**РЕЧЬ**

**Слайд #1**

Здравствуйте уважаемые члены государственной аттестационной комиссии, меня зовут Махмудов Бабур. Тема моей выпускной квалификационной работы – «исследование методов повышения производительности систем веб-шаблонов и разработка системы шаблонизации на их основе».

**Слайд #2**

Актуальность данной работы заключается в том, что существующие решения не эффективны в плане потребления вычислительных ресурсов, также большинство из них предназначены для использования в составе более крупных веб-фреймворков, что делает их не совсем удобными в эксплуатации людьми не технического профиля. Из этого следует, что, разработав производительную систему веб-шаблонов с высокими показателями удобства эксплуатации позволит экономить как вычислительные, так и трудовые ресурсы.

**Слайд #3**

Основная цель проведённого исследования – это поиск способов повышения производительности веб-шаблонов. Системы веб-шаблонов и их производительность являются объектом и предметом исследования соответственно.

**Слайд #4**

На данном слайде представлены задачи, которые были решены в ходе выполнения работы.

**Слайд #5**

В ходе исследовательской работы были определены основные факторы, которые негативно сказываются на производительности системы веб-шаблонов:

1. Подавляющее большинство существующих решений написаны на интерпретируемых языках (PHP, Python, JavaScript).
2. Присутствие механизм автоматического управления памятью, также известного как сборщик мусора.
3. Не всегда присутствует кеширования обработанных шаблонов, что приводит к необходимости повторной обработки.
4. Вычислительные ресурсы не используются полностью, из-за исполнения программы в одном потоке, и наличием блокировок, связанных с чтением и записью с медленных хранилищ данных.

**Слайд #6**

Здесь представлено схематическое сравнение интерпретируемых и компилируемых языков программирования.

При компиляции, компилятор обрабатывает исходный код и производит оптимизированных машинный бинарный код под целевую архитектуру ЭВМ.

При интерпретации обработка исходного кода происходит построчно, и результатом такого процесса является промежуточный код виртуальной машины, так же известной как байт-код. Далее этот байт-код подвергается процессу трансляции в машинный код целевой архитектуры.

Такой подход к исполнению программы в несколько раз медленнее по сравнению с компилируемым аналогом.

**Слайд #7** (может стоит убрать слайд)

Простой вычислительных ресурсов подразумевает, что в многоядерных процессорах при работе системы веб-шаблонов используется лишь одно ядро, в то время как остальные бездействуют.

**Слайд #8**

Для решения ранее описанных проблем, были предложены следующие подходы:

1. Использование компилируемых языков программирования вместо интерпретируемых.
2. В выбранном языке программирования не должно быть механизма сборки мусора, а в целях упрощения управления памятью необходимо чтобы язык поддерживал парадигму RAII (объект который получил память, ответственен за её освобождение).
3. Реализация кеширования шаблонов с применением вытесняющих алгоритмов LRU (вытеснение давно не используемых объектов из кеша) или LFU (вытеснение наименее используемых объектов).
4. Реализация системы веб-шаблонов как веб-сервис, который может параллельно и асинхронно обрабатывать большое количество запросов на построение веб-документов.

**Слайд #9**

Укрупнённо модель только что упомянутого веб-сервиса выглядит следующим образом:

Множество клиентов делает обращения к сервису посредством HTTP запросов, сервис перенаправляет запросы на соответствующие модули в зависимости от запроса.

**Слайд #10**

На данном слайде приведена IDEF0 диаграмма обработки таких HTTP запросов, по ней видно, что после предварительной обработки запроса веб-сервис перенаправляет запрос на соответствующую подсистему для дальнейшей обработки.

**Слайд #11**

Для реализации ранее описанного веб-сервиса построения веб-документов были использованы следующие технологии:

Серверная часть была написана на языке программирования Rust. Для поддержки многопоточного и асинхронного исполнения была использована библиотека tokio. А для непосредственно самой шаблонизации в веб-сервис был интегрирован шаблонизатор tera.

Пользовательский веб-интерфейс был написан с применением javascript-фреймворка vuejs и использованием стандартных html и css.

В силу клиент-серверной архитектуры, клиентом может выступать любая сущность, поддерживающая HTTP протокол обмена данными.

**Слайд #12**

На этом слайде приведён пример пользовательского веб-интерфейса разработанного сервиса. А также по ссылке можно ознакомится с демонстративной версией разработанной системы.

**Слайд #13**

По завершению разработки были проведены тесты для оценки производительности.

Тестирование велось на виртуальном сервере от Google в Стокгольме в то время как клиентские запросы исходили из штата Вирджиния, на слайде приведены характеристики этого сервера.

На диаграмме показано соотношение одновременных подключений в конкретные интервалы времени и времени ответа веб-сервиса. А именно по диаграмме видно, что на 550 одновременных подключений среднее время ответа системы составляет чуть более 4-ёх секунд.

**Слайд #14**

На данной картинке можно ознакомиться с нагрузкой, которая фиксировалась на сервере во время проведения теста. По ней видно, что оба ядра процессора нагружены только на 50%, т.е., не смотря на большое количество запросов, разработанная система веб-шаблонов хорошо справляется с поставленными задачами.

**Слайд #15**

В заключении можно сказать что все поставленные цели были достигнуты: были проанализированы проблемы низкой производительности веб-шаблонов, предложены способы устранения данных проблем, после чего разработана высокопроизводительная и удобная в использовании система веб-шаблонов.

**Слайд #16**

СПАСИБО ЗА ВАШЕ ВНИМАНИЕ