Санкт-Петербургский государственный университет

Кафедра системного программирования

Оптимизация и распределённое выполнение сетевой симуляции в Miminet с использованием Docker

Альшаеб Басель, группа 24.М71-мм

Научный руководитель: к.ф.-м.н. И. В. Зеленчук

Санкт-Петербург
2025

Введение

- В данном проекте рассматривается решение проблемы, связанной с обучением компьютерным сетям на факультете Математико-механическом.
- Из-за высоких затрат на физическое оборудование, таких как коммутаторы Cisco, и сложности в его приобретении, возникла необходимость в виртуализации.
- В качестве решения было выбрано использование эмулятора **Miminet**, который позволяет создавать и тестировать сети в виртуальной среде.
- Однако проблема с длительным временем ожидания студентов при использовании эмулятора остаётся актуальной, что требует оптимизации.
- Целью данного проекта является улучшение производительности и масштабируемости **Miminet** через распределение нагрузки между несколькими контейнерами.

Существующие решения (инструменты, подходы, алгоритмы)

- Перечислить инструменты/подходы, применяемые в области
- Указать их преимущества и недостатки (критика существующих решений/подходов)

Постановка задачи

Целью оптимизация сетевых симуляций в Miminet с использованием Docker

Задачи:

- Разработка архитектуры распределённого выполнения симуляций.
- Проектирование системы управления задачами и балансировки нагрузки.
- Экспериментальное тестирование предложенного подхода.
- Валидация результатов и демонстрация практической применимости.

Обзор существующей архитектуры Miminet

- **Miminet** — эмулятор сетей на базе Mininet.
- Все симуляции выполняются в одном контейнере Docker.
- Используются виртуальные узлы и Ethernet-интерфейсы.
- **Проблемы**:
  - Невозможность масштабирования при увеличении количества узлов.
  - Использование только одного процессора, что снижает производительность.
  - Ограниченная гибкость в сетевых конфигурациях из-за общего сетевого стека.

Ограничения текущей реализации

- **Масштабируемость**: Использование одного контейнера для всех симуляций ограничивает производительность при увеличении количества узлов.
- **Использование ресурсов**: Недостаточная эффективность при многозадачности и ограниченное использование многопроцессорных ресурсов.
- **Сетевые ограничения**: Ограниченная изоляция сетевых стеков между контейнерами, что снижает гибкость и может приводить к конфликтам.
- **Пример**: Время выполнения симуляций увеличивается при увеличении количества узлов, что замедляет обучение.

Решение проблемы

- **Распределение нагрузки**: Использование нескольких Docker-контейнеров для распределения вычислительных ресурсов.
- **Многопроцессорная обработка**: Эффективное использование доступных ядер процессора для увеличения производительности.
- **Динамическое масштабирование**: Автоматическое добавление контейнеров для оптимизации времени ожидания и увеличения пропускной способности.
- В результате: Уменьшение времени ожидания студентов, повышение производительности системы.

Используемые технологии

- **Flask**: Фреймворк для создания веб-приложений на Python, используется для обработки HTTP-запросов.
- **Docker**: Контейнеризация для изоляции процессов и ресурсов.
- **Celery + RabbitMQ**: Система для распределённой обработки задач и балансировки нагрузки.
- **Mininet**: Базовая платформа для создания и тестирования сетевых симуляций.
- **SQLAlchemy**: ORM для работы с базой данных.