|  |  |
| --- | --- |
| ОТЧЕТ  о прохождении производственной практики  осенний семестр, 2024/2025 учебный год | |
| ФИО обучающегося | Федоров Илья Вячеславович |
| Физтех-школа, группа | Физтех-школа Радиотехники и Компьютерных Технологий, Б01-107 |

Место прохождения практики

ИРЭ РАН

Задание на практику

* Рассмотреть уже готовые решения
* Сравнительный анализ существующих решений
* Подготовить объект исследования
* Определить критерии и инструменты оценки
* Описать основные идеи алгоритма

Проделанная работа и полученные результаты по научно-исследовательской практике

Анализ существующих подходов и алгоритмов.

Сжатие статических файлов и оптимизация загрузки давно являются ключевыми задачами для Web-приложений. Основные инструменты, которые часто применяются, включают:

1. Для сжатия файлов:

- Gzip — широко используемая технология, обеспечивает более быструю компрессию, но с меньшей степенью сжатия.

- Brotli — современный алгоритм, созданный Google, который обеспечивает лучшее сжатие, чем Gzip, но требует больше ресурсов CPU.

- Сборщики (Webpack, Vite) — инструменты, которые помогают минимизировать JavaScript/CSS и оптимизировать загрузку фронтэнда.

2. Для настройки сервера на раздачу статических файлов:

- Nginx — используется для кэширования, балансировки нагрузки и раздачи статического контента. Стандартные методы балансировки в Nginx часто основаны на алгоритмах round-robin или least-connections.

3. Для масштабирования рабочих узлов:

- Docker — контейнеризация приложений для обеспечения независимости и переносимости.

- Kubernetes — управление контейнерами для автоматического скейлинга и мониторинга.

Сравнительный анализ существующих решений

|  |  |  |
| --- | --- | --- |
| Инструмент | Преимущества | Ограничения |
| Gzip | Быстрое сжатие, поддержка широким рядом браузеров | Средний уровень сжатия (хуже Brotli) | |
| Brotli | Лучшее сжатие, особенно для текстового контента | Высокая загрузка CPU, не всегда оправдана для микросервисов| |
| Nginx | Высокая производительность, поддержка балансировки | Не анализирует CPU/RAM/географию пользователей |
| Kubernetes | Отличная поддержка автоскейлинга, отказоустойчивости | Может быть сложным для настройки при небольших масштабах |

Таким образом, несмотря на эффективность существующих решений, у них есть ограничения в гибкости, кастомизации и аналитике. Это делает создание собственного алгоритма балансировки нагрузки обоснованным шагом.

**Выбор объекта исследования**

Объектом исследования был выбран готовый проект видеохостинга, как пример типичного Web-приложения, которое активно использует серверные ресурсы и испытывает большие нагрузки. Видеохостинг идеально подходит для тестирования:

- Реальной работы сервера под высокой нагрузкой (обработка больших объёмов видеофайлов).

- Отображения метрик пользователя, таких как LCP и FCP.

Краткие характеристики объекта:

- Серверная часть: Node.js с несколькими инстансами.

- Распределение статических ресурсов: Nginx.

- Количество пользователей: до 5000 запросов в секунду в пиковое время.

Таким образом, видеохостинг позволяет оценить как клиентскую, так и серверную оптимизацию, а также протестировать нагрузочный алгоритм в реальных условиях.

Определение критериев и инструментов оценки

Для анализа успешности системы мы фокусируемся на двух аспектах: пользовательском опыте и производительности сервера.

Критерии оценки для клиента:

1. LCP (Largest Contentful Paint) — отражает скорость рендеринга крупнейшего визуального элемента.

2. FCP (First Contentful Paint) — отображает скорость загрузки первого полезного контента.

Критерии оценки для сервера:

1. Латентность (Latency) — среднее время ответа сервера на запрос.

2. RPS (Requests Per Second) — способность сервера обрабатывать большое количество запросов.

3. CPU Usage — загрузка процессора во время работы.

4. Memory Usage — потребление оперативной памяти сервером.

5. P95 Latency — время ответа для 95% случаев, помогает оценить пиковую нагрузку.

6. Error Rate — количество ошибок на уровне сервера (например, 5xx).

Инструменты:

- Фронтенд — встроенная консоль разработчика в браузере для анализа LCP/FCP.

- Бекенд — Apache JMeter, Gatling или k6 для имитации высокого RPS и анализа латентности.

Описать основные идеи алгоритма

Задача:

Создание собственной системы балансировки нагрузки между несколькими серверными узлами. Основные особенности:

1. Анализ нагрузки в реальном времени по трём параметрам:

- Загруженность CPU и RAM.

- Географическое положение запросов (для минимизации задержек по сети).

- Количество активных запросов на сервер.

2. Расчёт "веса" узлов для выбора оптимального сервера.

Пример взвешивания:

weight = (max\_cpu - current\_cpu) \* 0.6 + (max\_memory - current\_memory) \* 0.4 - latency\_factor

где:

- latency\_factor определяется расстоянием пользователя до сервера.

- Узел с максимальным весом обрабатывает запрос.

3. Реализация REST API для тестирования:

- С помощью API можно будет отправить нагрузочные запросы и собирать метрики (латентность, время отклика).

4. Инструменты разработки:

- Node.js и Express для создания системы.

- Grafana или Prometheus для мониторинга.

**Сравнительный анализ с Nginx и HAProxy**

1. Тестирование:

Провести несколько сценариев тестирования:

- 5000 RPS с равномерным распределением.

- Географически распределённая нагрузка (например, пользователи из Европы и США).

- Искусственная симуляция повышенного CPU/RAM на одном из узлов.

2. Ожидаемые результаты:

- В тесте с высокими нагрузками собственная система балансировки должна показывать лучшее использование ресурсов узлов за счёт динамического анализа CPU, RAM и латентности.

- В сравнении с Nginx и HAProxy можно ожидать снижение латентности для географически распределённых запросов.

|  |  |
| --- | --- |
| Отзыв руководителя  о работе обучающегося |  |
| Оценка за работу обучающегося, рекомендуемая руководителем |  |

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ дата составления отчета \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Контактный телефон: 8-9\_\_-\_\_\_-\_\_-\_\_

Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

Контактный телефон: 8-9\_\_-\_\_\_-\_\_-\_\_ e-mail:

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/