Лабораторная работа 3.3 «Моделирование и оптимизация эталонной модели для IOT системы»

(Методические указания по проекту STUDY_FOG)

ВВЕДЕНИЕ

Оптимизация – это процесс целенаправленного изменения параметров системы для достижения наилучших показателей её работы согласно выбранным критериям (например, минимальная задержка, максимальная пропускная способность, эффективное использование ресурсов). В контексте распределённых IoT-систем оптимизация направлена на поиск такого соотношения компонентов и их настроек, при котором обеспечивается стабильная и эффективная передача и обработка данных при минимальных временных затратах.

Цель лабораторной работы:

Исследовать влияние масштабирования и конфигурации компонентов эталонной архитектуры "Край-Туман-Облако" на сквозную задержку и общую производительность распределённой IoT-системы.

Цели симуляции:

- Исследовать, как количество устройств на каждом уровне (Edge, Fog, Cloud) влияет на сквозную задержку.
- Сравнить производительность стационарных и мобильных краевых устройств.
- Проанализировать влияние размера очереди на Fog-узлах на общую производительность системы.
- Научиться оптимизировать параметры системы для снижения задержек.

Необходимые навыки:

Базовое знакомство с Python, понимание основ сетевых взаимодействий и архитектур распределённых систем.

1. АРХИТЕКТУРА ЭТАЛОННОЙ СИСТЕМЫ

В первой лабораторной работе мы рассматривали систему, состоящую из конкретных устройств: робота-датчика, Fog-обработчика и т.д. На практике такие системы строятся по универсальным шаблонам — **эталонным архитектурам**. Наиболее распространенной для IoT является трехуровневая архитектура **"Край-Туман-Облако"**.

Краевой уровень (Edge): Устройства, генерирующие данные (аналоги роботов-датчиков). Могут быть стационарными или мобильными.

Туманный уровень (Fog): Промежуточные узлы для локальной обработки данных (аналоги Fog-обработчика). Имеют очереди для буферизации задач.

Облачный уровень (Cloud): Мощные централизованные серверы для глубокого анализа и долгосрочного хранения (аналоги ноутбука-сервера).

Модель в файле viz_cloud_fog_edge_pipeline.py позволяет нам экспериментировать с количеством устройств на каждом уровне и их параметрами, абстрагируясь от конкретной физической реализации.

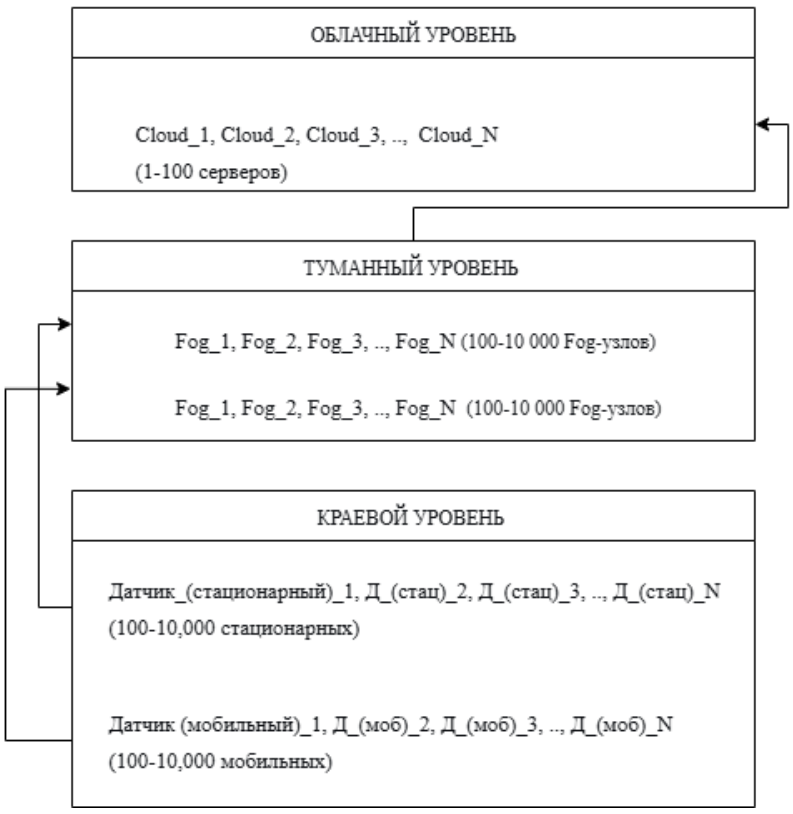

Рисунок 1 – Схема эталонной архитектуры системы

2. ПОДГОТОВКА К РАБОТЕ

2.1 Скачайте проект с GitHub репозитория robots-fog-mini-sim https://github.com/SA9Z/STUDY_FOG

2.2 Запустите эталонную модель:

>>>python viz_cloud_fog_edge_pipeline.py

2.3 Структура эталонной модели

Изучите основные компоненты модели:

- *DistributedSystemSimulator* — класс для создания распределённой системы
- *simulate_ethernet_architecture_custom()* — функция симуляции архитектуры
- *analyze_performance()* — анализ метрик производительности
- *plot_comprehensive_results()* — визуализация результатов

2.4 Настройка параметров системы

Для экспериментов с эталонной моделью найдите функцию *simulate_custom_config()* и измените параметры:

```
def simulate_custom_config():
    """Функция для быстрой настройки конфигурации системы"""

    # НАСТРОЙКА ПАРАМЕТРОВ СИСТЕМЫ
    CONFIG = {
        'edge_devices': 300,       # ↦ Количество краевых устройств (100-10000)
        'fog_nodes': 25,           # ↦ Количество Fog-узлов (100-10000)
        'cloud_servers': 8,        # ↦ Количество облачных серверов (1-100)
        'tasks': 200,              # ↦ Количество задач для симуляции
        'seed': 42                 # ↦ Seed для воспроизводимости результатов
```

}

После настройки параметров в функции simulate_custom_config() сохраните файл и запустите симуляцию командой:

>>> python viz_cloud_fog_edge_pipeline.py

Программа выполнит расчёт и отобразит на экране шесть графиков (как показано на Рисунке 2), а также выведет в консоль детальную статистику по метрикам производительности. Все графики автоматически сохраняются в папке проекта.

Каждый график отражает результаты расчетов следующих параметров системы:

График 1: "Сквозная задержка по задачам"

- Отображает индивидуальное время прохождения каждой задачи через всю систему
- Показывает вариативность задержек и выбросы

График 2: "Распределение задержек по уровням"

- Демонстрирует вклад каждого уровня (Edge, Fog, Cloud, Сеть) в общую задержку
- Помогает идентифицировать "узкие места" системы

График 3: "Сравнение типов устройств"

- Сравнивает производительность стационарных и мобильных устройств
- Показывает box-plot распределения задержек для каждого типа

График 4: "Накопительная задержка"

- Отображает кумулятивную сумму задержек по мере обработки задач
- Позволяет оценить общую временную нагрузку на систему

График 5: "Гистограмма распределения задержек"

- Показывает частотное распределение задержек
- Визуализирует 95-й перцентиль и среднее значение

График 6: "Информация о конфигурации и метрики"

- Содержит сводку параметров системы и ключевых метрик производительности

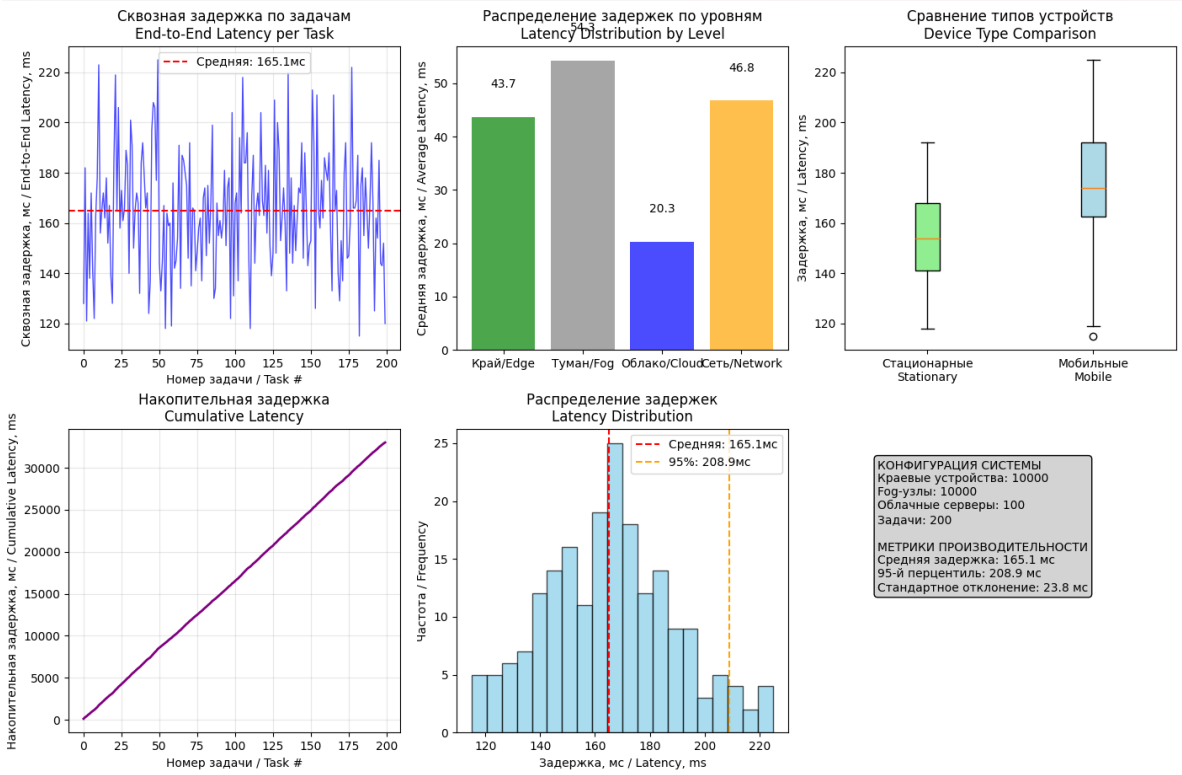

Рисунок 2 – Пример отображения метрик для оценки работы эталонной системы

Перед началом проведения экспериментов отразите расчёт основных параметров системы и рассчитанные графики в отчёте по вашей лабораторной работе, затем переходите к выполнению экспериментов. Основными параметрами являются:

- Средняя сквозная задержка – среднее арифметическое время прохождения данных от краевого устройства до облачного сервера для всех задач.
- 95-й перцентиль задержки – значение, ниже которого находятся 95% всех задержек в системе; показывает "худший случай" для большинства задач.
- Максимальная задержка – наибольшее время прохождения данных среди всех задач; указывает на потенциальные проблемы в системе.

• Средняя загрузка Fog-узлов – среднее время ожидания задач в очередях Fog-узлов; показатель загруженности промежуточных узлов обработки.

3. ПРАКТИЧЕСКИЕ ЭКСПЕРИМЕНТЫ С ЭТАЛОННОЙ МОДЕЛЬЮ

3.1 Эксперимент 1: Масштабирование системы

Цель: Исследовать, как изменение количества устройств на каждом уровне архитектуры влияет на сквозную задержку.

Шаги выполнения:

1. Базовый анализ масштабирования:

В файле viz_cloud_fog_edge_pipeline.py найдите функцию simulate_custom_config().

Последовательно установите и протестируйте три базовые конфигурации:

• Малая: 100 edge, 10 fog, 3 cloud

• Средняя: 500 edge, 50 fog, 10 cloud

• Крупная: 2000 edge, 200 fog, 20 cloud

Зафиксируйте в отчёте тенденцию изменения средней сквозной задержки при росте системы. Обычно наблюдается увеличение задержки с ростом количества устройств из-за возрастающей нагрузки на сеть и очереди обработки.

3.1.1 Индивидуальное задание:

Установите конфигурацию, соответствующую вашему варианту из таблицы.

Запустите симуляцию для своего варианта.

3.1.2 Анализ результатов:

Для своего варианта рассчитайте и занесите в отчёт метрики:

• Средняя сквозная задержка

• 95-й перцентиль задержки

- Максимальная задержка
- Средняя загрузка Fog-узлов (среднее fog_queue_delay)

Проведите анализ чувствительности системы для вашего варианта, согласно Таблице 1:

- При фиксированном количестве Fog-узлов и облачных серверов последовательно увеличивайте количество краевых устройств на 25%, 50%, 75%, 100%
- При фиксированном количестве устройств краевого и облачного слоя последовательно увеличьте количество краевых устройств на 10%, 20%, 30%...50%.
- При фиксированном количестве устройств краевого и туманного слоя последовательно увеличьте количество облачных устройств на 100%, 200%, 300%.

Для каждого изменения конфигурации зафиксируйте основные метрики и проанализируйте, как изменение соотношения компонентов влияет на производительность системы.

Таблица 1 - Варианты для выполнения Эксперимента 1

Вариант	**Краевые устройства (Edge)**	**Fog-узлы**	**Облачные серверы (Cloud)**	**Характеристика системы**
1	100	5	2	Много Edge на один Fog-узел
2	100	20	3	Много Fog-узлов на малое количество Edge
3	500	25	5	Сбалансированное соотношение
4	500	10	3	Потенциальное "узкое место" на уровне Fog
5	1000	50	8	Сбалансированная крупная система
6	1000	30	5	Умеренная нагрузка на Fog-узлы
7	2000	40	10	Высокая нагрузка на Fog-узлы
8	2000	100	15	Хорошо масштабируемая система

Вариант	Краевые устройства (Edge)	Fog-узлы	Облачные серверы (Cloud)	Характеристика системы
9	5000	200	20	Крупная система с избыточным Fog
10	5000	80	25	Крупная система с недостаточным Fog

3.2 Эксперимент 2: Анализ типов устройств

Цель: сравнить производительность стационарных и мобильных устройств.

Шаги выполнения:

1. Запустите базовую симуляцию с параметрами по своему варианту

2. Проанализируйте график "Сравнение типов устройств" и разницу в задержках. Обратите внимание на разницу в задержках между стационарными и мобильными устройствами

3. Найдите в коде настройки для разных типов устройств:

```
# В функции _init_edge_devices():
if device_type == "мобильный":
    processing_range = (25, 70)  # Немного выше задержка
    network_range = (8, 20)      # Менее стабильное соединение
else:
    processing_range = (20, 60)  # Стабильная задержка
    network_range = (5, 15)      # Стабильное соединение
```

4. Измените параметры мобильных устройств на более оптимистичные для своего варианта и сравните результаты

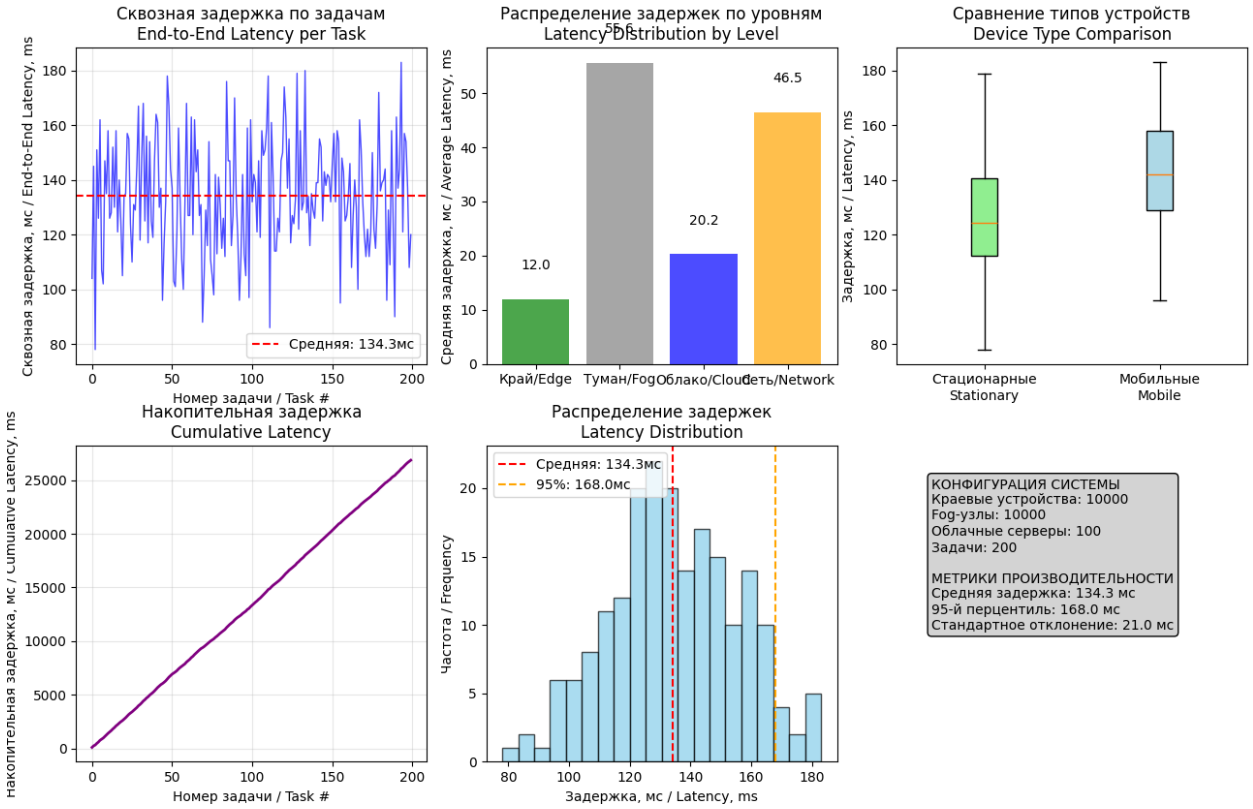

Рисунок 3 – Пример с уменьшенной задержкой

Ожидаемый результат: мобильные устройства показывают более высокую задержку из-за менее стабильного соединения.

3.3 Эксперимент 3: Оптимизация очередей Fog-узлов

Емкость очереди (queue_capacity) – это максимальное количество задач, которые Fog-узел может одновременно хранить в буфере ожидания перед их обработкой.

Что происходит в системе:

- Когда задачи приходят быстрее, чем Fog-узел их обрабатывает, они накапливаются в очереди
- Каждая задача в очереди создает дополнительную задержку (fog_queue_delay)
- Если очередь заполнена полностью, новые задачи не могут войти и получают штрафную задержку

Цель: Исследовать влияние размера очереди на Fog-узлах на общую производительность системы.

Шаги выполнения:

1. Найдите в коде инициализацию Fog-узлов:

python

```
# В функции _init_fog_nodes():
'queue_capacity': 50,  # ↦ Измените этот параметр
```

2. Протестируйте разные значения: 20, 50, 100, 200

3. Запустите симуляцию для каждого значения *queue_capacity*

4. Проанализируйте:

- Задержки в очередях Fog-узлов (*avg_fog* в метриках)
- Факты переполнения очередей (смотрите, когда *fog_queue_delay* имеет большие значения)
- Общую сквозную задержку (*avg_end_to_end*)
- Найдите самое эффективное значение ёмкости

Ожидаемый результат: слишком маленькие очереди приводят к потерям задач, слишком большие — к росту задержек.

4. СОСТАВЛЕНИЕ ОТЧЁТА

Заполните отчёт по выполнению лабораторной работы. Сделайте выводы.