Ring concept

Consistent Hashing-এর মূল ভিত্তি হলো **Ring Concept** বা বৃত্তাকার ধারণা। সাধারণ হ্যাশিং যেখানে একটি লিনিয়ার বা সোজা লাইনের মতো কাজ করে, সেখানে কনসিস্টেন্ট হ্যাশিং ডাটা এবং সার্ভারকে একটি কাল্পনিক বৃত্তে সাজায়।

নিচে রিং কনসেপ্টের বিস্তারিত আলোচনা করা হলো:

১. হ্যাশ রিং (The Hash Ring)

কল্পনা করুন একটি বিশাল বৃত্ত, যার শুরু 0 থেকে এবং শেষ $2^{32}-1$ এ। এই বৃত্তের পরিধি বরাবর প্রতিটি পয়েন্ট একটি হ্যাশ ভ্যালু নির্দেশ করে। যেহেতু এটি বৃত্তাকার, তাই শেষ পয়েন্টটি আবার প্রথম পয়েন্টের সাথে মিলে যায়। একেই বলা হয় **Hash Ring**।

২. সার্ভার ম্যাপিং (Mapping Servers on the Ring)

প্রথমে আমরা আমাদের সার্ভারগুলোকে (যেমন: S_1, S_2, S_3) হ্যাশ করে এই রিংয়ের বিভিন্ন অবস্থানে বসাই।

- প্রতিটি সার্ভারের নাম বা আইপি অ্যাড্রেসকে একটি হ্যাশ ফাংশন (যেমন MD5 বা SHA-1) দিয়ে রান করানো হয়।
- রেজাল্ট অনুযায়ী সার্ভারগুলো রিংয়ের নির্দিষ্ট পয়েন্টে জায়গা করে নেয়।

৩. ডাটা ম্যাপিং (Mapping Data/Keys on the Ring)

একইভাবে, আমাদের ডাটা বা কি (Keys) গুলোকেও একই হ্যাশ ফাংশন দিয়ে রিংয়ের ওপর বসানো হয়। প্রতিটি ডাটা রিংয়ের কোনো না কোনো পয়েন্টে অবস্থান নেয়।

৪. ডাটা অ্যাসাইনমেন্ট (Data Assignment Logic)

এখন প্রশ্ন হলো, কোন ডাটা কোন সার্ভারে সেভ হবে? রিং কনসেপ্টের নিয়মটি খুব সহজ:

- একটি ডাটা যে পয়েন্টে আছে, সেখান থেকে সে ঘড়ির কাঁটার দিকে (**Clockwise**) ঘুরতে শুরু করবে।
- ঘুরতে ঘুরতে সে প্রথম যে সার্ভারটির দেখা পাবে, সেই সার্ভারেই ডাটাটি জমা হবে।

৫. রিং কনসেপ্টের সুবিধা (কেন এটি শ্রেষ্ঠ?)

ক. সার্ভার রিমুভ করলে (Server Failure)

যদি একটি সার্ভার (যেমন S_2) নষ্ট হয়ে যায়, তবে শুধু ওই সার্ভারে থাকা ডাটাগুলো রিং বরাবর সামনে এগিয়ে যাবে এবং তার পরের সার্ভারে (S_3) গিয়ে জমা হবে। বাকি সার্ভারগুলোর ডাটা আগের জায়গায় একদম স্থির থাকবে।

ফলাফল: মাত্র $1/n$ পরিমাণ ডাটা রি-লোকেট করতে হয়।

খ. সার্ভার যোগ করলে (Server Addition)

যদি নতুন একটি সার্ভার রিংয়ের কোনো জায়গায় যোগ করা হয়, তবে শুধুমাত্র সেই নতুন সার্ভারের ঠিক পেছনে থাকা কিছু ডাটা নতুন সার্ভারে চলে আসবে। পুরো সিস্টেমের কোটি কোটি ডাটা সরানোর প্রয়োজন হবে না।

৬. ভার্চুয়াল নোড (Virtual Nodes)

কখনও কখনও রিংয়ের ওপর সার্ভারগুলো সমান দূরত্বে থাকে না, ফলে কোনো সার্ভারে বেশি ডাটা চলে যায় (একে **Hotspot** বলে)। এই সমস্যা সমাধানে প্রতিটি ফিজিক্যাল সার্ভারের অনেকগুলো কাল্পনিক কপি বা **Virtual Nodes** রিংয়ের বিভিন্ন জায়গায় ছড়িয়ে দেওয়া হয়। এতে ডাটা ডিস্ট্রিবিউশন একদম নিখুঁত এবং সমান হয়।

সারাংশ:

রিং কনসেপ্ট মূলত একটি ডাইনামিক ম্যাপিং সিস্টেম। এটি নিশ্চিত করে যে সার্ভার আসা-যাওয়ার ফলে যেন পুরো সিস্টেমের চেইন ভেঙে না যায়, বরং শুধুমাত্র ক্ষতিগ্রস্ত অংশটুকুই রি-অ্যাডজাস্ট হয়।

🧠 Ring Concept কী?

👉 Hash space-কে একটা circle (ring) হিসেবে কল্পনা করা

- Start = 0
- End = max hash value
- End connects back to start

0 ———————— MAX

↑__________________↓

📌 তাই একে **hash ring** বলা হয়।

🔢 1. Hash Space

ধরি:

0 → 2^{32} - 1

সবকিছু (server, key) hash হয়ে এই space-এর কোথাও বসে।

🖥️ 2. Servers on the Ring

Each server:

hash(serverId) → position on ring

Example:

Server A → 20°

Server B → 120°

Server C → 250°

B(120)

●

A(20) ● ● C(250)

🔑 3. Keys on the Ring

Each key:

hash(key) → position

Example:

Key K1 → 30°

Key K2 → 200°

📍 4. Assignment Rule (MOST IMPORTANT)

🔥 A key goes to the first server found clockwise

Example:

- K1 (30°) → Server B (120°)
- K2 (200°) → Server C (250°)

📌 Clockwise = always move forward, wrap if needed.

🔁 5. Wrap-Around Concept

Key at:

300°

Next clockwise server?

→ Server A (20°) ← wrap-around

📌 Ring is circular.

➕ 6. Adding a Server (MAGIC ✨)

Before:

A ———— B ———— C

Add D between A & B:

A —— D —— B ———— C

📌 Only keys between **A → D** move to D
❌ Others unchanged

➖ 7. Removing a Server

If B fails:

Keys of B → next clockwise server (C)

✓ Minimal data movement
✓ Graceful degradation

🧩 8. Virtual Nodes (Ring Enhancement)

To fix uneven load:

Server A → A1, A2, A3

Server B → B1, B2, B3

They are placed at different positions.

📌 This smooths distribution.

⚖️ Ring vs Modulo Hashing

Aspect	**Ring**	**Modulo**
Node change	Minimal movement	Full reshuffle
Failure handling	Local	Global
Scalability	High	Poor

🏆 FAANG Interview Explanation (Say This 💎)

“The ring concept maps both nodes and keys into the same hash space arranged as a circle. Each key is assigned to the nearest clockwise node, which ensures that adding or removing nodes causes minimal redistribution.”

🧠 One-Line Memory Hook

Ring = circular hash space with clockwise ownership