Министерство образования Республики Беларусь
Учреждение образования
Белорусский государственный университет
информатики и радиоэлектроники
Кафедра инженерной психологии и эргономики

На правах рукописи

УДК 004.738.1-048.34

Дулина
Руслан Александрович

ОПТИМИЗАЦИЯ ПРОЦЕССА ЗАГРУЗКИ СТРАНИЦ МНОГОФУНКЦИОНАЛЬНЫХ САЙТОВ: КРИТЕРИИ И ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ

Автореферат на соискание академической степени
магистра технических наук

1-23 80 08 – Психология труда, инженерная психология, эргономика

Магистрант Р.А. Дулина

Научный руководитель
И.И. Петровский, канд.
технических наук, доцент

Заведующийкафедрой ИПиЭ
К.Д. Яшин, кандидат
технических наук, доцент

Нормоконтролер
Е.С. Иванова,
ассистент кафедры ИПиЭ

Минск 2019

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

В настоящее время наблюдается усиление спроса на автоматизированные системы сборки приложений. И если раньше сборка продуктов никак не была связана с качеством продукта, то теперь с развитием программных продуктов и невероятным скачком в развитии клиентской части все больше требований возникает к ресурсам, которые должны стать эффективным инструментом в обслуживании пользователя. Современные бизнес-реалии привлекли внимание к областям, которые не столько нацелены на создание технического решения конкретной задачи, сколько на разработку нового типа программного обеспечения промежуточного уровня (middleware), который объединяет функциональные возможности других уже существующих типов программного обеспечения. Такой подход упрощает ведение бизнеса компании.

Сборка затрагивает не только отдельно взятый продукт, систему, но и взаимодействие различных крупных систем для ведения общего бизнеса или решение глобальных задач. Именно последний тип взаимодействия будет рассмотрен в рамках диссертации.

Настоящая работа будет посвящена изучению оптимизации загрузки страницы на этапе сборки приложения, разработке собственного подхода к решению проблемы сборки кода для многостраничных сайтов. Разрабатываемое решение основано на принципах возможного изменения алгоритмов сборки без существенного изменения архитектуры системы. Подход к обеспечению требуемого уровня удобства использования также освещён в рамках диссертации.

В диссертации будут рассмотрены различные инструменты, подходы и методы обработки как структурированных, так и неструктурированных данных. Также будет произведен сравнительный анализ различных техник кластеризации и их практическая реализация в процессах сборки приложения.

Соблюдение эргономических принципов будет достигаться исследованием юзабилити метрик, полученных путем тестирования приложения автоматизированными системами.

Для того, чтобы исследование позволило решить ряд реальных задач, разработка основывается на требованиях к реальной системе Cobalt Platform. Данная платформа регистрирует до 202370 активных сессий ежедневно.

ВВЕДЕНИЕ

С развитием программных продуктов и невероятным скачком в развитии клиентской части, все больше вопросов появляется в обслуживании и поддержке клиентского кода. Современные стандарты позволяют использовать модульную структуру построения клиентской части. Модули позволяют облегчить поддержку кода и сократить количество ошибок в коде, однако привносят новые проблемы в управление и построение этого кода.

И для решения этой задачи нужен бандлер (bundler). Это инструмент для сборки модулей в единые пакеты. Сборщик модуля заменяет загрузчик модулей и создаёт пакет, содержащий весь код, во время сборки. Получающиеся пакеты совместимы с браузером, которому не нужен доступ к файловой системе. Бандлер нужен для поиска всех выражений модулей (имеющих ошибочный, с точки зрения браузера, JS-синтаксис) и замены на настоящее содержимое каждого требуемого файла. В финале получается единый JS-файл.

Если попробовать использовать модульный код в браузере, то получится ошибка, в которой говорится: «модуль не определён». У браузера нет доступа к файловой системе, и он не может эффективно подгружать сотни и сотни модулей по мере их надобности.

ECMAScript 5 и более ранние версии не были спроектированы с учётом модулей. Со временем разработчики нашли различные возможности симулировать модульную архитектуру на JavaScript.

Стандарт ES-2015(ES6) был принят в июне 2015. Пока что большинство браузеров реализуют его частично. В настоящее время существует один транслятор JS-кода, позволяющий переписать код на ES-2015(ES6) в код на предыдущем стандарте ES5 - Babel.JS. Обычно Babel.JS работает на сервере в составе системы сборки JS-кода (например: Webpack, brunch, SystemJS, RequireJS, Browerify) и автоматически переписывает весь код в ES5. Задача всех систем собрать код наиболее быстро и эффективно. Система Webpack имеет встроенные возможности оптимизации сборки модулей.

Однако, они ориентированы на сборки одностраничных или малостраничных приложений, с возможностью ручной настройки процесса сборки.

Таким образом, на рынке присутствует всего лишь одна интеллектуальная система сборки приложения, позволяющая получить приемлемые результаты после преобразования кода в ES5. Webpack предлагает два плагина для решения этих задач: CommonsChunkPlugin и (DllPlugin, DllReferencePlugin). Подробная реализация и сравнения будут рассмотрены в основной части работы.

После проведенного анализа становится ясно, что для эффективной сборки многостраничных сайтов (позволяющих получить хорошие productivity метрики) на рынке не существует ни одного решения.

В диссертации для решения проблем повышения эффективности будут использовать подходы Big Data. Число предприятий, использующих Big Data в своей работе, непрерывно растет. Практика последних лет показала, что использование результатов анализа больших массивов данных может принести реальный эффект, который положительно отразится на банковских счетах компании. Но, неожиданно для себя, предприниматели сталкиваются с тем, что кроме ряда преимуществ, существует огромное количество проблем при внедрении Big Data, решение которых требует привлечения довольно значительных ресурсов. И процесс построения многостраничных сайтов это как раз тот случай, когда приходится признать, что в Big Data есть проблемы. Чем больше объем накопленных данных, тем требовательнее система хранения и управления этими данными.

Задачей юзабилити метрик является определение, удобен ли некоторый искусственный объект, модель (такой, как веб-страница, пользовательский интерфейс или устройство) для его предполагаемого применения. В юзабилити нет такого инструмента, который помог бы однозначно определить, насколько ваш программный продукт хорош или плох. Однако, есть много инструментов, которые помогут это сделать. Особенно когда мы говорим о productivity и effectiveness метриках. Основной задачей диссертации будет улучшение времени, необходимого для выполнения задачи.

Как говорилось ранее, решение будет носить характер middleware и может быть применено для любого многостраничного приложения, которое базируется на стандартах ES6 и выше. Данное решение позволит заложить хорошие юзабилити метрики на этапе настройки сборки самого кода.

При проектировании крупной системы затрачивается огромное количество ресурсов. Таким образом, цена архитектурной ошибки очень высока. Не менее важно соблюдение инженерно-психологических принципов. Данные принципы необходимо соблюдать на всех этапах жизненного цикла продукта.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во введении рассмотрено современное состояние проблемы (проблема оптимизации загрузки многостраничный веб-сайтов) определены основные направления исследований, а также дается обоснование актуальности темы диссертационной работы.

В общей характеристике работы сформулированы ее цель и задачи, показана связь с научными программами и проектами, даны сведения об объекте исследования и обоснован его выбор, представлены положения, выносимые на защиту, приведены сведения о личном вкладе соискателя, апробации результатов диссертации и их опубликованность.

В первой главе рассматриваются подходы к решению задач оптимизации. Рассматриваются основные метрики, влияющие на юзабилити программного продукта. В рамках данной диссертации исследуется эффективность методов кластеризации в уменьшении времени загрузки приложений. Скорость заключается в высоком уровне юзабилити системы в целом. В связи с этим в работе сформулированы следующие задачи:

– выявить влияния кэша приложения и размера подгружаемого кода на метрики эффективности и продуктивности,

– изучить факторы влияющие на размер страницы,

– формализовать общие требование эффективности кластеризации дынных страницы,

– разработать систему сборки клиентского кода с использование методов кластеризации,

– рассмотреть средства сбора метрик эффективности.

Во второй главе приведен процесс разработки системы сборки клиентской части приложения, который должен значительно ускорять время загрузки страницы. Рассматривается дилемма между загрузкой необходимого минимума ресурсов для конкретной страницы и кэшированием ресурсов в рамках всего приложения.

1. Время загрузки страницы можно уменьшить, избавившись от неиспользуемых ресурсов и использовать оптимизированный уникальный пакет для каждой конкретной страницы. Такие пакеты содержат только ресурсы, которые действительно требуются для страницы, очень мало ресурсов будет потрачено впустую, а время загрузки страницы должно быть оптимальным. Однако, если конечный пользователь просматривает несколько страниц с пакетами, построенными на основе этой стратегии, вы обнаружите, что приложение генерирует много трафика и, наконец, UX ухудшается - каждый комплект пакетов уникален и ничего не кэшируется между страницами.

Время загрузки страницы для всего приложения ухудшается с каждой новой открытой страницы,

Другая основана на идее кэшировать всех необходимых ресурсы для всего приложения, загружая их на первую запрашиваемую страницу. Прохождение других страниц не требует загрузки каких-либо ресурсов, поскольку они были кэшированы при первой загрузке. Однако широкомасштабное приложение, которое содержит десятки страниц, содержит десятки мегабайт (даже минимизированные) зависимостей и время загрузки страницы для первой запрашиваемой страницы, является неприемлемым. Более того, большинство пользователей обычно не используют все доступные функции / страницы, а принудительная загрузка будет довольно неэффективной.

В третьей главе рассматривается процесс построения системы, которая будет направлена на оптимизацию загрузки страниц. В этой главе приводится испытание системы и ее показатели. Благодаря использованию кластеризации при построении JavaScript ES6 кода мы получили 30% прироста скорости для Home страницы и около 70% на SearchResults. Как говорилась раньше в многостраничных сайтах до 200 страницу прирост производительности может доходить и до 90%.

В приложениях приведены блок схема преобразования Webpack графа и диаграмма последовательности многопоточной сборки бандлов.

ЗАКЛЮЧЕНИЕ

В рамках диссертации была разработана автоматизированная система сборки кода. Было уменьшено не только время загрузки страниц, но и реализована эффективная система сборки кода для этих страниц. Система показывает высокую производительность за счет распределения нагрузки по раздельным потокам в рамках одного процесса, вместо запуска нескольких процессов. Благодаря этому потоки поддерживают подключение всех доступных основному процессу модулей.

Разработана и реализована логика построения пакетов в зависимости от иерархии, которая сокращает время сборки модулей до 10 раз, а при учете сборки модулей с *.map файлами, до 20 раз.

Было достигнуто уменьшение расхода места на серверах до 80%, где хранятся артефакты собранных приложений – чем больше приложение, тем более ощутимым становится выигрыш от применения данной технологии/подхода.

Решение, приведенное в диссертации, является универсальным для всех типов многостраничных сайтов, однако требует настройки алгоритмов кластеризации в соответствии с доменной областью. Отдельно необходимо упомянуть, что для достижения наилучшего результата необходимо экспериментировать с выбором мер расстояний, а иногда даже менять алгоритм. Никакого единого решения не существует.

Оптимизация самого процесса построения пакетов возможно потребует оптимизировать работу поиска в графе. В диссертации использовался алгоритм поиска в ширину, однако для некоторых продуктов поиск в глубину может оказаться намного быстрее. Но даже в таком виде: без настройки – данный алгоритм и сам подход может быть использован для любого программного продукта для увеличения скорости загрузки страницы.

Подход, изложенный в работе, позволяет уменьшить время загрузки страниц до предельно возможного для текущей доменной области за счет использования кэша браузеров и правильной загрузки оптимизированных сборок кода. Медленная загрузка страниц – первый враг посещаемости сайта. По результатам исследования, проведённого ещё в 2011 году агентством Forrester Research, 40% посетителей обычно покидают страницу, которая загружается более 3-х секунд. При долгой (больше 4-5 секунд) загрузке вероятность того, что посетитель интернет-магазина совершит какую-нибудь покупку, падает почти на 50%. Таким образом получается, что метрика времени первой загрузки страницы является важнейшей среди всех метрик, которые можно применить к веб продукту.

В решение заложен подход к повышению юзабилити без существенной перестройки системы, программного продукта. Это, наверное, самый важный аспект диссертации. Данный факт означает, что внедрение данного решения в программное средство любой компании, занимающейся веб приложениями, будет стоить не больших денег. Это выполнимо лишь при условии использования ES6 в клиентской части кода.

Система разработана на современной платформе с использованием последней версией JavaScript. На данный момент JavaScript самый используемый язык программирования в мире, используемый на разных платформах, таких как веб-браузеры, мобильные и VR устройства, а также веб-сервера. А еще и один из наиболее быстро развивающихся. Хоть большинство вебсайтов все еще используют ES5 и старые подходы к построению приложений, но для дальнейшего развития использование ES6, ES7 становится обязательным. А при росте самого сайта и увеличении страниц любая компания столкнется с проблемами оптимизации времени загрузки страницы. Наша компания столкнулась с этой проблемой одной из первых, и приведенное в этой работе решение является уникальным программным продуктом, которое решает все текущие проблемы с производительностью, временем построения кода и дальнейшем его использовании.

Данное решение было внедрено в линейку продуктов компании «Thomson Reuters», крупнейшим из которых является Westlaw/WestlawNext. Число активных пользователей ежедневно превышает 100000.

Работа была представлена на 54 СНТК студентов, магистрантов, аспирантов БГУИР в апреле 2018 г.

Список публикаций соискателя

[1-А] Дулина, Р.А. Оптимизация процесса загрузки страниц многофункциональных сайтов: критерии и практическая реализация / Р.А. Дулина // Тезисы к 54-й научной конференции. – Мн: БГУИР, 2018