SYSTEM DESIGN

CAP Theorem | Brewer's Theorem

CAP Theorem

কোনো একটি distributed system একসাথে তিনটি জিনিস – Consistency, Availability, আর Partition Tolerance – সবগুলো একসাথে পুরোপুরি দিতে পারে না। এর মধ্যে সর্বোচ্চ দুইটা একসাথে বজায় রাখা সম্ভব।

সহজ বাংলায় যদি বলি,
যখন আপনার অ্যাপ বা সার্ভার একাধিক জায়গায় ছড়িয়ে থাকে (distributed system),
তখন আপনি একসাথে সব সুবিধা পাবেন না –
আপনাকে ৩টার মধ্যে যেকোনো ২টার সুবিধা বেছে নিতে হবে।

কিভাবে CAP Theorem এলো?

2000 সালে Eric Brewer নামক একজন computer scientist এই theorem প্রস্তাব করেন, যেটা পরে 2002 সালে Seth Gilbert এবং Nancy Lynch এটি প্রমাণ করেন।

CAP THEOREM এর প্রধান তিনটি জিনিস তাহলে কি কি :

✅ Consistency (একরকম ডেটা):

System এর সব নোড/সার্ভার এ একই data থাকবে। আপনি যেখান থেকেই data পড়েন না কেন, একই result পাবেন।

✅ Availability (সবসময় রেসপন্স দিবে):

সার্ভার কখনোই "down" দেখাবে না – রেসপন্স দিতেই থাকবে।

✅ Partition Tolerance (নেটওয়ার্ক ভেঙে গেলেও সিস্টেম চালু থাকবে):

সার্ভারের মধ্যে যোগাযোগ বিচ্ছিন্ন হলেও সিস্টেম চলবে বন্ধ হবেনা ।

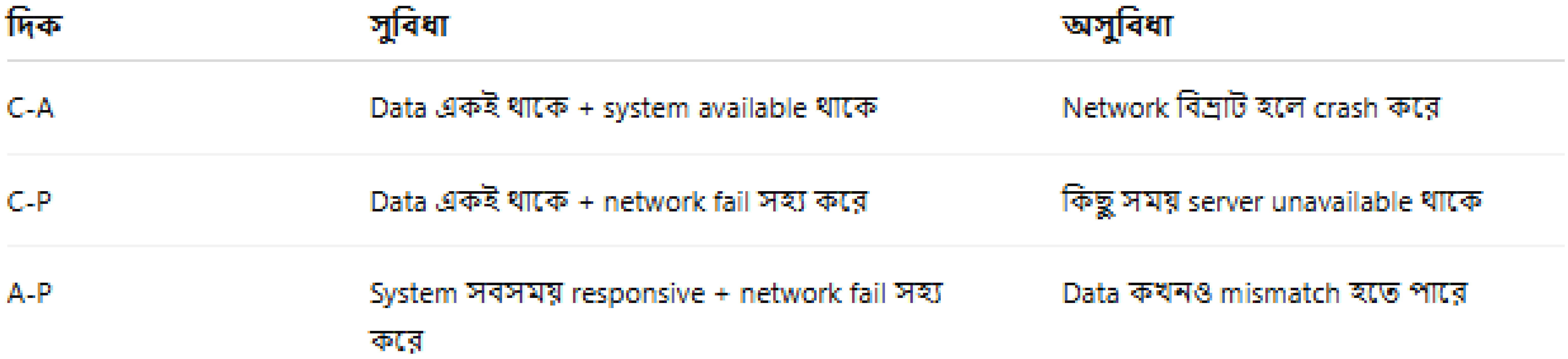
যদি CAP Theorem না জানেন তাহলে কী সমস্যায় পড়তে পারেন?

ধরেন আপনি একটা বড় system বানালেন, যেমন Facebook, Amazon, বা Daraz-এর মতো ecommerce site। তখন:

আপনি জানতেনা না CAP theorem.আপনি ভাবলেন system তো কাজ করছেই।

👉 কিন্তু বাস্তবে distributed system এ network fail, latency, data mismatch এসব হবে – তখন আপনি ভুল design করবেন downtime বাড়বে, user data mismatch হবে।

CAP THEOREM এর সুবিধা ও অসুবিধা:



দিক	সুবিধা	অসুবিধা
C-A	Data একই থাকে + system available থাকে	Network বিভ্রাট হলে crash করে
C-P	Data একই থাকে + network fail সহ্য করে	কিছু সময় server unavailable থাকে
A-P	System সবসময় responsive + network fail সহ্য করে	Data কখনও mismatch হতে পারে

C = CONSISTENCY
A = AVAILABILITY
P = PARTITION TOLERANCE

চলুন একটি বাস্তব উদাহরণ এর মাঝে বুঝি বিষয়টা :

মোবাইল ব্যাংকিং APP (নগদ / বিকাশ) কোথায় চিন্তা করি :

ধরুন আপনি রিমোট এলাকায় আছেন যেখানে নেটওয়ার্ক দুর্বল, কিন্তু আপনার বিকাশ APP ওপেন হচ্ছে ,

এখন আপনি কোনো মতে আপনার গার্লফ্রেইন্ড কে টাকা সেন্ড করলেন :
APP আপনাকে দেখালো যে : "টাকা পাঠানো হয়েছে।"

কিন্তু RECEIVER-এর ফোনে আসলেই টাকা ঢুকেছে কিনা – আপনি কিন্তু তা জানেন না।

এখানে তাহলে কী হলো?

APP “AVAILABLE” ছিল (আপনি USE করতে পেরেছেন)
NETWORK মাঝপথে FAIL করায় CENTRAL SERVER আর CLIENT যোগাযোগ করতে পারেনি – অর্থাৎ “PARTITION” হয়েছিল

কিন্তু আপনি জানেন ই না আসল TRANSACTION SUCCESS হয়েছে কিনা – “CONSISTENCY” GUARANTEE নাই।

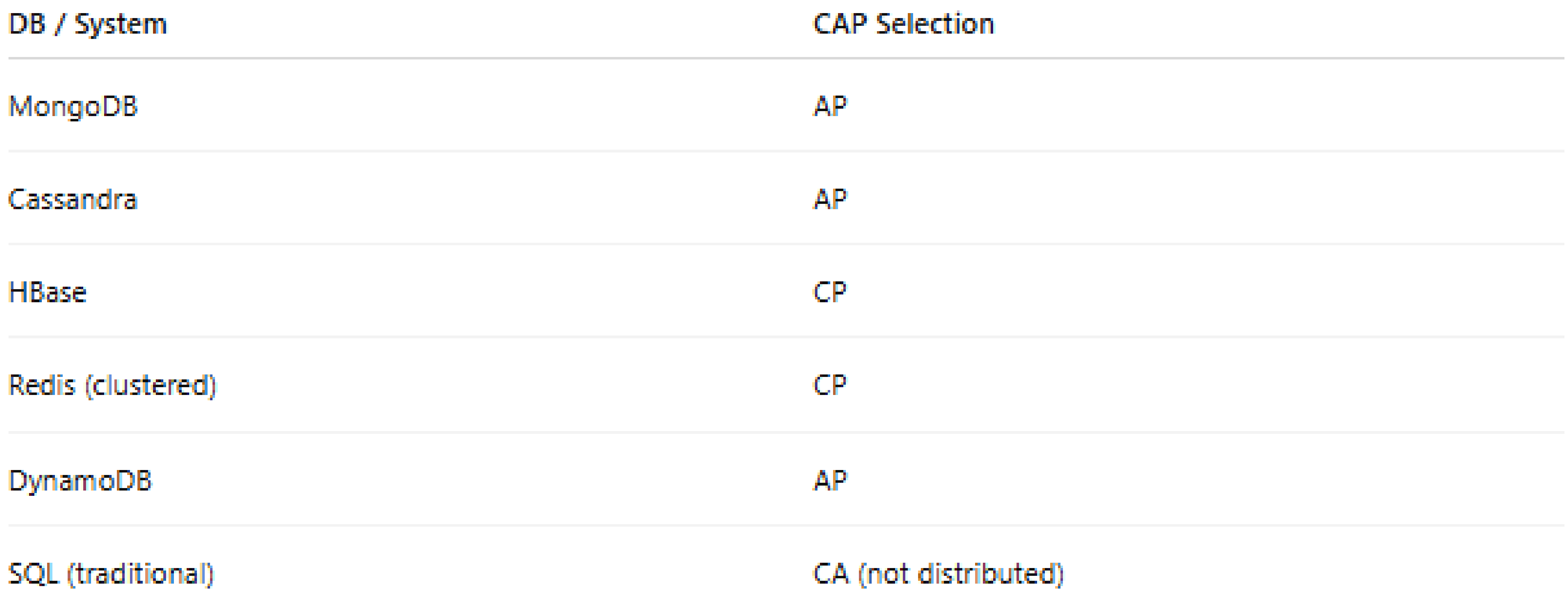
👉 অর্থাৎ, এটা A + P SYSTEM।

LEARN HUNTER

তাহলে আপনি কবে কখন কোনটা বেছে নিবেন এইটা জানা সব চেয়ে বেশি জরুরি

1. যদি Data loss একেবারেই চলবে না এমন হয় তাহলে => CP (Consistency + Partition)
2. যদি System সবসময় চালু রাখতে হবে => AP (Availability + Partition)
3. আপনার ছোট system, কিন্তু network fail চলবে না তাহলে => CA (Consistency + Availability)

CAP Theorem এর ভিত্তিতে Popular Technologies গুলো:



DB / System	CAP Selection
MongoDB	AP
Cassandra	AP
HBase	CP
Redis (clustered)	CP
DynamoDB	AP
SQL (traditional)	CA (not distributed)

MongoDB — AP :

MongoDB হল NoSQL document-based database। এটা availability আর partition tolerance-কে বেশি গুরুত্ব দেয়।

- আপনি query করলে MongoDB দ্রুত response দেয় (Available ✅)
- Network fail হলেও system চালু থাকে (Partition Tolerant ✅)
- কিন্তু সব node-এ একই সময়ে updated data নাও থাকতে পারে (Consistency ❌)

বাস্তব উদাহরণ:

ধরো তুমি একটা product এর দাম ১০০ টাকা করলা। Network partition এর কারণে এক user ৯০ টাকা দামে দেখতেছে, আরেকজন ১০০ টাকা – এটা Consistency break।

Cassandra — AP :

Cassandra হল একটি distributed, decentralized database। এটি Facebook তৈরি করেছিল।

- Data পড়া-লেখা দুইটাই দ্রুত হয় (Available ✅)
- Node গুলোর মধ্যে কোনটা fail হলেও কাজ চলে (Partition Tolerant ✅)
- কিন্তু সব জায়গায় একই সময়ে exact data পাবার গ্যারান্টি নেই (Consistency ❌)

বাস্তব উদাহরণ:

আপনি comment করলেন, কিন্তু কিছুক্ষণ পর অন্য user দেখলো না – পরে sync হয়ে গেল।

HBASE – CP

HBase হল Hadoop ecosystem এর অংশ, যা large scale data store করে।

- Network fail হলেও system মারা যায় না (Partition Tolerant ✅)
- সব node এ data always same রাখে (Consistent ✅)
- কিন্তু কোনো node unavailable হলে পুরো system response দেয় না (Availability ❌)
-

বাস্তব উদাহরণ:
তুমি একটা transaction করলে, এটা নিশ্চিত হবে যে সব node ঠিক আছে, data consistent – কিন্তু কিছু সময় delay হতে পারে।

REDIS (CLUSTERED MODE) – CP

Redis হল in-memory key-value store, কিন্তু যখন clustered mode এ use করো তখন distributed হয়।

- Network fail সহ্য করতে পারে (Partition Tolerant ✅)
- সব data consistent রাখে (Consistent ✅)
- কিন্তু কিছু node unavailable হলে read/write বন্ধ হতে পারে (Availability ❌)
-

বাস্তব উদাহরণ:
যদি কোনো shard down হয়ে যায়, Redis failover না হওয়া পর্যন্ত query আসবে না।

DYNAMODB — AP

Amazon এর NoSQL DB। High availability এবং scalability দেয়।

- Always responsive (Available ✅)
- Network fail হলেও crash হয় না (Partition Tolerant ✅)
- But consistency sacrificed (Consistency ❌)

বাস্তব উদাহরণ:
তুমি ১ম বার একটা price update করলে, সাথে সাথে সব user সেটি দেখতে পাবে না, একটু সময় লাগবে।

SQL (TRADITIONAL, E.G., MYSQL, POSTGRESQL) — CA (NOT DISTRIBUTED)

Traditional SQL DB (যেমন MySQL) সাধারণত একটিমাত্র server-এ চলে।

- সব data consistent থাকে (Consistency ✅)
- System always responds (Availability ✅)
- কিন্তু যদি DB আর client এর মধ্যে connection বিচ্ছিন্ন হয় — কোনো ব্যবস্থা নাই (Partition Tolerance ❌)

বাস্তব উদাহরণ:
তুমি যখন internet ছাড়াই try করো DB connect করতে – কিছুই হবে না। Partition সহ্য করতে পারে না।

শেষ কথা:

সুতরাং এইটাই ছিল আজকে CAP Theorem নিয়ে বিশদ আলোচনা যেটা আমাদের সফটওয়্যার ইঞ্জেনিয়ারিং এর ক্ষেত্রে প্রচুর পরিমানে সাহায্য করে আশা করি আজকের পরে এই বিষয় নিয়ে আর দ্বিধা দ্বন্ধে ভুগবেন না। পরবর্তীতে আমরা Cache নিয়ে বিস্তারিত আলোচনা করবো