УДК 004.422.83

Александров А.А.

студент

Чувашский государственный университет им. И.Н. Ульянова

г. Чебоксары

МЕТОДЫ И СТРАТЕГИИ ОПТИМИЗАЦИИ ПРОИЗВОДИТЕЛЬНОСТИ ФРОНТЕНДА В ВЫСОКОНАГРУЖЕННЫХ СИСТЕМАХ ЭЛЕКТРОННОЙ КОММЕРЦИИ

Аннотация. В статье рассматриваются методы и стратегии оптимизации производительности фронтенда в высоконагруженных системах электронной коммерции. Актуальность исследования обусловлена прямым влиянием скорости загрузки интерфейсов на ключевые бизнес-метрики. Проведен анализ архитектурных подходов, включая микро-фронтенды и различные стратегии рендеринга, а также сравнительная оценка производительности популярных фреймворков с примерами реализации типового компонента. Особое внимание уделено современным метрикам Core Web Vitals.

Ключевые слова: производительность фронтенда, электронная коммерция, микро-фронтенды, стратегии рендеринга, Core Web Vitals.

METHODS AND STRATEGIES FOR FRONTEND PERFORMANCE OPTIMIZATION IN HIGH-LOAD E-COMMERCE SYSTEMS

Abstract. The article discusses methods and strategies for optimizing frontend performance in high-load e-commerce systems. The relevance of the research is driven by the direct impact of interface loading speed on key business metrics. The analysis covers architectural approaches, including micro-frontends and various rendering strategies, as well as a comparative evaluation of popular

frameworks' performance with code examples of a typical component implementation. Special attention is paid to modern Core Web Vitals metrics.

Keywords: frontend performance, e-commerce, micro-frontends, rendering strategies, Core Web Vitals.

Введение. Развитие электронной коммерции сопровождается ростом сложности пользовательских интерфейсов и требований к их производительности. Согласно исследованиям, задержка загрузки страницы даже на одну секунду может снижать конверсию на 7% [1]. Проблема усугубляется архитектурной сложностью современных фронтенд-систем: традиционные монолитные одностраничные приложения создают узкие места при развертывании и ограничивают масштабирование в периоды пиковых нагрузок. Кроме того, обилие фреймворков и библиотек при отсутствии систематизированных критериев их выбора приводит к тому, что разработчики часто руководствуются субъективными предпочтениями, а не объективными показателями производительности [2].

Целью настоящей статьи является систематизация и сравнительный анализ современных методов и стратегий оптимизации производительности фронтенда в высоконагруженных системах электронной коммерции, включая архитектурные подходы, стратегии рендеринга, особенности реализации на популярных фреймворках и метрики оценки эффективности.

Методы и исследования. Информация об применённых методах исследования. Для достижения цели работы был использован комплекс теоретических методов научного познания. В первую очередь применён метод систематического обзора и критического анализа научной литературы, посвящённой проблематике производительности фронтенда в электронной коммерции. Это позволило выявить ключевые направления исследований, установить степень изученности проблемы и определить нерешённые

вопросы. Также использовались сравнительно-сопоставительный метод для оценки различных архитектурных подходов (микро-фронтенды, стратегии рендеринга) и механизмов реактивности (виртуальный DOM, сигналы), а также метод контент-анализа для извлечения и систематизации данных из рецензируемых научных статей, диссертационных исследований и материалов конференций. Для иллюстрации теоретических выводов применялся метод выборочного рефакторинга — разработка и сравнение небольшого типового компонента (счётчика товаров в корзине) на разных фреймворках, что позволило наглядно продемонстрировать различия в подходах к управлению состоянием и обновлению интерфейса.

Целью настоящей статьи является систематизация и сравнительный анализ современных методов и стратегий оптимизации производительности фронтенда в высоконагруженных системах электронной коммерции, включая архитектурные подходы, стратегии рендеринга, особенности реализации на популярных фреймворках и метрики оценки эффективности.

Результаты оригинального авторского исследования. Выбор стратегии рендеринга является фундаментальным решением, определяющим воспринимаемую производительность приложения. В современной электронной коммерции используются четыре основных подхода: клиентский рендеринг (CSR), при котором браузер получает минимальный HTML и всю логику на JavaScript, что обеспечивает высокую интерактивность после загрузки, но страдает от медленного первого отображения контента; серверный рендеринг (SSR), где HTML генерируется на сервере для каждого запроса, что дает быстрый первый отклик, но увеличивает время до интерактивности и нагрузку на сервер; статическая генерация (SSG), обеспечивающая максимальную скорость загрузки за счет предварительной сборки HTML, но непригодная для динамического контента с часто

меняющимися ценами; и инкрементальная статическая регенерация (ISR) как гибридный подход, позволяющий обновлять статические страницы после сборки. Для высоконагруженных платформ оптимальным признается комбинированный подход: ISR для каталогов товаров и потоковый SSR для персонализированных страниц корзины и рекомендаций [4].

Микро-фронтенды представляют собой расширение микросервисной архитектуры на клиентскую часть, когда приложение разбивается на независимо разрабатываемые и развертываемые модули, соответствующие бизнес-возможностям: каталог, корзина, оформление заказа, личный кабинет. Юдин с соавторами отмечают, что основная цель такого подхода — облегчить поддержку и разработку больших приложений, над которыми работают разные команды [3]. Исследования Артюхина и соавторов показывают, что среди паттернов интеграции микро-фронтендов модульная федерация (Module Federation) обеспечивает наилучшие показатели производительности по сравнению с Single-SPA, веб-компонентами и iframe [2]. Однако Кожо с соавторами предупреждают, что внедрение микро-фронтендов вводит дополнительную сложность в инфраструктуру и требует зрелых практик DevOps. В их кейсе сопоставимых результатов можно было достичь и с монолитным фронтендом, но микро-фронтендовая архитектура оказалась наиболее удобной благодаря возможности повторного использования инфраструктуры микросервисов [5].

Ключевым трендом последних лет является переход от виртуального DOM к системам реактивности на основе сигналов. Виртуальный DOM, используемый в React и Vue, создает легковесное представление реального DOM в памяти, вычисляет различия и применяет минимальные обновления. Недостатком является избыточная нагрузка: даже при изменении одного небольшого значения может перевычисляться целое дерево компонентов.

Оучаиб в своем исследовании показывает, что сигналы, применяемые в Solid и внедряемые в новые версии Angular, обеспечивают точно-зернистую реактивность: компонент рендерится единожды, а сигналы напрямую подписывают DOM-узлы на изменения, обновляя только конкретный узел без перезапуска всего компонента. Это снижает накладные расходы и потребление памяти при росте сложности приложения [4].

Для иллюстрации различий в производительности рассмотрим реализацию типового компонента электронной коммерции — счетчика количества товара в корзине с отображением общей стоимости. В React с использованием хуков изменение количества вызовет повторный рендеринг всего компонента, как показано на рисунке 1:

```
import React, { useState } from 'react';

function CartItem({ price }) {
  const [quantity, setQuantity] = useState(1);

  const increase = () => setQuantity(q => q + 1);
  const total = price * quantity;

  return (
    <div>
      <p>Цена: ${price}</p>
      <p>Количество: {quantity}</p>
      <button onClick={increase}>+</button>
      <p>Итого: ${total}</p>
    </div>
  );
}
```

Рис.1. Фрагмент кода на React

В этом примере React после каждого нажатия кнопки выполняет диффинг виртуального DOM всего компонента, включая пересчет итоговой стоимости и всех дочерних элементов. В Solid с использованием сигналов

при изменении количества обновятся только текстовые узлы, как показано на рисунке 2:

```
import { createSignal } from 'solid-js';

function CartItem({ price }) {
  const [quantity, setQuantity] = createSignal(1);

  const increase = () => setQuantity(q => q() + 1);
  const total = () => price * quantity();

  return (
    <div>
      <p>Цена: {price}</p>
      <p>Количество: <span>{quantity()}</span></p>
      <button onClick={increase}>+</button>
      <p>Итого: <span>{total()}</span></p>
    </div>
  );
}
```

Рис. 2. Фрагмент кода на Solid

Сам компонент не перерендеривается, и виртуальный DOM не создается. Vue 3 предлагает компромиссный вариант, используя комбинацию виртуального DOM на уровне компонентов и системы реактивности на основе прокси, как показано на рисунке 3:

```vue
<template>
  <div>
    <p>Цена: {{ price }}</p>
    <p>Количество: {{ quantity }}</p>
    <button @click="increase">+</button>
    <p>Итого: {{ total }}</p>
  </div>
</template>

<script setup>
import { ref, computed } from 'vue';

const props = defineProps(['price']);
const quantity = ref(1);

const increase = () => quantity.value++;
const total = computed(() => props.price * quantity.value);
</script>
```

Рис. 3. Фрагмент кода на Vue 3

В результатах бенчмарков Оучаиба фиксируется преимущество фреймворков, реализующих точно-зернистую реактивность, по показателям требуемого объема памяти и по параметрам предсказуемости масштабирования производительности при усложнении структуры приложения в сравнении с традиционной парадигмой виртуального DOM, что подтверждается данными в источнике [4].

В контексте высоконагруженных систем значимость приобретают методы оптимизации загрузки ресурсов, при этом разделение кода на чанки с последующей загрузкой по требованию выступает как ключевой механизм, обеспечивающий подгрузку кода страницы оформления заказа исключительно при переходе к соответствующему интерфейсу, применение современных форматов изображений WebP и AVIF в сочетании с адаптивной стратегией их загрузки рассматривается как средство сокращения объема передаваемых данных и повышения эффективности визуализации, интеллектуальное кэширование ответов API посредством сервис-воркеров выступает как обеспечение работоспособности приложения при потере

сетевого соединения и как фактор снижения сетевой нагрузки, оптимизация критического пути рендеринга, включающая вынос блокирующего JavaScript и CSS из критического потока, фиксируется как мероприятие, существенно ускоряющее первоначальный отклик страницы [5].

Оценка производительности в современном электронном бизнесе осуществляется на основе метрик Core Web Vitals, влияние которых на поведенческие показатели пользователей [1], при этом показатель Largest Contentful Paint интерпретируется как мера времени загрузки основного контента и подлежит ограничению менее чем 2,5 секунд, показатель Interaction to Next Paint рассматривается как характеристика времени отклика на взаимодействие пользователя и подлежит ограничению менее чем 100 миллисекунд, показатель Cumulative Layout Shift интерпретируется как индикатор визуальной стабильности и подлежит поддержанию значения менее 0,1 в целях предотвращения случайных кликов, причем оптимизация перечисленных метрик рассматривается как фактор повышения конверсии, увеличения уровня лояльности пользователей и укрепления репутации бренда, что отражено в источнике [1].

Заключение. В результате осуществления анализа формулируются следующие выводы. Производительность фронтенда рассматривается как стратегический фактор успеха в электронной коммерции, обусловленный влиянием оптимизации клиентской части на метрики Core Web Vitals и через них на показатели конверсии и удержания пользователей. Архитектурный выбор детерминируется бизнес-потребностями, в рамках которых для крупных торговых площадок микро-фронтенды обеспечивают независимость разработки и масштабируемость, тогда как для проектов меньшего размера монолитная архитектура представляется более прагматичным решением. Фиксируется смещение парадигмы реактивности в сторону систем,

основанных на сигналах, которые демонстрируют улучшение показателей производительности по сравнению с традиционным виртуальным DOM. Оптимизация загрузки ресурсов понимается как комплексный процесс, требующий объединения подходов к рендерингу, стратегическому разделению кода и современным методам работы с изображениями. Выбор фреймворка подлежит обоснованию на основании предполагаемых сценариев нагрузки и сложности интерфейса, а не исключительно на основании популярности технологии, что обеспечивает выверенность технического решения применительно к бизнес-целям. Перспективные направления исследований интерпретируются как применение машинного обучения для предиктивной оптимизации интерфейса и интеграция edge-вычислений для обеспечения персонализированного рендеринга.

Список литературы

1. Шардаков Е. А. Совершенствование клиентского пути на веб-сайте интернет-провайдера для повышения конверсии и вовлеченности пользователей: магистерская диссертация : дис. – б. и., 2025.

2. Юдин А. В., Макиевский С. Е., Адышкин С. С., Грошева П. Ю. Анализ производительности и особенностей функционирования микрофронтендов // Computational Nanotechnology. 2024. Т. 11, № 1. С. 25–35.

3. Andi K., Ramirez D., Chango Sailema W. G. Comparativa de frameworks más usados de JavaScript mediante la creación de una aplicación web // Mikarimin. Revista Científica Multidisciplinaria. 2024. Vol. 10, № 3. P. 81–100.

4. Ouchaib J. Benchmarking Modern Frontend Frameworks: A Comparative Analysis of Rendering Performance: Master‘s thesis. Turin: Politecnico di Torino, 2025. 72 p.

5. Kojo R. H. H., Corte Real L. F., Ferreira R. C., Rosa T. O., Goldman A. Exploring Micro Frontends: A Case Study Application in E-Commerce // arXiv preprint arXiv:2506.21297. 2025.