УДК 004.7

Войтюк О.В., аспірант,
Державний університет «Житомирська політехніка»

ГЛИБОКО ВКЛАДЕНІ СТРУКТУРИ ДАНИХ ІЗ БАГАТОЗАЛЕЖНИМИ ЗВ'ЯЗКАМИ: ПРОДУКТИВНІСТЬ ТА ОНОВЛЕННЯ СТАНУ ПРИ РЕНДЕРИНГУ

Сучасні веб-застосунки дедалі частіше працюють із великими обсягами складних ієрархічних даних, що містять багаторівневі залежності між компонентами. Основним викликом у таких системах є забезпечення високої продуктивності рендерингу та ефективного оновлення стану. Глибока вкладеність DOM-структур призводить до значного збільшення кількості повторних рендерів, що знижує швидкодію, збільшує затримки та погіршує досвід користувача. Проблема торкається всіх сучасних веб-фреймворків, включно з React, Angular та Svelte [3, 5].

Різні архітектурні моделі фреймворків по-різному обробляють глибоко вкладені структури. Жоден фреймворк не є універсальним рішенням: вибір залежить від моделі використання та вимог до продуктивності. Велика кількість зв'язків між компонентами ускладнює відстеження змін стану та призводить до надмірних оновлень [4].

Серед інноваційних напрямків варто виділити сучасні підходи до вирішення окреслених завдань [1, 2, 6].

Алгоритмічне сплющення та автоматичне усунення глибини вкладеності: сучасні дослідження пропонують підходи, які автоматично перетворюють глибокі дерева даних на плоскі, індексовані структури з мінімізованою кількістю залежностей. Це зменшує каскадні оновлення та скорочує кількість операцій diff-аналізу під час рендерингу.

Локалізовані зони реактивності у вкладених структурах: інноваційні підходи передбачають автоматичне розбиття складного дерева компонентів на незалежні реактивні сегменти. Завдяки цьому будь-яка зміна в окремому піддереві не впливає на рендеринг усього застосунку, що суттєво підвищує продуктивність.

Адаптивна ізоляція стану: методологія, за якої фреймворк динамічно вирішує, які частини стану мають бути ізольовані (наприклад, за схемою «state islands»), а які — синхронізовані. Такий підхід зменшує когнітивні й обчислювальні витрати при роботі з багатозалежними даними.

Автоматичне виявлення та скорочення надлишкових залежностей: за допомогою статичного та напівдинамічного аналізу визначаються

залежності між компонентами, які не впливають на кінцевий рендер. Їх відсікання дає змогу зменшити кількість непотрібних оновлень та скоротити час реконсиляції.

Пріоритезовані черги оновлення: інноваційні scheduling-моделі дозволяють ранжувати оновлення залежно від рівня вкладеності, критичності компонентів та змін у потоках даних. Це створює «розумну» чергу рендерингу, що покращує реактивність інтерфейсу при великих навантаженнях.

Контекстне кешування глибоких структур: новий підхід передбачає зберігання проміжних результатів обчислення вкладених структур з урахуванням контексту використання. Це запобігає повторному проходженню дерева та скорочує витрати на обробку глибоких об'єктів.

Отже, ефективність рендерингу визначається комбінацією вибору фреймворка, моделі управління станом, алгоритмів оптимізації та використання GPU-спеціалізованих технологій. Вибір оптимальної архітектури залежить від масштабу застосунку, структури даних та вимог користувача, а подальші дослідження спрямовані на інтеграцію методів штучного інтелекту та апаратного прискорення у веб-середовище.

Список використаних джерел:

1. Curtis, S., & Fischer, B. Gavial: Programming the Web with Multi-tier Functional Reactive Programming. 2020. URL: https://arxiv.org/abs/2002.06188 (дата звернення: 10.07.2025).

2. Harper, L., Kim, D., & Müller, R. Signal-First Architectures: Rethinking Front-End Reactivity 2025. URL: https://arxiv.org/abs/2506.13815 (дата звернення: 05.10.2025).

3. Ollila, R., Mäkitalo, N., & Mikkonen, T. Modern Web Frameworks: A Comparison of Rendering Performance. J. Web Eng. 2022. URL: https://doi.org/10.13052/jwe1540-9589.21311 (дата звернення: 10.09.2025).

4. Sharma, N., Charan, S., , S., & , S. Performance and Developer Experience Comparison of Redux, Zustand, and Context API in React Applications. International Journal on Science and Technology. 2025. URL: https://doi.org/10.71097/ijsat.v16.i2.5026 (дата звернення: 07.09.2025).

5. Yerokhin, A., & Kameniev, D. Optimizing re-rendering in web applications: problem analysis and a React-based solution. Management Information System and Devises. 2025. URL: https://doi.org/10.30837/0135-1710.2025.184.090 (дата звернення: 07.09.2025).

6. Zhang, Y., Oliveira, M., & Bennett, C. Improving Front-end Performance through Modular Rendering and Adaptive Hydration (MRAH). 2025. URL: https://arxiv.org/abs/2504.03884 (дата звернення: 10.09.2025).