ОПТИМИЗАЦИЯ ИСПОЛЬЗОВАНИЯ ДАННЫХ В ВЕБ-ПРИЛОЖЕНИИ ПРИ ЕГО РАЗРАБОТКЕ

Кривиков А.Д.
НИ ТПУ, ИШИТР, 8ПМ31, e-mail: adk26@tpu.ru

Аннотация

Данная работа посвящена исследованию способов оптимизации веб-приложения при работе с данными. В работе проводится обзор существующих подходов к оптимизации веб-приложений, основанных на работе с большим объемом информации, анализируются преимущества методов, оценивается их применимость в работе.

Ключевые слова: производительность, веб-приложение, оптимизация.

Введение

На протяжении нескольких лет происходит постоянный рост объема информации, с которой приходится работать в современных веб-приложениях. Этот феномен касается всех сфер деятельности компании, начиная с продажи собственных товаров и услуг и заканчивая корпоративными платформами, которые могут хранить и использовать большой объем данных. Согласно отчету платформы Demandsage [1], который основан на исследовании компании Statista [2], предполагается, в 2024 году во всем мире будет сгенерировано 147 зеттабайт данных. Это говорит об увеличении производства данных на 22,5 % во всем мире. Кроме того, к 2025 году во всем мире будет произведен 181 зеттабайт данных.

С появлением большего количества новых данных возникают проблемы, связанные с эффективным использованием и обработкой больших объемов информации: отображение данных, их чтение и запись. Оптимизация производительности веб-приложений становится первоочередной задачей, так как пользователи систем ожидают быстрого взаимодействия с веб-приложениями. Медленно работающие веб-приложения могут вызывать негативный опыт у пользователей системы, что может привести к потере клиентов, ухудшению репутации компании, отказа от использования приложения, а также к потере финансовых средств компании.

Так, согласно отчету Google Consumer Insights [3], 53 % посетителей мобильных сайтов покинут страницу, если она загружается более трех секунд. А в соответствии с экспериментом технологического агентства Portant [4], коэффициент конверсии веб-сайта падает в среднем на 4,2 % с каждой секундой загрузки, от нуля до пяти секунд.

В соответствии с вышеперечисленным, цель работы заключается в исследовании методов оптимизации веб-приложений при работе с большим объемом информации и поиске способов улучшения производительности систем.

Основная часть

Для обеспечения высокой производительности веб-приложений необходимо применять различные методы оптимизации. Данные методы могут применяться к различным аспектам веб-приложения, включая реализацию пользовательского интерфейса, проектирование архитектуры веб-системы, а также обеспечение взаимодействия компонентов приложения в сети.

При анализе были рассмотрены ключевые подходы, которые могут эффективно использоваться для оптимизации основных компонентов веб-приложений и которые формируют основу системы, влияют на ее функциональность и производительность.

1. Отображения элементов веб-страницы

— Lazy Loading (Ленивая загрузка) – специальный паттерн проектирования веб-приложения, который относится к стратегиям загрузки данных из базы данных или других источников по требованию, а не заранее. Этот способ способствует повышению производительности приложения, так как избегает загрузки и обработки данных, которые в данный момент не нужны. Это особенно полезно при работе с большими объемами данных или сложными структурами [5].

— Использование плейсхолдеров – специальных элементов, которые предназначены для временного отображения контента или данных на веб-странице до того момента, когда он будет

заполнен реальным содержимым. Использование плейсхолдеров может снизить нагрузку на сервера, так как они могут быть быстро загружены и отображены на клиентской стороне, пока обрабатываются и подготавливаются данные для отображения [6].

— Использование пагинации (постраничной загрузки данных) – процесса разбиения большого объема данных на отдельные страницы для удобства навигации и улучшения производительности веб-приложения. Пагинация позволяет загружать и отображать только часть данных на каждой странице, что позволяет снизить нагрузку на сервер.

— Минимизация анимационных эффектов. Анимации требуют дополнительных вычислений и ресурсов, особенно при работе с большим объемом данных. Избыточное использование анимации может увеличить нагрузку на браузер, что может сказаться на производительности веб-приложения в целом.

2. Данных веб-приложения

— Сжатие (компрессия) данных. Сжатие данных может существенно улучшить производительность веб-приложения при работе с большими объемами информации, так как данный процесс уменьшает размер данных путем применения различных алгоритмов, которые удаляют избыточную информацию или повторяющиеся элементы [7].

— Использование специальных форматов данных. Некоторые форматы данных могут сжимать информацию более эффективно, что позволяет сократить размер передаваемых или хранимых данных, уменьшая нагрузку на сеть и сервер.

— Кэширование данных. Кэширование позволяет временно хранить результаты вычислений, запросов к базе данных или другие данные для быстрого доступа в будущем. Существуют различные технологии кэширования, которые позволяют ускорить доступ к данным и снизить нагрузку на сервер, так как запросы к базе данных или сложные вычисления могут быть выполнены один раз, а результат сохранен в кэше [8].

— Асинхронные операции позволяют выполнять ресурсоемкие задачи параллельно, в фоновом режиме, что помогает улучшить производительность веб-приложения, ведь данный процесс уменьшает блокировку пользовательского интерфейса, что ведет к более быстрому доступу данным и улучшению пользовательского опыта.

3. Архитектура веб-приложения

— Микросервисная архитектура. Данный подход позволяет разбить приложение на отдельные микросервисы, каждый из которых отвечает за отдельную функциональность, что способствует лучшей масштабируемости, которая ведет к достижению оптимальной производительности. Кроме того, разделение функциональности на микросервисы позволяет избежать отказа системы в целом, ведь каждая часть системы может быть специализирована на обработке определенного типа данных и обслуживаться независимо [9].

— Serverless архитектура. В данной модели разработки программного обеспечения (модель облачных вычислений) облачный провайдер может выступать в роли веб-сервера, при этом разработчики могут создавать и развертывать приложения, не беспокоясь о управлении инфраструктурой серверов. Такой вид архитектуры позволяет обеспечить удобный доступ к данным и обработку потоков информации. Однако, стоит учитывать ограничения Serverless платформ, такие как отсутствие универсальной поддержки serverless-функций из-за поддержки одного языка программирования, или ограничение во времени запуска функций, которые не используются долгое время [10].

— CQRS (Command Query Responsibility Segregation) паттерн предлагает разделение операций записи и операций чтения в отдельные части системы. Данный способ позволяет справиться с потенциальной проблемой производительности, возникающей при попытке одновременно обрабатывать большой объем записей и запросов на чтение [11].

Кроме вышеперечисленных аспектов, необходимо также учитывать структуру хранимых и обрабатывающихся данных, что может повлиять на вид технологий хранения и обработки данных и тип баз данных. Также необходимо учитывать базовые методы оптимизации: оптимизация клиент-серверных запросов, запросов к базе данных, использование индексов в системе хранения данных и другие методы.

Описание тестирования

В ходе демонстрационного тестирования было оценено время загрузки страницы при выводе большого объема информации и время загрузки данных при использовании предложенных методов оптимизации.

Важно учитывать, что многие описанные методы стоит применять для определенных задач при разработке веб-приложения. Многие факторы могут зависеть от типа хранимых и отображаемых данных, архитектуры веб-приложения и других аспектов, которые влияют на систему и могут не позволять использовать определенные методы. Для проведения тестирования были выбраны методы оптимизации отображения данных на веб-странице и использование кэширования, так как эти методы просты в реализации и могут в первую очередь применятся в независимости от структуры и технологии веб-приложения.

Тестирования выполнялось с использованием следующих технологий веб-разработки:

— Django – фреймворк для веб-разработки на языке Python [12];

— Dash – фреймворк для разработки веб-приложений на языке Python, который разработан на библиотеках Flask, ReactJS и PlotlyJS [13];

— NodeJS - среда выполнения JavaScript на стороне сервера, которая позволяет создавать масштабируемые и быстрые веб-приложения [14];

— ReactJS – библиотека для создания пользовательских интерфейсов на языке программирования JavaScript [15];

— Redis – это ключ-значение хранилище данных с открытым исходным кодом, которое часто используется в качестве кэша, базы данных или брокера сообщений. [16].

Данные технологии были выбраны не для демонстрации их сравнения на практических примерах, а для исследования работоспособности и универсальности предложенных методов в разных аспектах веб-разработки, так как часть технологий используется на клиентской стороне, другая часть – на серверной.

Результаты тестирования

Результаты тестирования времени отображения элементов веб-страницы представлены на рисунках 1 и 2. В данном тестировании были использованы методы оптимизации отображения разного количества элементов (текстовые данные и изображения) веб-страницы при использовании разных технологий веб-разработки.

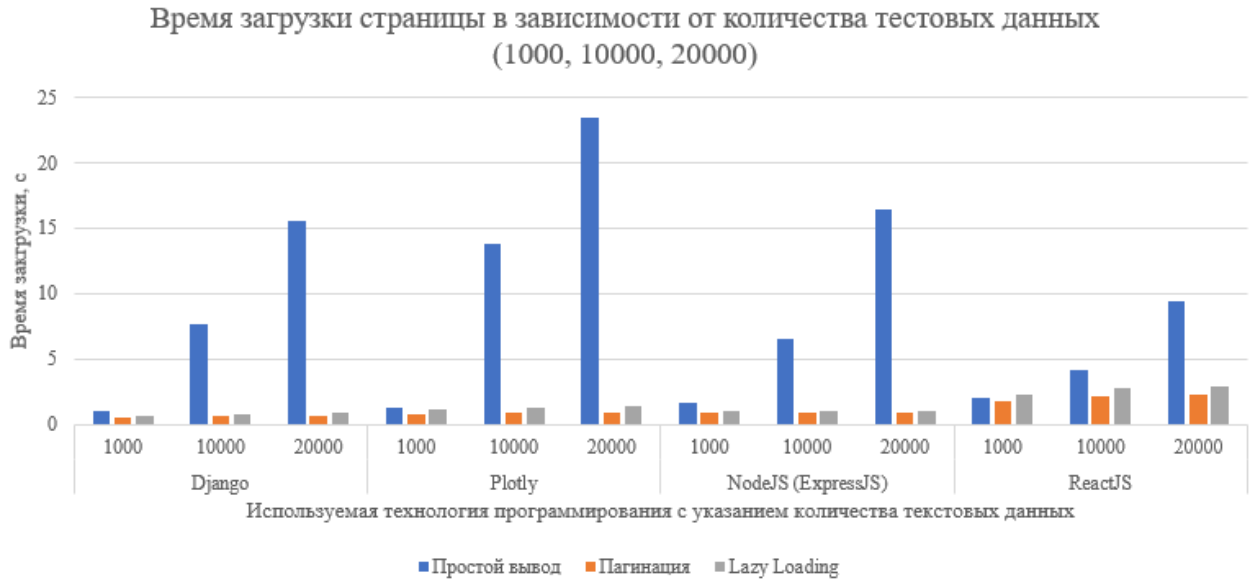

Рис. 6. Зависимости времени отображения текстовых данных на веб-странице от способа оптимизации

Рис. 7. Зависимость времени отображения изображений на веб-странице от способа оптимизации

На графиках видно, что при использовании небольшого количества данных методы пагинации и использования Lazy Loading не уменьшают время загрузки, так как они позволяют ограничить количество подгружаемого контента на веб-странице. Однако при увеличении выводимого количества данных предложенные методы позволяют сократить время загрузки страницы, что позволяет пользователю быстрее получить доступ к информации.

Результаты тестирования времени загрузки веб-страницы с использованием кэширования представлены на рисунках 3 и 4. Данное тестирование проводилось с использованием веб-фреймворка Django и технологии Redis.

Рис. 8. Зависимость времени загрузки от количества запросов к базе данных и применения кэширования

Рис. 9. Зависимость времени загрузки от применения кэширования части веб-страницы

График на рисунке 3 показывает, что общее время загрузки контента на веб-странице увеличивается с увеличением количества SQL запросов. При применении алгоритма кэширования время загрузки контента имеет примерно одно и тоже значение, вне зависимости от количества SQL запросов (для данной случая с максимальным числом SQL запросов – 10). Однако можно заметить, что при использовании одного SQL запроса время загрузки с применением кэширования больше, чем без него. Это связано с тем, что реализация алгоритма кэширования требует больше времени из-за дополнительной обработки данных и хранения кэша.

График на рисунке 4 показывает, что кэширование отображения также позволяет снизить время загрузки контента на веб-странице.

Таким образом, кэширование одного запроса имеет смысл, если он имеет долгое время выполнения. При большем количестве запросов кэширование помогает уменьшить время отображения контента на веб-странице при условии, что данные, возвращаемые запросами, не меняются. Кэширование отображения позволяет сохранять веб-страницу целиком или только ее часть, поэтому данный метод может быть эффективно использован для кэширования неизменяющихся частей веб-страницы.

Заключение

Оптимизация производительности веб-приложений при работе с большим объемом информации является важной задачей, так как это позволяет обеспечить более быстрый отклик и загрузку приложения. Кроме того, оптимизация позволяет улучшить пользовательский опыт, снизить нагрузку на серверы.

В результате исследования методов оптимизации были выделены основные методы, которые могут использоваться при разработке веб-приложений, предназначенных для работы с большими объёмами данных. Было проведено тестирование методов оптимизации отображения элементов веб-страницы и применения кэширования, которые показали действенность и работоспособность при использовании различных технологий разработки веб-приложений.

Список используемых источников

1. Big Data Statistics For 2024 (Growth, Market Size & More) // Demandsage.com: сайт. – URL: https://www.demandsage.com/big-data-statistics/ (дата обращения: 13.03.2024)

2. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025 // Statista.com: сайт. – URL: https://www.statista.com/statistics/871513/worldwide-data-created/ (дата обращения: 13.03.2024)

3. Consumer Insights. Mobile site load time statistics // Thinkwithgoogle.com: сайт. – URL: https://www.thinkwithgoogle.com/consumer-insights/consumer-trends/mobile-site-load-time-statistics/ (дата обращения: 13.03.2024)

4. Site Speed is (Still) Impacting Your Conversion Rate // Portent.com: сайт. – URL: https://www.portent.com/blog/analytics/research-site-speed-hurting-everyones-revenue.htm#:~:text=The first 5 seconds of, (between seconds 0-5) (дата обращения: 14.03.2024)

5. Lazy loading // developer.mozilla.org: сайт. – URL: https://developer.mozilla.org/ru/docs/Web/Performance/Lazy_loading (дата обращения: 15.03.2024)
6. Использование техники content placeholder // amorgunov.com: сайт. – URL: https://amorgunov.com/posts/2018-11-05-content-placeholder/ (дата обращения: 15.03.2024)
7. Compression // MDN Web Docs: сайт. – URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Compression (дата обращения: 15.03.2024)
8. Четыре уровня кэширования в сети: клиентский, сетевой, серверный и уровень приложения // tproger.ru: сайт. – URL: https://tproger.ru/translations/cache-levels-on-the-web (дата обращения: 15.03.2024)
9. Микросервисная архитектура: что это, кому подойдёт, с чего начать // Yandex.cloud: сайт. – URL: https://yandex.cloud/ru/blog/posts/2022/03/microservice-architecture (дата обращения: 15.03.2024)
10. Всё, что вы хотели знать о бессерверных технологиях, но боялись спросить // Yandex.cloud: сайт. – URL: https://yandex.cloud/ru/blog/posts/2023/11/about-serverless (дата обращения: 15.03.2024)
11. CQRS // Wikipedia: сайт. – URL: https://ru.wikipedia.org/wiki/CQRS (дата обращения: 15.03.2024)
12. Django documentation: сайт. – URL: https://docs.djangoproject.com/en/5.0/ (дата обращения: 15.03.2024)
13. Dash documentation: сайт. – URL: https://dash.plotly.com/ (дата обращения: 16.03.2024)
14. Node.js documentation: сайт. – URL: https://nodejs.org/docs/latest/api/ (дата обращения: 17.03.2024)
15. React.js documentation: сайт. – URL: https://react.dev/learn (дата обращения: 16.03.2024)
16. Redis documentation: сайт. – URL: https://redis.io/docs/ (дата обращения: 16.03.2024)