**EKSPLORASI PROBLEM YANG TERDAPAT DI WEB DEVELOPMENT**

***Complex Component Architecture and Design Patterns***

*Laporan ini disusun untuk memenuhi tugas mata kuliah Web Development*



Disusun oleh :

Muhammad Rizki N. 221524051

Kelompok 5

**JURUSAN TEKNIK KOMPUTER DAN INFORMATIKA**

**PROGRAM STUDI D4 TEKNIK INFORMATIKA**

**POLITEKNIK NEGERI BANDUNG**

**2024**

**Laporan Hasil Eksperimen: Peningkatan Performa Aplikasi React Menggunakan React.memo dan useMemo**

**1. Pendahuluan**

Aplikasi React yang besar atau memiliki banyak komponen dinamis sering kali menghadapi masalah performa, terutama ketika terdapat banyak data yang harus diproses dan dirender. Ketika setiap perubahan di aplikasi menyebabkan komponen yang tidak perlu dirender ulang, waktu respons aplikasi dapat meningkat. Salah satu cara untuk mengatasi masalah ini adalah dengan memanfaatkan optimasi menggunakan React.memo dan useMemo, yang dapat mengurangi render ulang komponen yang tidak perlu.

**2. Identifikasi Masalah**

Aplikasi dengan daftar item besar (misalnya 10.000 item) sering kali melakukan rendering ulang seluruh komponen meskipun perubahan hanya terjadi pada bagian kecil dari aplikasi. Hal ini mengakibatkan penurunan performa, terutama dalam hal waktu respons, yang berdampak pada pengalaman pengguna. Dalam eksperimen ini, kami mengidentifikasi bahwa tanpa optimasi, komponen yang tidak mengalami perubahan masih dirender ulang, sehingga terjadi pemborosan sumber daya.

**3. Tujuan Eksperimen**

Tujuan dari eksperimen ini adalah untuk:

* Membandingkan performa aplikasi React dengan dan tanpa optimasi menggunakan React.memo dan useMemo.
* Menilai pengaruh optimasi terhadap waktu rendering dan jumlah render ulang komponen.
* Memberikan solusi performa yang dapat diterapkan pada aplikasi React dengan dataset besar.

**4. Metodologi Eksperimen**

Eksperimen dilakukan dalam dua tahap:

* **Tanpa Optimasi**: Aplikasi React berisi daftar 10.000 item yang dirender menggunakan komponen ItemList dan ItemComponent tanpa menggunakan React.memo dan useMemo.
* **Dengan Optimasi**: Aplikasi yang sama dioptimasi menggunakan React.memo untuk mencegah rendering ulang komponen yang tidak mengalami perubahan dan useMemo untuk menyimpan hasil perhitungan (filtering) yang mahal.

Alat yang digunakan:

* **React.js** sebagai framework JavaScript.
* **Chrome DevTools** untuk analisis performa.
* **Data dummy** sebanyak 10.000 item yang dihasilkan secara acak.

Pengujian dilakukan dengan melakukan perubahan pada input pencarian (search term) dan kategori filter.

**5. Pelaksanaan Eksperimen**

1. **Tanpa Optimasi**:
   * Aplikasi menjalankan filtering sederhana untuk menampilkan daftar item berdasarkan pencarian dan kategori. Setiap perubahan pada input atau filter menyebabkan komponen ItemList dan seluruh ItemComponent dirender ulang, terlepas dari apakah item tersebut berubah atau tidak.
   * Hasil rendering dicatat di konsol untuk memeriksa seberapa sering komponen dirender ulang.
2. **Dengan Optimasi (Menggunakan React.memo dan useMemo)**:
   * React.memo diterapkan pada komponen ItemList dan ItemComponent untuk mencegah rendering ulang jika props tidak berubah.
   * useMemo digunakan untuk menyimpan hasil filtering sehingga perhitungan ulang hanya dilakukan ketika input pencarian atau kategori filter berubah.

**6. Analisis Hasil Eksperimen**

**a. Tanpa Optimasi:**

* Setiap kali pengguna mengetik di input pencarian atau mengganti kategori filter, seluruh komponen ItemComponent dirender ulang. Hal ini termasuk item yang tidak relevan dengan filter atau pencarian.
* Untuk 10.000 item, waktu rendering semakin lama seiring dengan jumlah perubahan yang dilakukan, mengakibatkan penurunan kinerja aplikasi.

**b. Dengan Optimasi (Menggunakan React.memo dan useMemo):**

* Setelah menerapkan React.memo pada komponen ItemList dan ItemComponent, komponen hanya dirender ulang ketika ada perubahan pada props.
* Hasil filtering menggunakan useMemo disimpan dan dihitung ulang hanya jika input pencarian atau kategori berubah.
* Jumlah render ulang komponen ItemComponent berkurang secara drastis, hanya komponen yang relevan dengan pencarian dan kategori yang dirender ulang.
* Dari analisis menggunakan Chrome DevTools, terdapat pengurangan yang signifikan dalam waktu rendering setelah optimasi diterapkan.

**c. Perbandingan Kinerja:**

* **Tanpa Optimasi:** Setiap perubahan pada input menyebabkan lebih dari 10.000 render ulang komponen ItemComponent.
* **Dengan Optimasi:** Hanya item yang relevan dengan filter yang dirender ulang, mengurangi jumlah render menjadi hanya beberapa komponen yang sesuai dengan pencarian.

**7. Kesimpulan**

Berdasarkan eksperimen yang dilakukan, penerapan React.memo dan useMemo dalam aplikasi React yang memproses daftar besar item secara signifikan meningkatkan performa aplikasi. Jumlah render ulang komponen dapat diminimalkan, dan hasil perhitungan mahal seperti filtering dapat disimpan untuk mencegah perhitungan ulang yang tidak diperlukan.

Penerapan optimasi ini sangat relevan untuk aplikasi yang menangani data besar dan dinamis, seperti e-commerce atau aplikasi manajemen data, di mana responsivitas dan performa menjadi prioritas utama.

**8. Saran**

* **Gunakan React.memo secara selektif**: Tidak semua komponen perlu dioptimasi. Fokus pada komponen yang sering dirender ulang atau komponen dengan dataset besar.
* **Gunakan useMemo untuk perhitungan mahal**: Optimasi dengan useMemo dapat meningkatkan performa jika terdapat operasi filtering atau sorting yang berat.
* **Analisis performa secara berkala**: Gunakan alat seperti Chrome DevTools atau React Profiler untuk memantau kinerja aplikasi Anda dan mengidentifikasi potensi masalah performa.