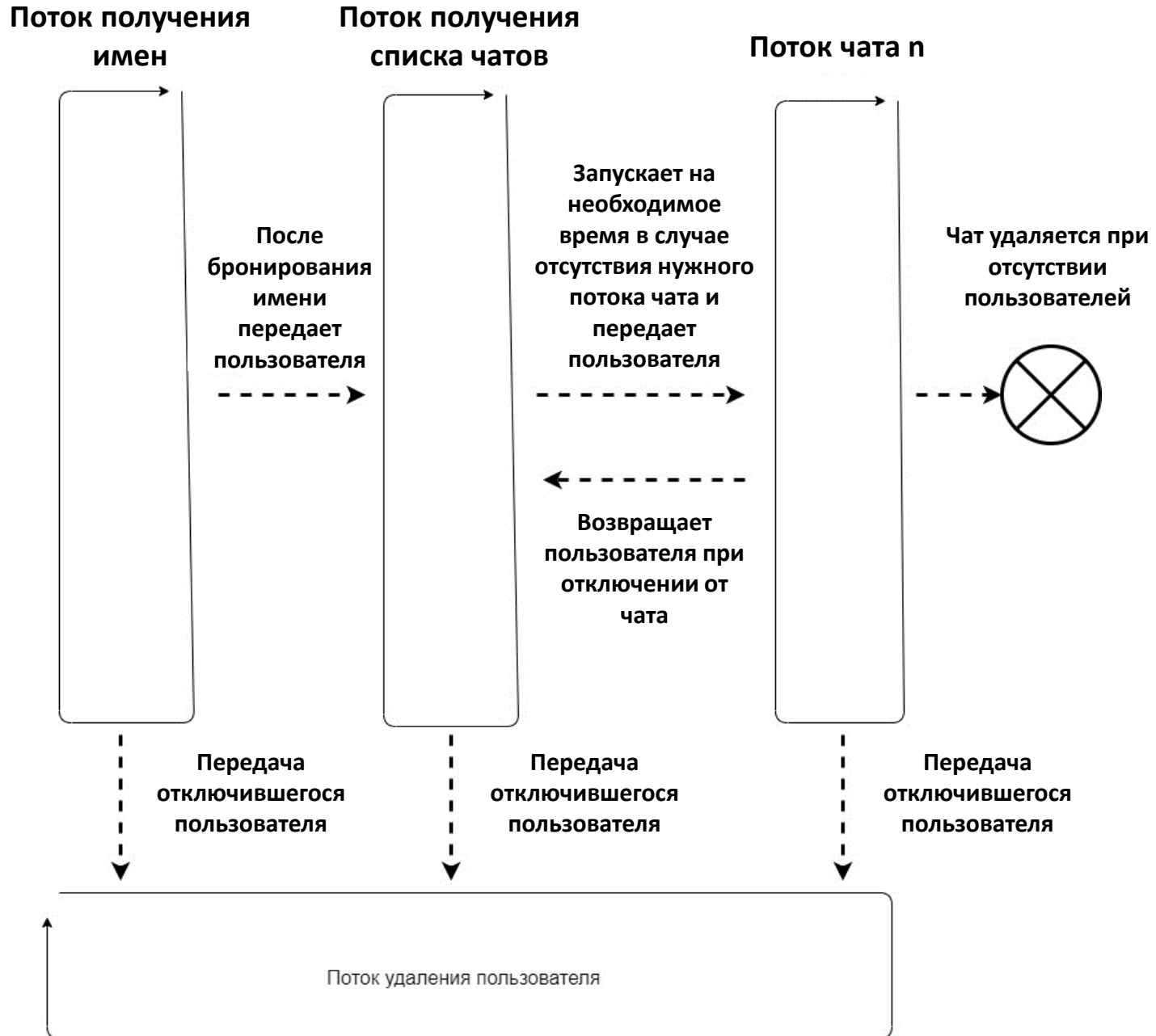
Инструменты, технологии, особенности изучения языка программирования

Изучены и применены на практике:

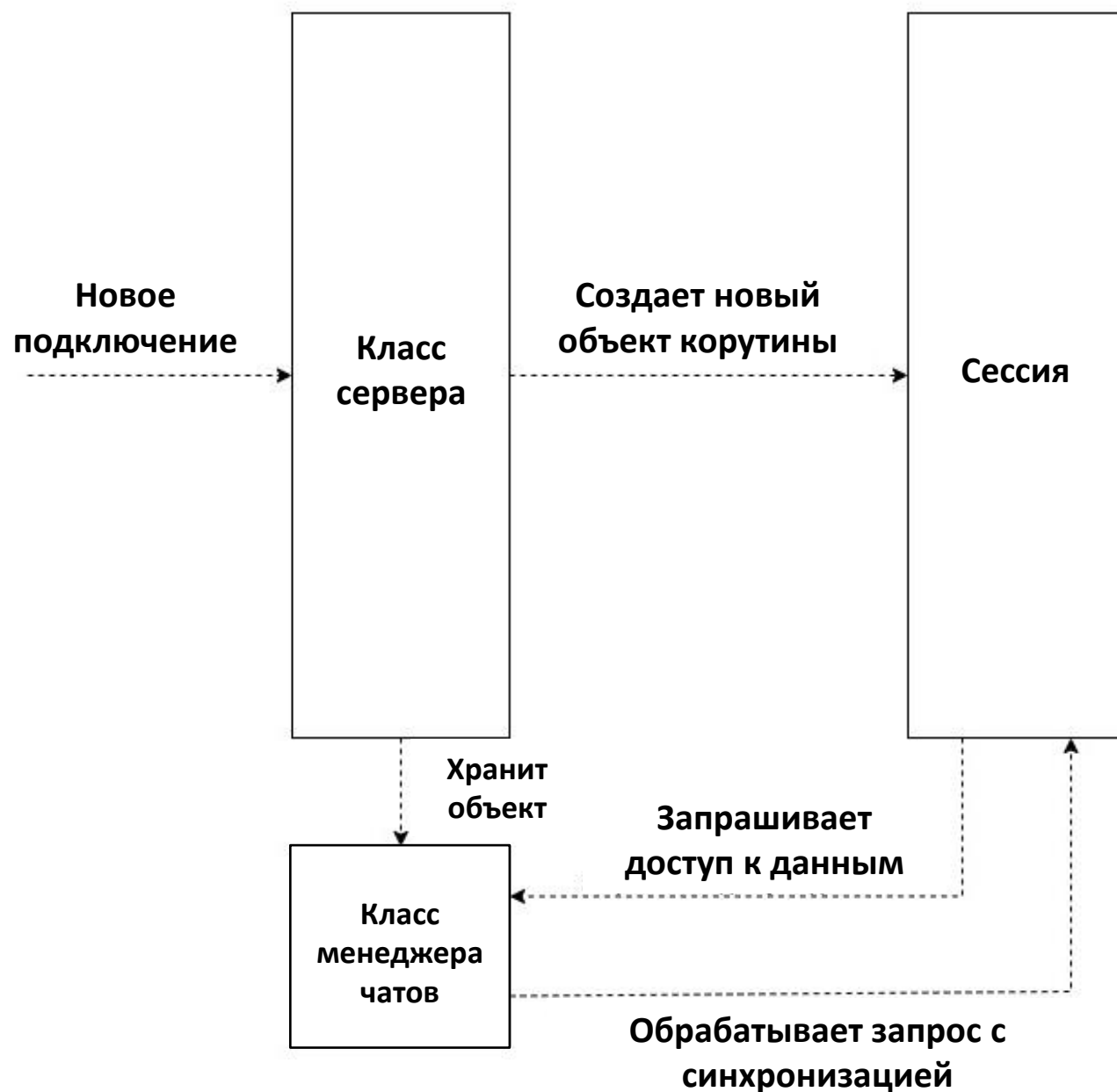
- Библиотеки BOOST – собрание библиотек классов, использующих функциональность языка C++ и предоставляющих удобный кроссплатформенный высокоуровневый интерфейс для лаконичного кодирования различных повседневных подзадач программирования
- QT – фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++
- Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений
- Grafana – свободная программная система визуализации данных, ориентированная на данные систем ИТ-мониторинга. Реализована как веб-приложение в стиле «приборных панелей» с диаграммами, графиками, таблицами, предупреждениями.

Для разработки применялся C++, при отладке программного продукта отдельное внимание уделялось управлению памятью.

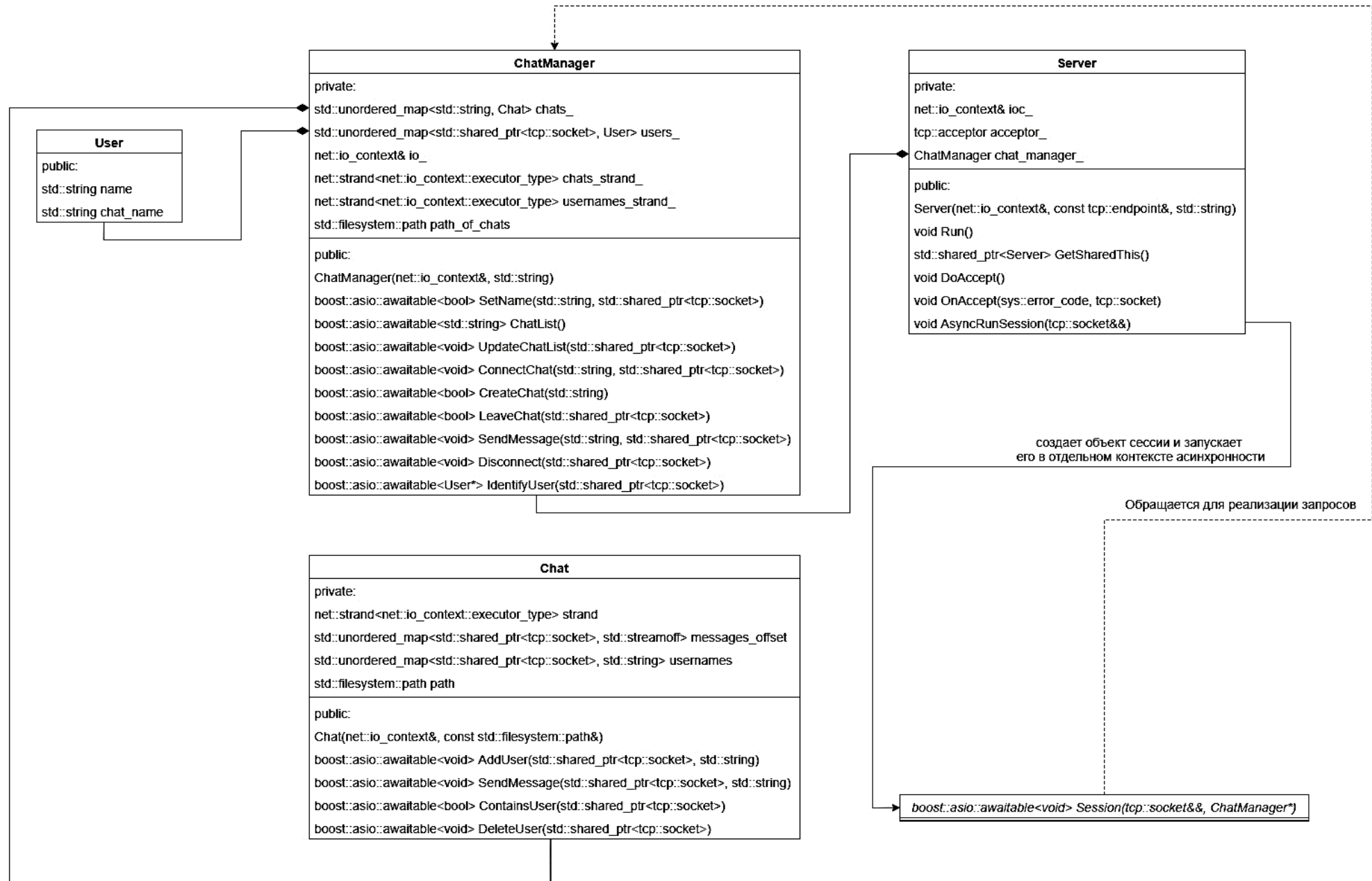
Многопоточный сервер



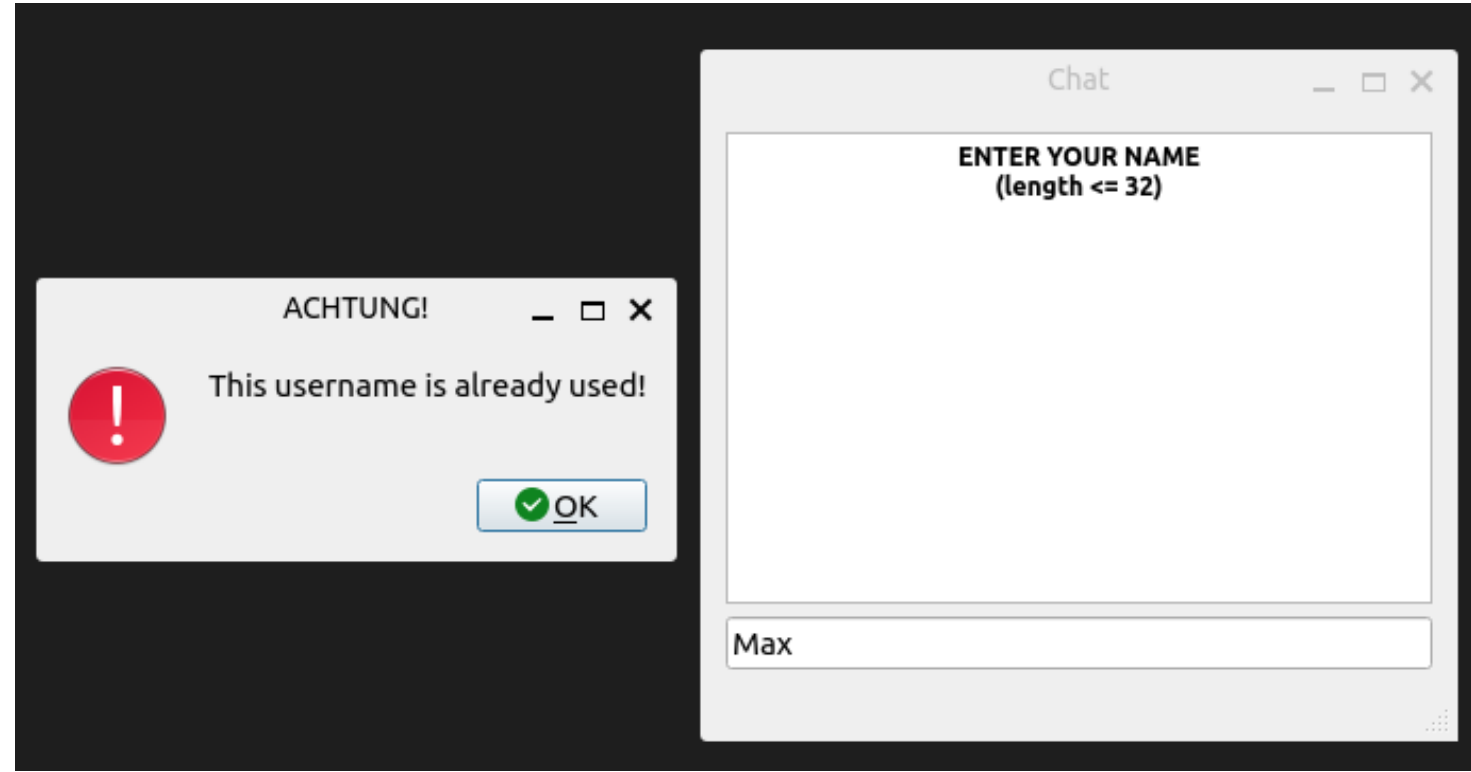
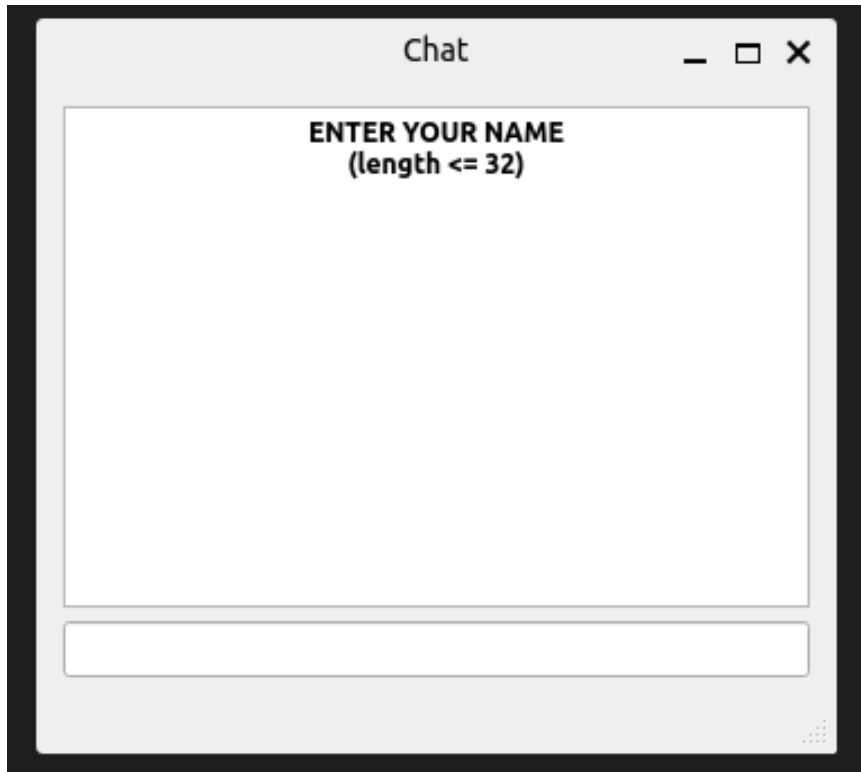
Многопоточно-асинхронный сервер



Многопоточно-асинхронный сервер: диаграмма классов

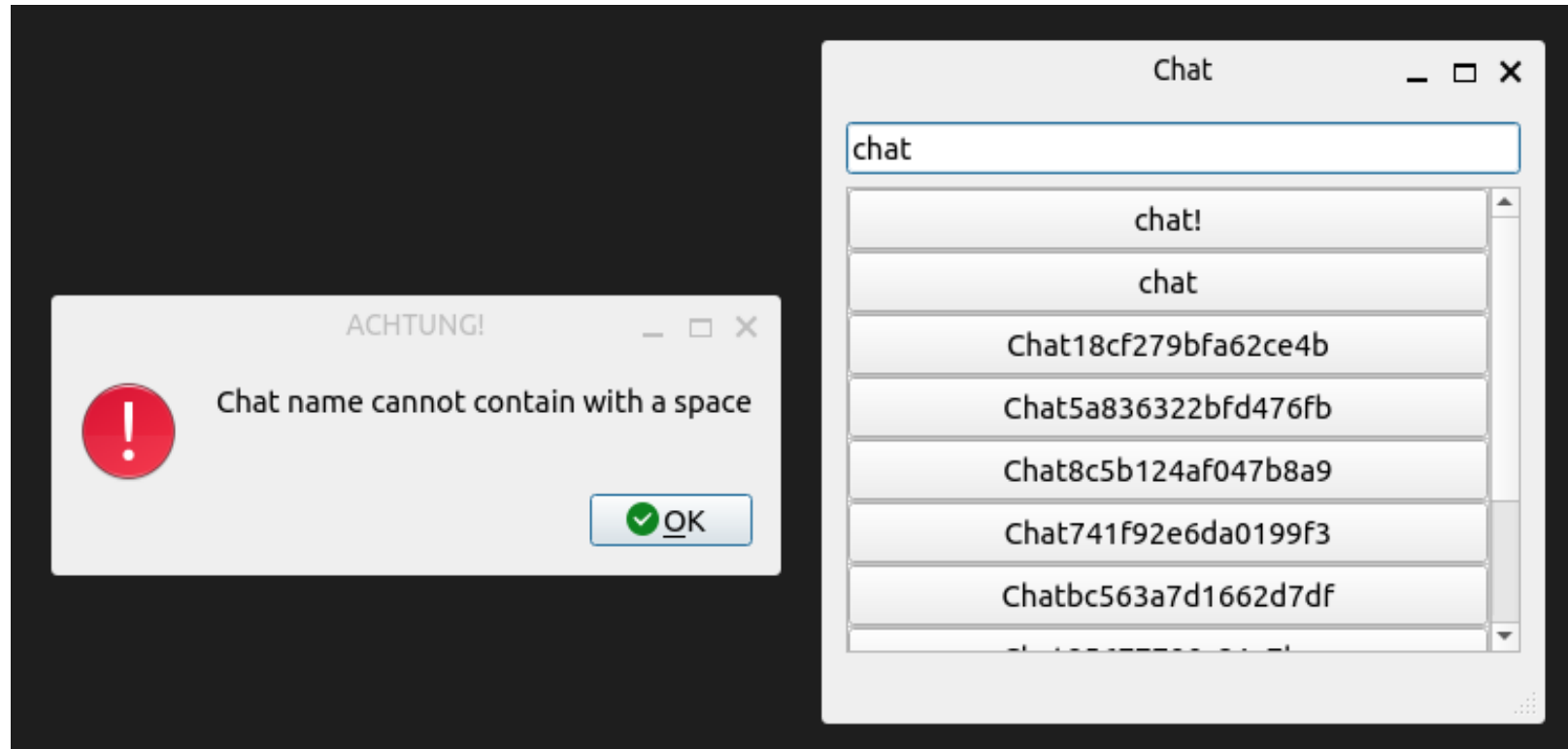
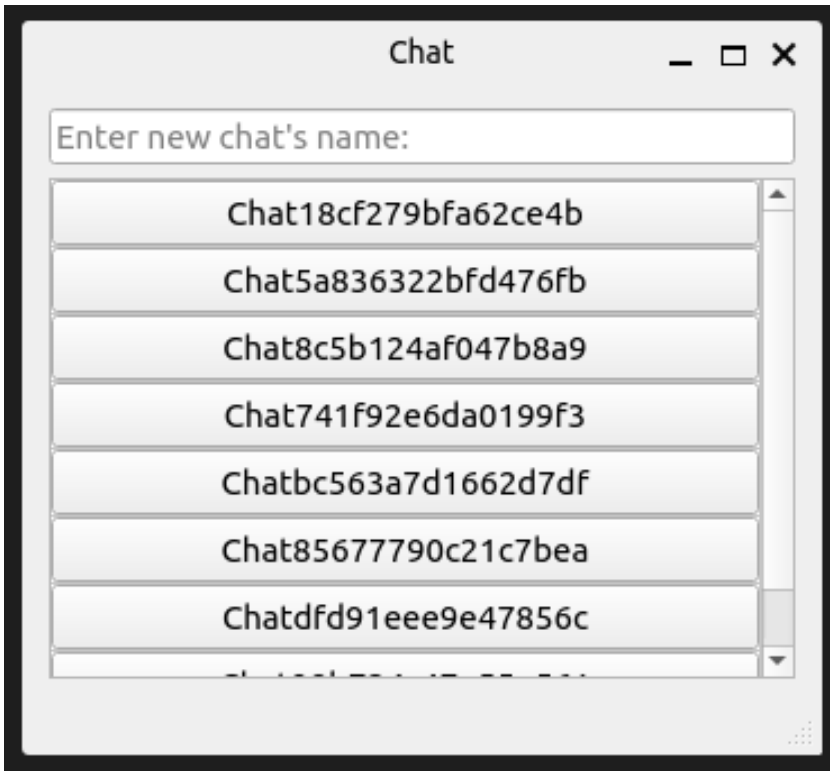


Клиентское приложение: установка имени



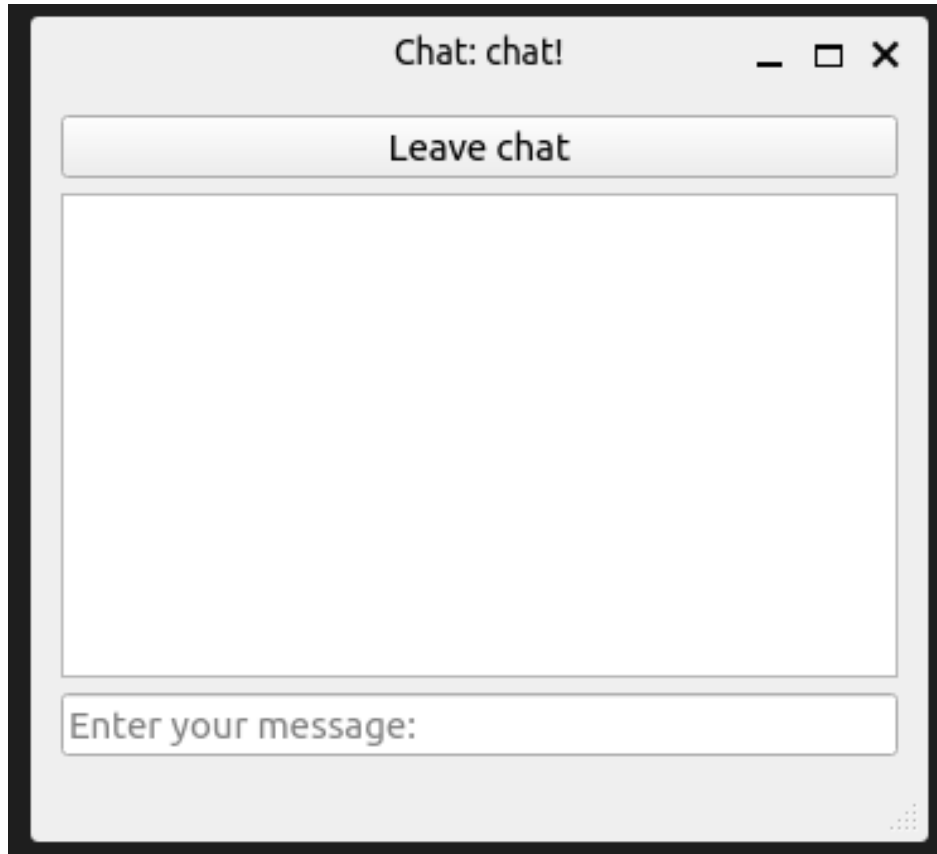
Попытка занять уже используемое имя

Клиентское приложение: список чатов

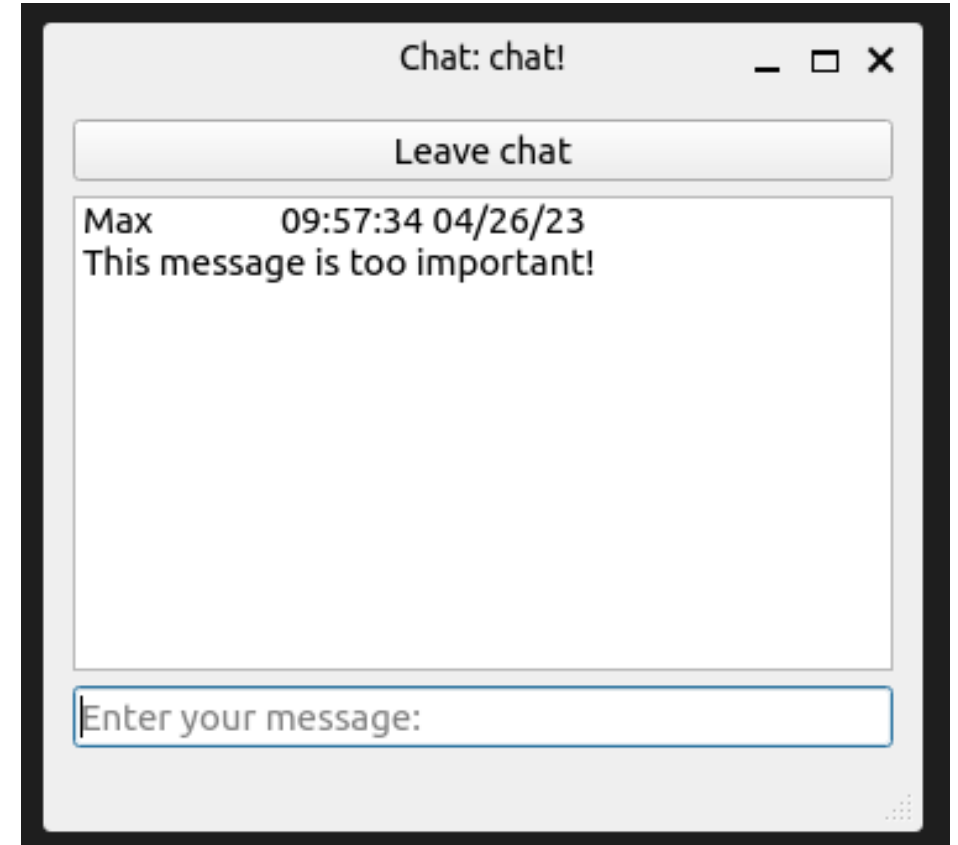


Попытка создать чат с неправильным названием

Клиентское приложение: нахождение в чате

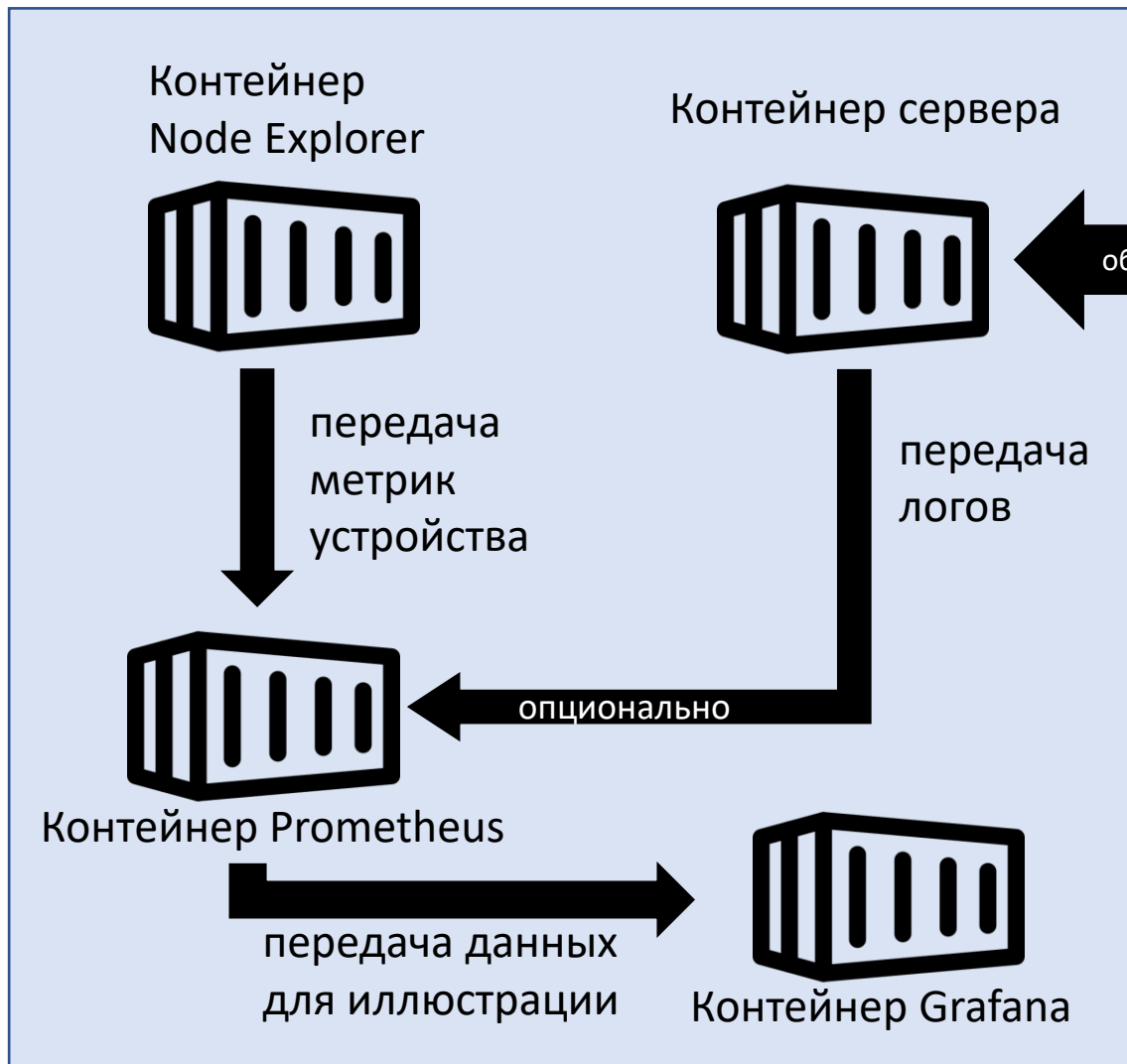


Стартовое состояние – пустой новый чат

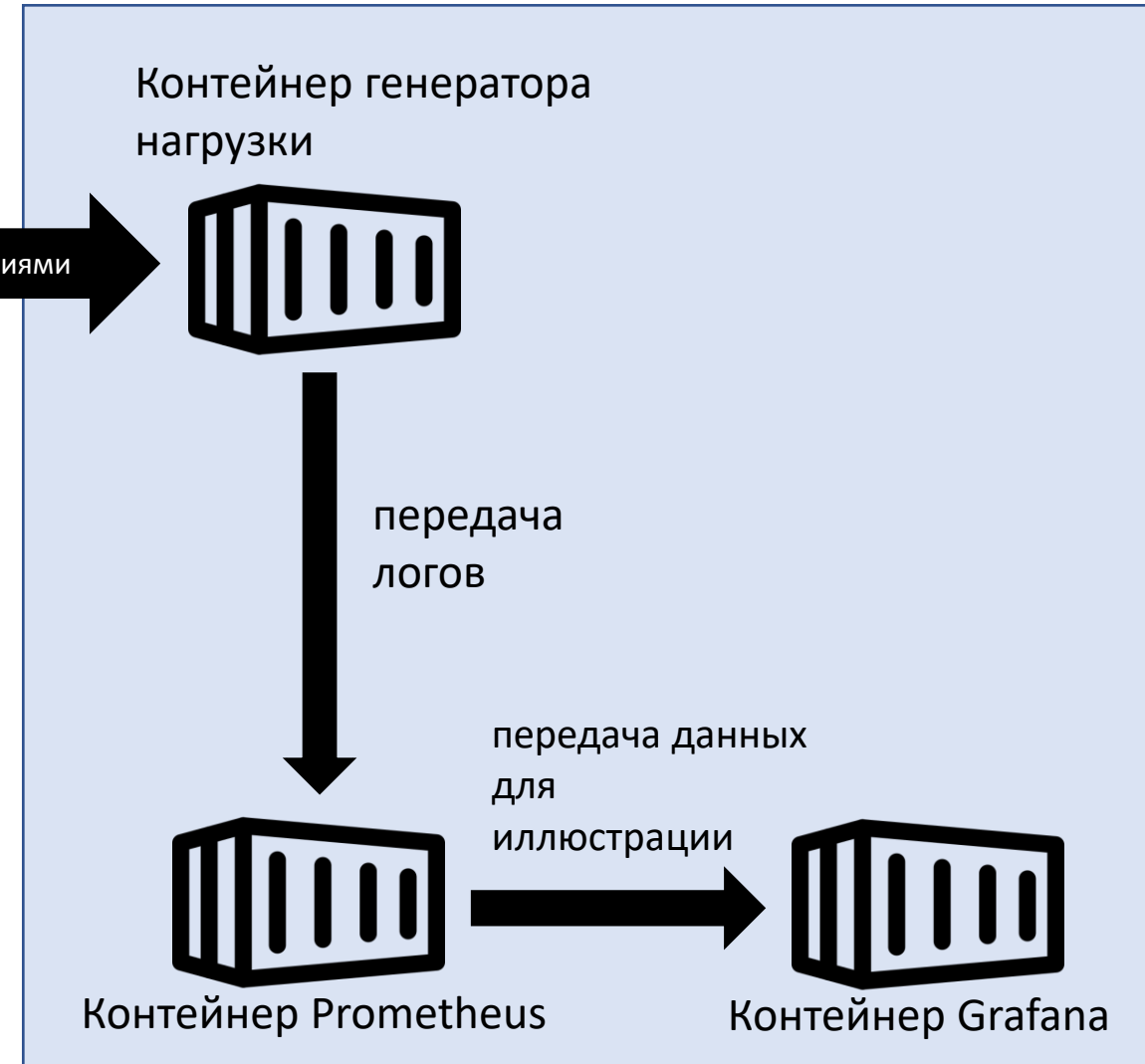


Отправка сообщения

Тестирующая система




Устройство сервера



Устройство клиента

Тестирующая система: контейнеры



Контейнер сервера	Запускается с одним открытым портом Поддерживает работу сервера и передает логи в Prometheus через скрипт на Python
Контейнер генератора нагрузки	Запускается в сетевом контексте устройства Создает нагрузку на сервер и замеряет задержки ответов
Контейнер Node Explorer	Запускается в сетевом контексте устройства Передает метрики ОП, ЦП
Контейнер Prometheus	Запускается в сетевом контексте устройства на порте 9090 Хранит обрабатываемые метрики в базе данных и поддерживает PromQL – язык запросов к ней и обработки данных метрик
Контейнер Grafana	Запускается в сетевом контексте устройства на порте 3000 Работает как иллюстративная среда со множеством инструментов

Тестирование: средства и методология

Устройства тестирования

Серверное устройство:

- 4 процессорных ядра Intel Ice Lake
- 12 Гб RAM
- 60 Гб дискового пространства

Клиентское устройство:

- 4 процессорных ядра Intel Ice Lake
- 8 Гб RAM
- 35 Гб дискового пространства

Этапы тестирования

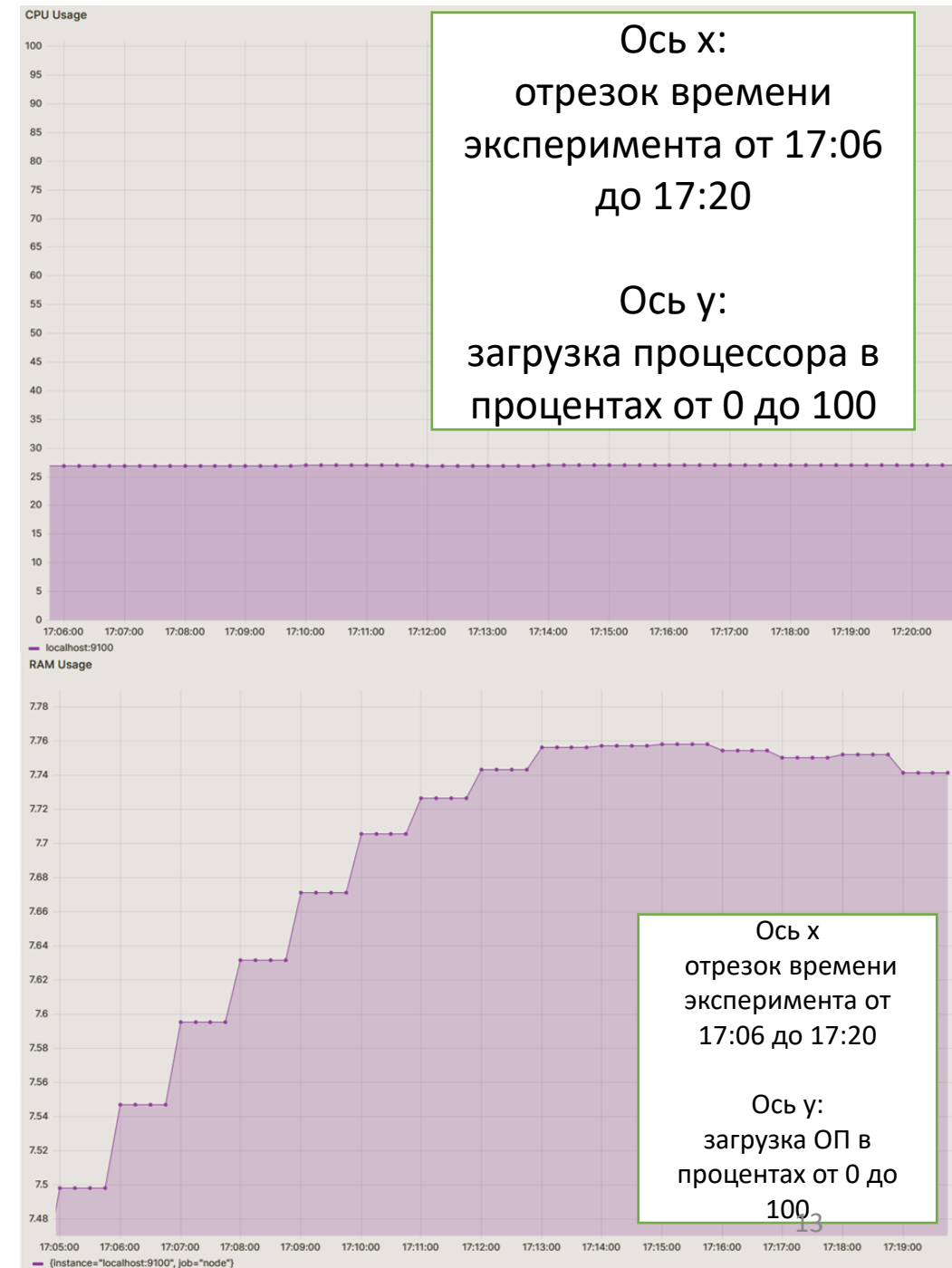
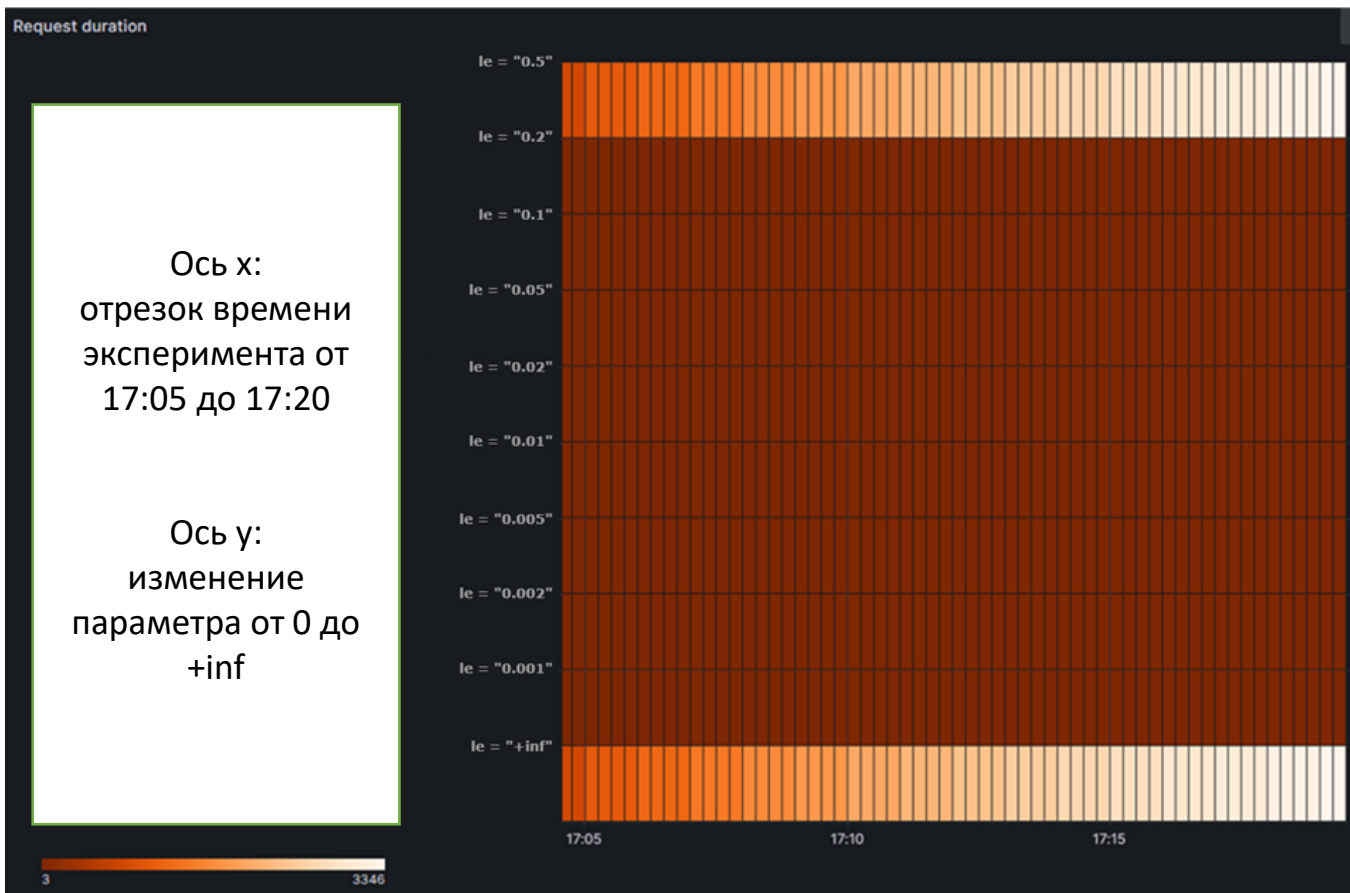
Генератор нагрузки способен выдавать многократно большее количество запросов за ограниченное время относительно реального пользователя, поэтому достаточно рассмотреть следующие случаи:

1. 2 клиента
2. 10 клиентов
3. 100 клиентов

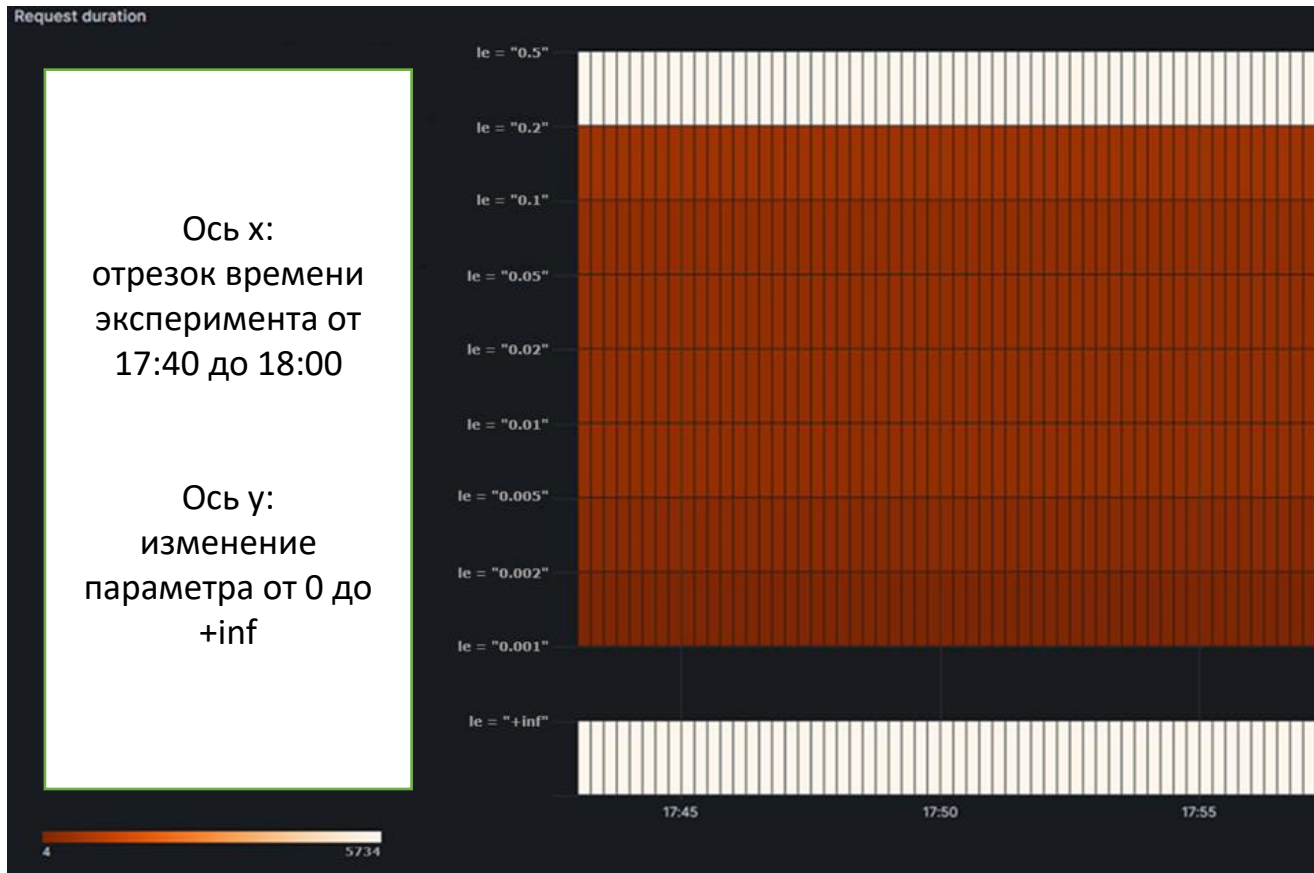
Анализируемые метрики

1. Задержка до ответа, в сек
2. Процент использования ресурсов процессора
3. Процент использования ресурсов ОП

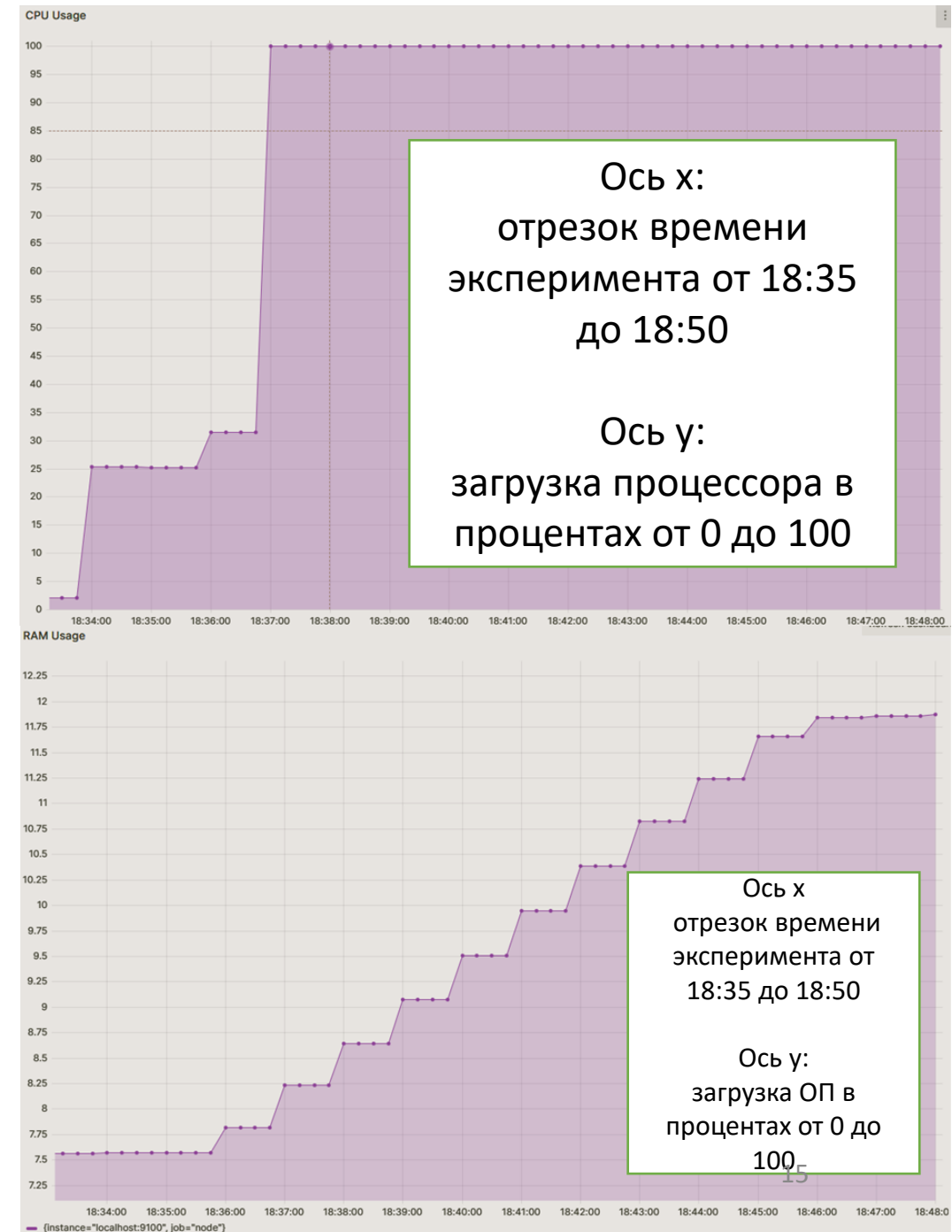
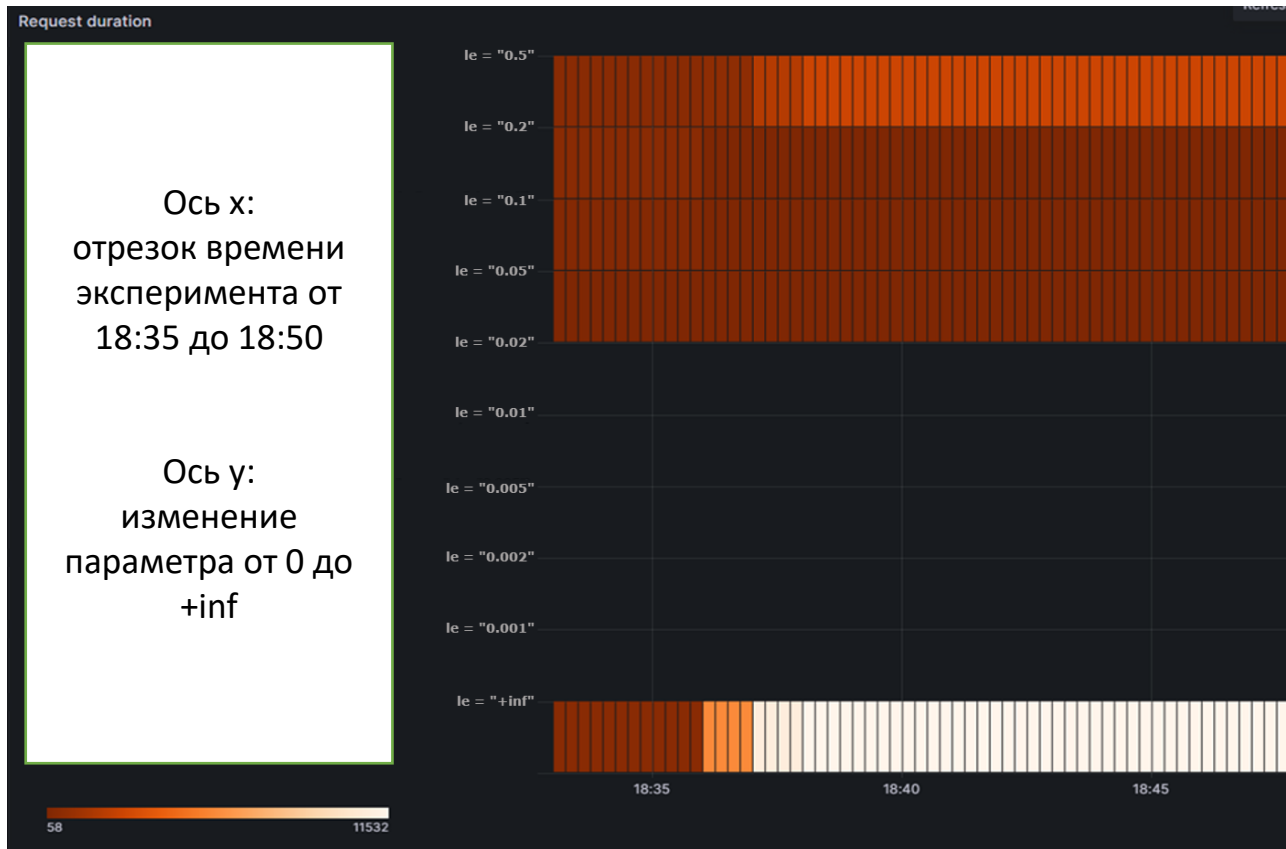
Тестирование: многопоточный сервер 2 пользователя



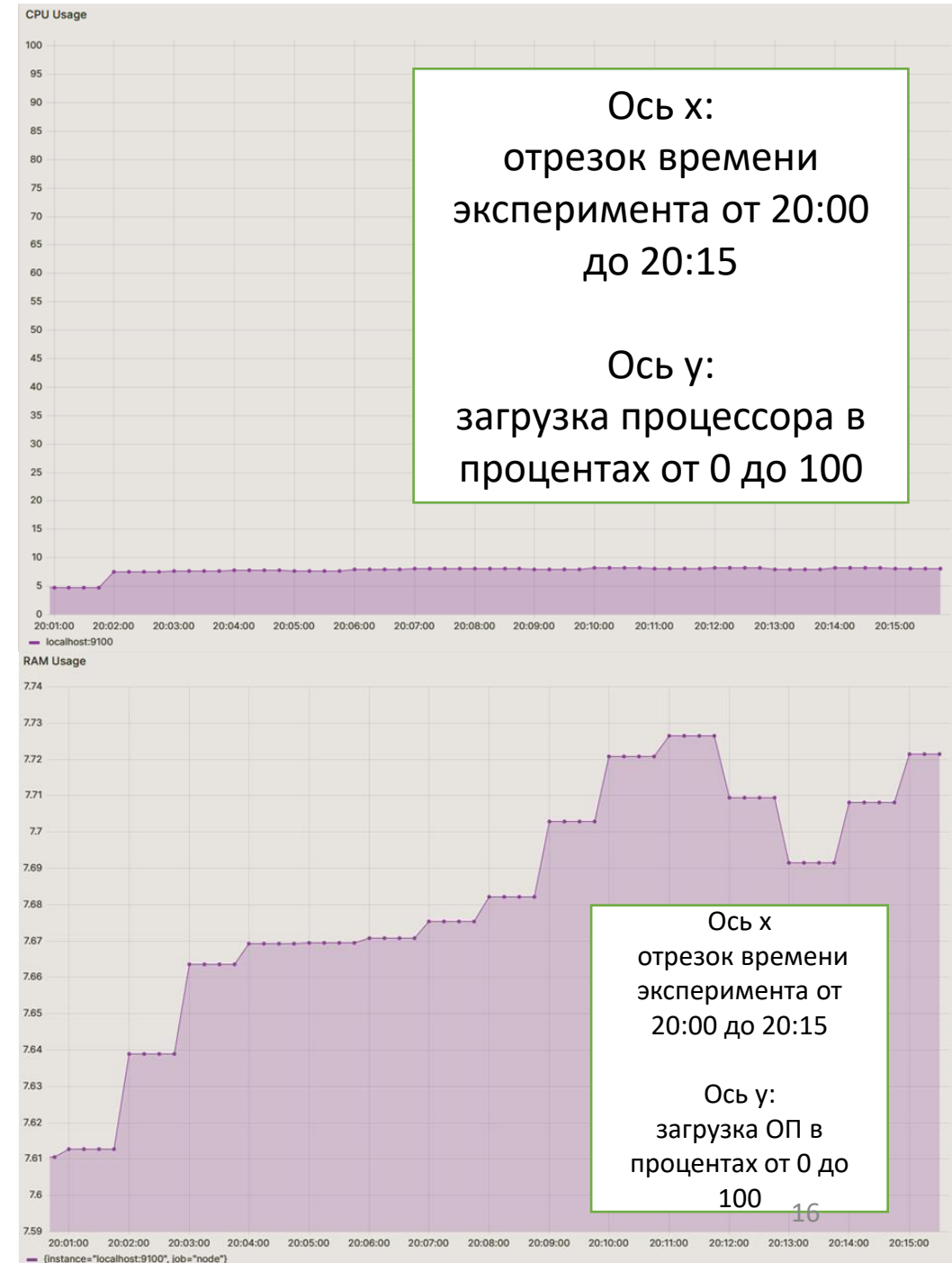
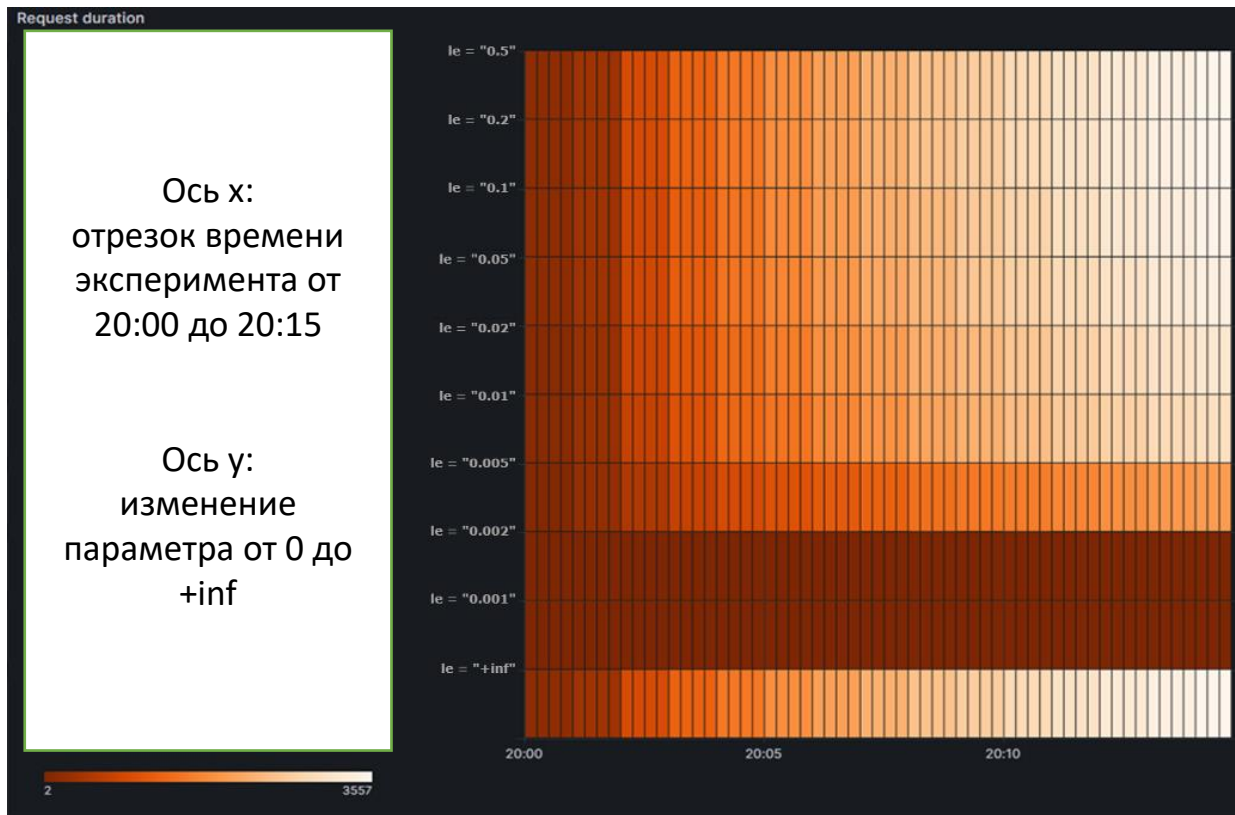
Тестирование: многопоточный сервер 10 пользователей



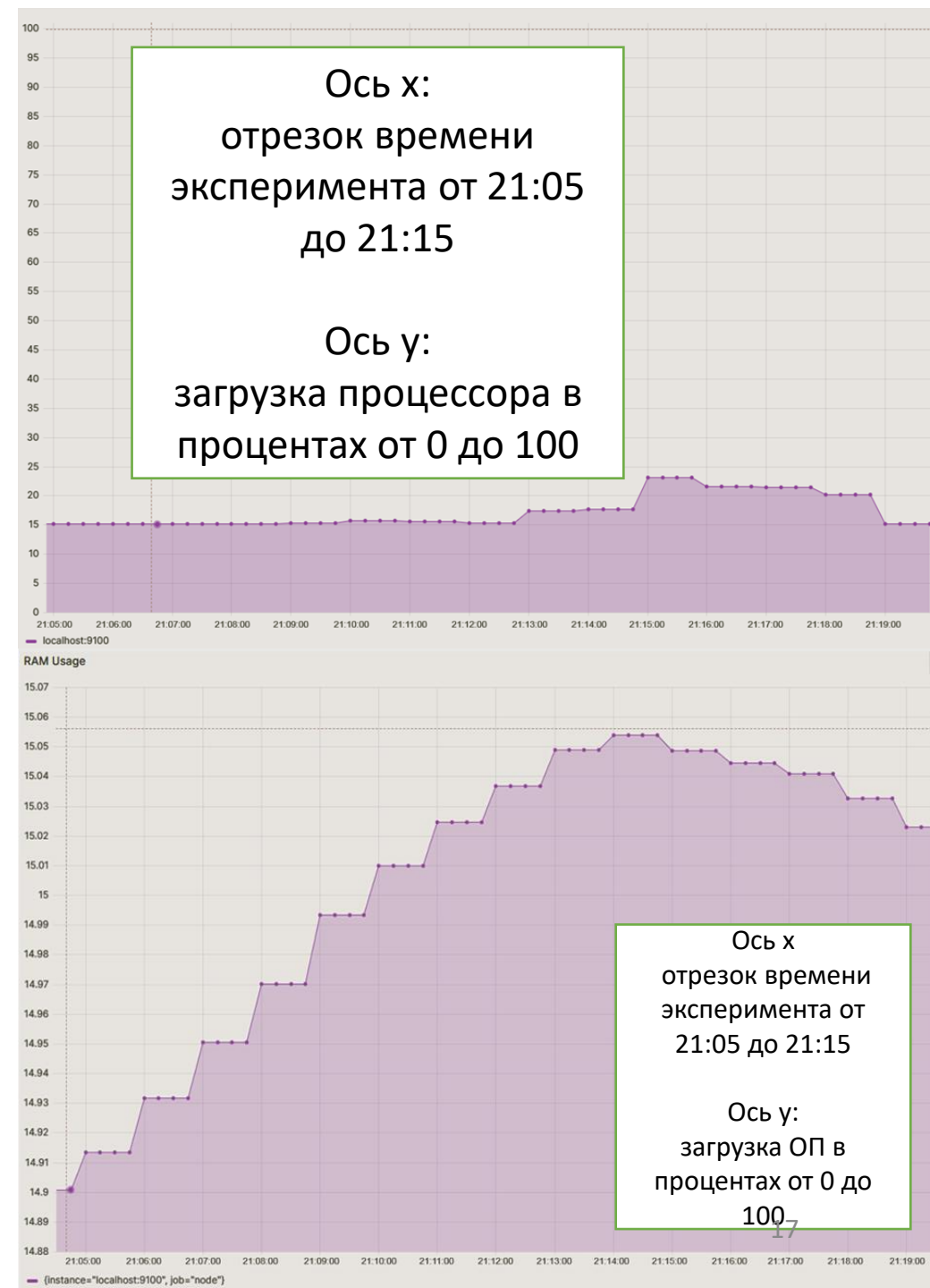
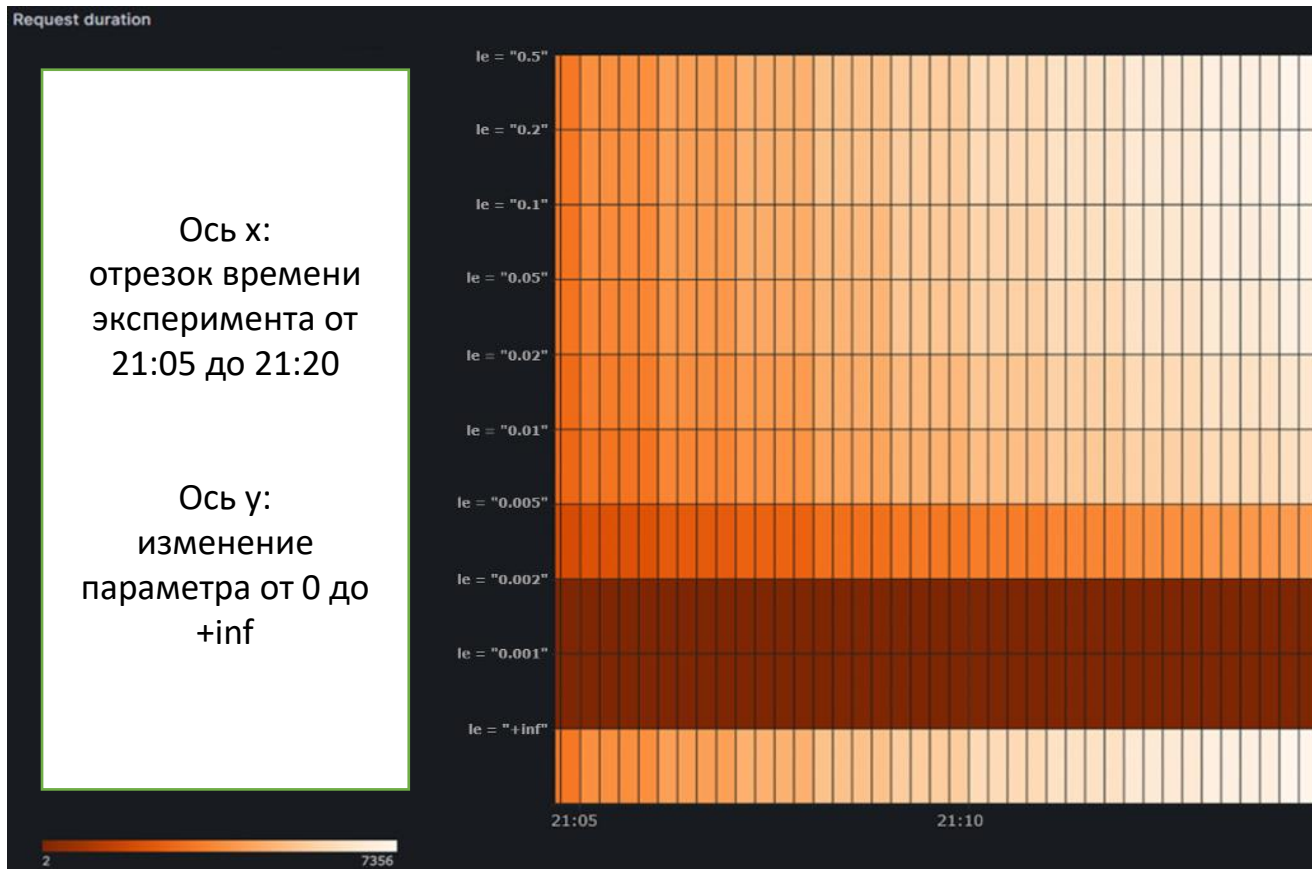
Тестирование: многопоточный сервер 100 пользователей



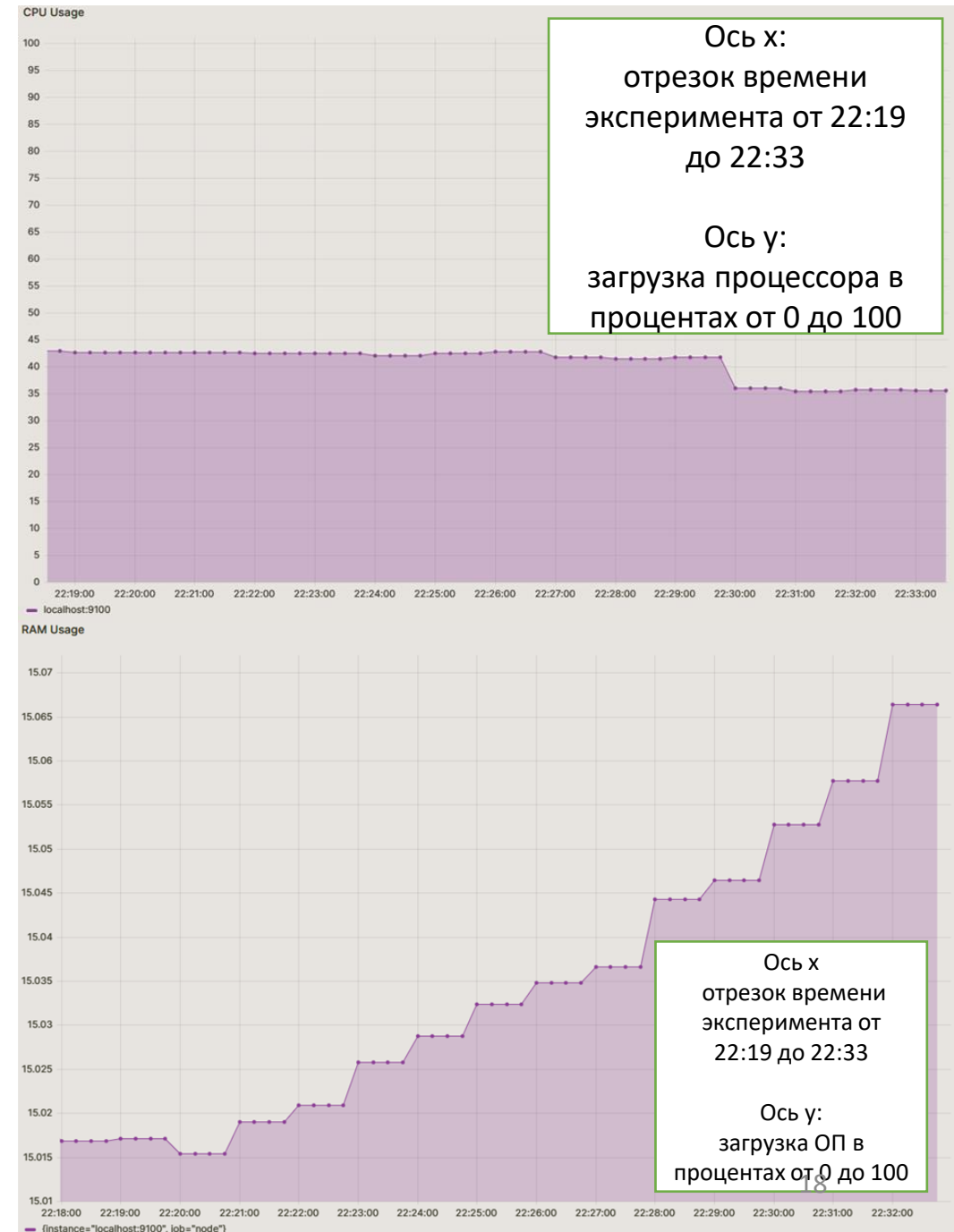
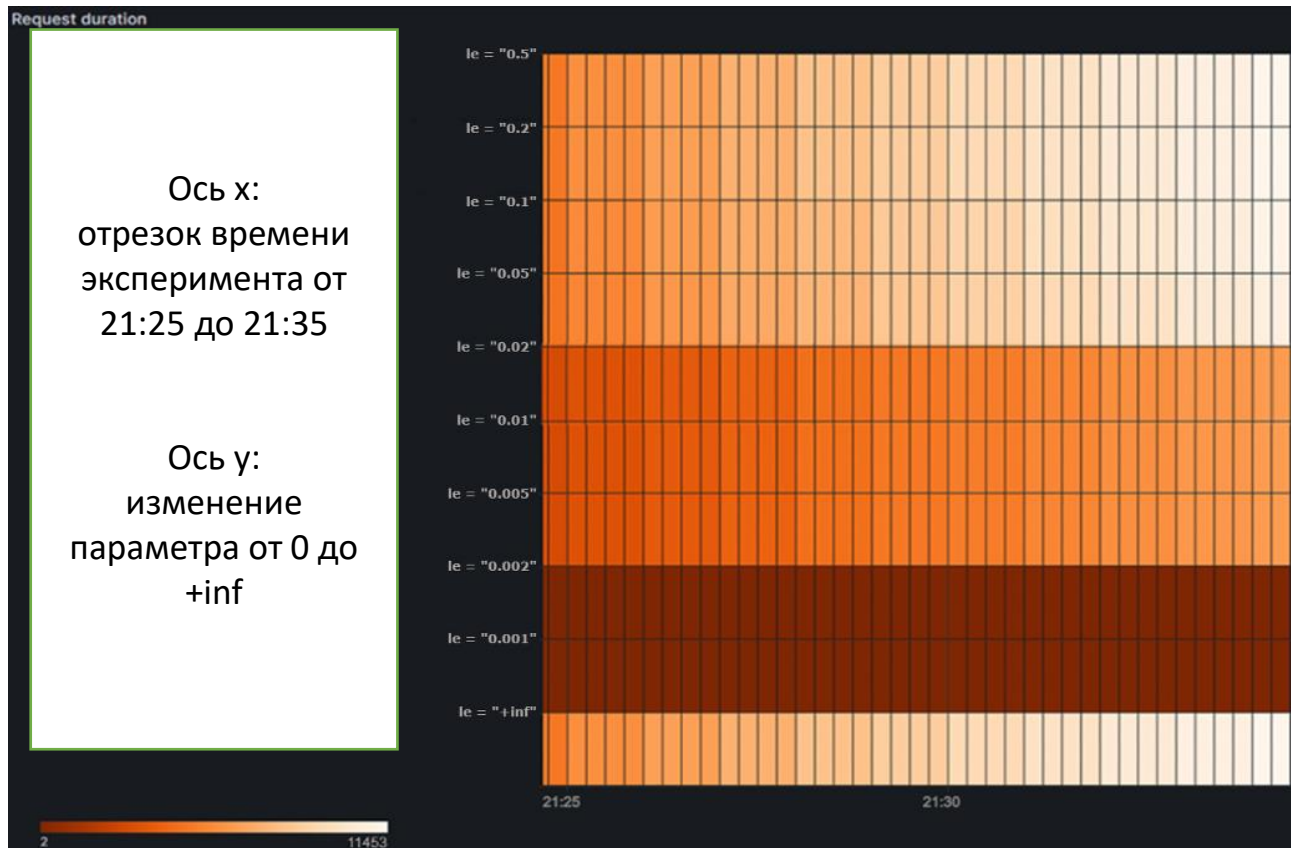
Тестирование: многопоточно-асинхронный сервер 2 пользователя



Тестирование: многопоточно-асинхронный сервер 10 пользователей



Тестирование: многопоточно-асинхронный сервер 100 пользователей



Подведение результатов

Многопоточный
сервер

Многопоточно-
асинхронный
сервер



Предпочтительный вариант:

Многопоточно-асинхронный
сервер

1. Оптимизированное выполнение подзадач с использованием асинхронности
2. Использование фиксированного пула потоков
3. Опора на ООП дает возможность применения умных указателей для упрощенного управления ресурсами памяти

Заключение

- А. Была создана платформа для сравнения производительности и анализа преимуществ и недостатков многопоточной и многопоточно-асинхронной серверных архитектур относительно друг друга.
- В. После практического освоения технологий BOOST, были созданы реализации серверов, клиентского приложения и имитатора клиентской нагрузки.
- С. Проведено тестирование, в результате которого на основе полученных графически иллюстрированных данных проведен сравнительный анализ.

Получены полезные для разработчика сведения о различиях производительности двух архитектур.

Тестирующая система показала себя работоспособной и готовой к дальнейшему применению и удобной модификации под свои нужды разработчиками.

Спасибо за внимание!