Performance Optimization

🔹 What is Performance Optimization?

Performance Optimization মানে হলো:

👉 আপনার application-কে **fast**, **responsive**, আর **scalable** করা
👉 কম **CPU / Memory / Network** খরচে বেশি কাজ করানো

Simple ভাষায়:

- Page দ্রুত load হবে
- UI lag করবে না
- Server কম চাপ নেবে
- User happy 😄

🔹 Why Performance Optimization is IMPORTANT?

User Impact

- 1s delay → conversion 7% কমে
- Slow UI → user churn

Engineering Impact

- Less infra cost 💰
- High traffic handle করতে পারবে
- FAANG interview-এ strong signal

🔹 Types of Performance Optimization

1. Frontend Performance

2. Backend Performance

3. Network Performance

4. Database Performance

আমরা এক এক করে দেখি 👇

🟦 1. Frontend Performance Optimization (React / Next.js)

🔸 1. Avoid Unnecessary Re-renders

React re-render slow করে

❌ Bad

```
const Component = () => {
  const obj = {};
  return <Child data={obj} />;
};
```

✅ Good

```
const obj = useMemo(() => ({}), []);
```

🔸 2. Memoization

- React.memo
- useMemo
- useCallback

```
const Button = React.memo(({ onClick }) => {
  return <button onClick={onClick}>Click</button>;
});
```

✓ Prevents unnecessary render

🔸 3. Code Splitting & Lazy Loading

Only load what you need

```
const Dashboard = lazy(() => import("./Dashboard"));
```

Next.js:

```
const HeavyComponent = dynamic(() => import('./Heavy'), {
  ssr: false
});
```

🔸 4. Image Optimization (VERY IMPORTANT)

Next.js:

```
<Image src="/img.png" width={400} height={300} />
```

✓ Lazy loading
✓ Responsive images
✓ Smaller size

🔸 5. Virtualization (Large Lists)

Render only visible items

- react-window
- react-virtualized

<List height={400} itemCount={10000} itemSize={35} />

🟦 2. Backend Performance Optimization

🔸 1. Caching (Most Important)

Types:

- In-memory cache (Redis)
- CDN cache
- Browser cache

Client → Cache → DB

✓ Faster response
✓ DB load কমে

🔸 2. Async & Non-blocking Code

Node.js example:

❌ Blocking:

fs.readFileSync()

✅ Non-blocking:

fs.readFile()

🔸 3. Load Balancing

- Horizontal scaling
- Distribute traffic

Types:

- Round Robin
- Least Connection

🟦 3. Network Performance Optimization

🔸 1. Reduce Payload Size

- Gzip / Brotli compression
- Remove unused fields

🔸 2. CDN

Static assets served closer to user

User → CDN → Server

🔸 3. HTTP/2 & HTTP/3

- Multiplexing
- Faster connections

🟦 4. Database Performance Optimization

🔸 1. Indexing (MOST ASKED)

CREATE INDEX idx_user_email ON users(email);

✓ Fast read
❌ Slight slow write

🔸 2. Query Optimization

❌

SELECT * FROM users;

✅

SELECT name, email FROM users WHERE id=1;

🔸 3. Read Replicas

- Read traffic → replicas
- Write → primary DB

🟦 Next.js Specific Optimizations (🔥 Interview Gold)

✓ Server Components
✓ Static Generation (SSG)
✓ Incremental Static Regeneration (ISR)
✓ Edge Functions
✓ Prefetching (<Link />)

🔹 Metrics You Should Know (FAANG)

- LCP (Largest Contentful Paint)

- FID / INP
- CLS
- TTFB

Tools:

- Lighthouse
- Web Vitals
- Chrome DevTools

🔹 **Real-life Analogy 🚗**

Performance Optimization =
🚦 Traffic system optimization

- Flyover = cache
- Traffic police = load balancer
- Express lane = CDN

🔹 **Interview One-Liner (🔥)**

“I optimize performance by reducing unnecessary work, caching aggressively, and pushing computation closer to the user.”

🔹 **Summary (TL;DR)**

✓ Avoid unnecessary renders
✓ Use memoization wisely
✓ Cache everywhere
✓ Optimize DB queries
✓ Use CDN & compression
✓ Measure before optimizing

পারফরম্যান্স অপ্টিমাইজেশান (Performance Optimization) হলো একটি সিস্টেম বা অ্যাপ্লিকেশনকে এমনভাবে উন্নত করা যাতে এটি দ্রুত কাজ করে, কম রিসোর্স (যেমন: RAM, CPU) ব্যবহার করে এবং ব্যবহারকারীকে একটি স্মুথ অভিজ্ঞতা প্রদান করে।

নিচে সফটওয়্যার এবং ওয়েব অ্যাপ্লিকেশনের ক্ষেত্রে পারফরম্যান্স অপ্টিমাইজেশানের মূল বিষয়গুলো বিস্তারিত আলোচনা করা হলো:

১. ফ্রন্টএন্ড অপ্টিমাইজেশান (Frontend Optimization)

ব্যবহারকারী সরাসরি যা দেখেন, তার গতি বাড়ানোই এখানে মূল লক্ষ্য।

- **Asset Compression:** আপনার ইমেজ, CSS এবং JavaScript ফাইলগুলোকে কম্প্রেস বা ছোট করা। বড় সাইজের ছবি লোড হতে সময় নেয়, তাই WebP ফরম্যাট ব্যবহার করা ভালো।
- **Lazy Loading:** পেজ লোড হওয়ার সাথে সাথেই সব ইমেজ বা ভিডিও লোড না করে, ইউজার যখন স্ক্রল করে সেই জায়গায় পৌঁছাবে তখন লোড করা।

- **Minification:** কোড থেকে অপ্রয়োজনীয় স্পেস, কমেন্ট এবং বড় ভ্যারিয়েবল নাম সরিয়ে ফাইলের সাইজ কমানো।
- **Caching:** ব্রাউজার ক্যাশিং ব্যবহার করা যাতে বারবার একই ফাইল সার্ভার থেকে ডাউনলোড করতে না হয়।

২. ব্যাকএন্ড এবং ডাটাবেস অপ্টিমাইজেশান

সার্ভার সাইডে প্রসেসিং স্পিড বাড়ানো এবং ডাটাবেস কোয়েরি দ্রুত করা।

- **Database Indexing:** ডাটাবেসে ইনডেক্স ব্যবহার করলে ডাটা খুঁজে পেতে অনেক কম সময় লাগে। এটি একটি বইয়ের 'Index' পেজের মতো কাজ করে।
- **Query Optimization:** অপ্রয়োজনীয় ডাটা ফেচ না করা। যেমন: SELECT * ব্যবহার না করে শুধুমাত্র প্রয়োজনীয় কলামগুলো সিলেক্ট করা।
- **Caching with Redis:** বারবার ডাটাবেসে হিট না করে ঘনঘন প্রয়োজন হয় এমন ডাটাগুলো **Redis** বা **Memcached**-এর মতো ইন-মেমোরি স্টোরেজে রাখা।
- **Load Balancing:** যখন অনেক ইউজার একসাথে অ্যাপ ব্যবহার করে, তখন ট্রাফিককে একাধিক সার্ভারে ভাগ করে দেওয়া।

৩. কোড লেভেল অপ্টিমাইজেশান

প্রোগ্রামিং করার সময় কিছু নিয়ম মেনে চললে কোড দ্রুত রান করে।

- **Algorithm Efficiency:** সঠিক ডেটা স্ট্রাকচার এবং অ্যালগরিদম ব্যবহার করা। উদাহরণস্বরূপ, বড় ডেটার ক্ষেত্রে ও (n^2) এর বদলে ও (n log n) অ্যালগরিদম ব্যবহার করা।
- **Avoid Memory Leaks:** কোডে এমন কোনো অবজেক্ট রাখা যাবে না যা আর প্রয়োজন নেই কিন্তু মেমোরি দখল করে আছে।

- **Asynchronous Programming:** ভারী কোনো কাজ (যেমন ফাইল আপলোড বা ইমেইল পাঠানো) করার সময় মেইন থ্রেড ব্লক না করে ব্যাকগ্রাউন্ডে বা এসিনক্রোনাসলি করা।

৪. পারফরম্যান্স মাপার টুলস

অপ্টিমাইজ করার আগে জানতে হবে সমস্যা কোথায়। এর জন্য কিছু জনপ্রিয় টুলস হলো:

- **Google Lighthouse:** ওয়েব পেজের স্পিড এবং এসইও চেক করার জন্য।
- **Chrome DevTools:** নেটওয়ার্ক রিকোয়েস্ট এবং মেমোরি ইউসেজ দেখার জন্য।
- **New Relic / Datadog:** সার্ভার সাইড মনিটরিংয়ের জন্য।