Token bucket

Rate Limiting হলো একটি সিকিউরিটি পদ্ধতি যা কোনো ইউজার বা আইপি অ্যাড্রেস কতবার একটি এপিআই (API) কল করতে পারবে তা নিয়ন্ত্রণ করে। আর **Token Bucket** হলো এই রেট লিমিটিং করার সবচেয়ে জনপ্রিয় এবং সহজ একটি অ্যালগরিদম।

নিচে এটি কীভাবে কাজ করে তা বিস্তারিত আলোচনা করা হলো:

১. Token Bucket অ্যালগরিদম কী?

কল্পনা করুন একটি বালতি (Bucket), যার ভেতরে নির্দিষ্ট সংখ্যক টোকেন রাখা যায়। প্রতিটি টোকেন একটি পারমিশন বা অনুমতির মতো।

- **বালতির ধারণক্ষমতা (Capacity):** বালতিতে সর্বোচ্চ কতটি টোকেন থাকতে পারবে (যেমন: ১০টি)।
- **টোকেন জেনারেটর:** প্রতি সেকেন্ডে বালতিতে একটি নির্দিষ্ট হারে নতুন টোকেন যোগ হয় (যেমন: প্রতি সেকেন্ডে ২টি)।
- **অনুরোধ প্রসেসিং:** যখনই কোনো ইউজার একটি রিকোয়েস্ট পাঠায়, সিস্টেম বালতি থেকে একটি টোকেন বের করে নেয়।

২. এটি কীভাবে কাজ করে? (Step-by-Step)

১. টোকেন আছে কি না দেখা: যখন একটি রিকোয়েস্ট আসে, সিস্টেম চেক করে বালতিতে কোনো টোকেন আছে কি না।

২. রিকোয়েস্ট গ্রহণ: যদি টোকেন থাকে, একটি টোকেন ফেলে দেওয়া হয় এবং রিকোয়েস্টটি সার্ভারে পাঠানো হয়।

৩. রিকোয়েস্ট প্রত্যাখ্যান: যদি বালতি খালি থাকে (কোনো টোকেন নেই), তবে রিকোয়েস্টটি রিজেক্ট করা হয় (সাধারণত HTTP 429 Too Many Requests এরর দিয়ে)।

৪. রিফিলিং: সময় বাড়ার সাথে সাথে বালতিতে আবার নতুন টোকেন জমা হতে থাকে যতক্ষণ না এটি পূর্ণ হয়।

৩. কেন Token Bucket সেরা? (Burst Traffic)

এই অ্যালগরিদমের সবচেয়ে বড় সুবিধা হলো এটি **Burst Traffic** হ্যান্ডেল করতে পারে।

উদাহরণ: আপনার রিফিল রেট প্রতি সেকেন্ডে ২ টি, কিন্তু বালতির সাইজ ১০টি। এখন যদি ১ সেকেন্ডের জন্য কোনো রিকোয়েস্ট না আসে, তবে বালতি পূর্ণ হয়ে ১০টি টোকেন থাকবে। পরের সেকেন্ডে যদি হুট করে ১০ জন ইউজার আসে, তবে তারা সবাই সাথে সাথে একসেস পাবে। কিন্তু ১১ নম্বর জন আটকে যাবে।

৪. সুবিধা ও অসুবিধা

সুবিধা	**অসুবিধা**
এটি মেমোরি খুব কম খরচ করে (শুধু টোকেন কাউন্ট সেভ করতে হয়)।	এটি রিয়েল-টাইমে মেমোরিতে কাউন্ট আপডেট করে বলে ডিস্ট্রিবিউটেড সিস্টেমে (যেমন অনেকগুলো সার্ভারে) সিঙ্ক করা কিছুটা কঠিন।
বার্স্ট ট্রাফিক (হঠাৎ অনেক রিকোয়েস্ট) অনুমোদন করে।	
এটি ইমপ্লিমেন্ট করা খুব সহজ।	

৫. বাস্তব জীবনের উদাহরণ

একটি এমজমেন্ট পার্কের কথা ভাবুন। পার্কের গেটে একজন লোক প্রতি ৫ মিনিট পর পর ১০টি করে টিকিট (টোকেন) বাক্সে রাখছেন।

- যদি গেটে অনেক মানুষ জমে থাকে, তারা টিকিট পাওয়া মাত্র ভেতরে ঢুকবে।
- কিন্তু টিকিট শেষ হয়ে গেলে তাদের পরবর্তী ৫ মিনিট অপেক্ষা করতে হবে নতুন টিকিটের জন্য।

৬. কোডিং উদাহরণ (Pseudo-code)

আপনি যদি এটি কোড দিয়ে বুঝতে চান, তবে লজিকটি হবে এরকম:

JavaScript

```
let tokens = capacity;
let lastRefillTime = Date.now();

function handleRequest() {
  refill(); // বর্তমান সময় অনুযায়ী টোকেন যোগ করো
  if (tokens > 0) {
    tokens--;
    return "Request Allowed";
  } else {
    return "Rate Limit Exceeded";
  }
}
```

সারাংশ:

Token Bucket অ্যালগরিদম আপনার সার্ভারকে অতিরিক্ত চাপের হাত থেকে বাঁচায় এবং নিশ্চিত করে যে কোনো একজন ইউজার যেন পুরো সিস্টেমের ব্যান্ডউইথ দখল করে না নেয়।

🚦 Rate Limiting কী?

Rate Limiting মানে—

একজন user / IP / API key
নির্দিষ্ট সময়ের মধ্যে **নির্দিষ্ট সংখ্যক request** করতে পারবে

কেন দরকার?

✓ Abuse prevention
✓ DDoS protection
✓ Fair usage
✓ Server overload ঠেকানো

🪣 Token Bucket Algorithm কী?

Token Bucket হলো **সবচেয়ে popular + practical rate limiting algorithm**।

Core idea:

- একটা **bucket** আছে
- bucket-এ **token জমা হয় সময়ের সাথে**
- প্রতিটা request → **১টা token খরচ**
- token **না** থাকলে → request reject ❌

🧠 Visualization

Token Bucket (capacity = 10)

🪙 🪙 🪙 🪙 🪙 🪙 🪙 🪙 🪙 🪙

↑

Refill rate

⚙️ How it works (Step by Step)

Example config:

Bucket capacity = 10 tokens

Refill rate = 1 token / second

Scenario:

1. User first request → token আছে ✅
2. Burst 10 request → সব pass ✅
3. 11th request → token নাই ❌
4. 1 second wait → 1 token যোগ
5. আবার request → pass ✅

💥 Burst traffic handle করতে পারে কেন?

👉 Token জমা থাকে
👉 sudden burst হলেও allow করে

এটাই Token Bucket এর সবচেয়ে বড় advantage ⭐

🧪 Pseudocode (Conceptual)

```
if tokens > 0:
  tokens--
  allow request
else:
  reject request
```

🧠 Token Bucket vs Leaky Bucket

Topic	Token Bucket	Leaky Bucket
Burst allow	✅ Yes	❌ No
Traffic smoothing	Medium	High
Flexibility	High	Low
Real-world use	API	Network shaping

🧠 Token Bucket vs Fixed Window

Issue	Fixed Window	Token Bucket
Burst problem	❌ Yes	✅ Solved
Accuracy	Low	High
Fairness	Medium	High

🧠 Where Token Bucket is used?

- **AWS API Gateway**
- **NGINX**
- **Kong API Gateway**
- **Envoy**
- **Redis-based rate limiter**

🧱 Distributed Token Bucket (Real System)

Problem:

Multiple server হলে?

Server 1

Server 2

Server 3

Solution:

👉 **Central store** (Redis)

Client

↓

Load Balancer

↓

API Server

↓

Redis (Token Bucket State)

⚠️ **Challenges**

❌ Race condition
❌ Clock sync
❌ Redis latency

🛡️ **Solutions**

✓ Lua script in Redis
✓ Atomic operations
✓ Local + global rate limit

📌 **HTTP Status Code**

429 Too Many Requests

Retry-After: 5

🧠 Interview Golden Lines ⭐

“Token Bucket allows controlled bursts while maintaining average rate limits.”

“It is preferred over fixed window for API rate limiting due to better fairness.”

📝 Memory Trick

Token = Permission

No token = No entry 🚫

🔥 Example Use Case

Login API:

10 requests / minute

Burst allowed = 10