📘 CAP Theorem Notes

1. Introduction to CAP Theorem

- Proposed by Eric Brewer (2000).
- In the presence of a network partition, a distributed system can guarantee at most two of the following three:
  1. **Consistency (C)**
  2. **Availability (A)**
  3. **Partition Tolerance (P)**
- Real distributed systems must choose trade-offs depending on requirements.

---

2. Properties Explained

2.1 Consistency (C)

- Every read sees the most recent write, like a single up-to-date database.
- Equivalent to: "all users get the same answer at the same time."
- Example: Banking system must always show the correct balance.

2.2 Availability (A)

- The system always responds to requests (success or failure) even if some nodes are down.
- It doesn’t guarantee the freshest data, only that you get some response.
- Example: Shopping website shows cached prices when DB is lagging.
- Doubt: “If the system is down, how can it respond?”
  - Availability means the system as a whole keeps serving because at least some replicas are up (not the crashed one).

2.3 Partition Tolerance (P)

- System continues to operate despite network failures (messages between nodes may be lost).

- You can’t avoid partitions in large networks → must tolerate them.
- Doubt: “Partition tolerance sounds like availability — both say system keeps working if one node is down?”
  - Difference:
    - Availability → node/server crashes but others still serve.
    - Partition tolerance → network split between groups of nodes, both groups may still serve requests independently.

---

3. Trade-Offs in CAP

3.1 CA (Consistency + Availability)

- Works only when no partitions exist.
- If a partition happens, one property must be dropped.
- Rare in real-world distributed systems.

3.2 CP (Consistency + Partition Tolerance)

- Guarantees:
  - Always consistent
  - Survives partitions
- Sacrifices: Availability (some requests rejected).
- Examples: ZooKeeper, HDFS NameNode
- Behavior:
  - If a node (like C) is partitioned away, it refuses writes (❌ try later).
  - A+B continue serving and stay consistent.
  - When partition heals, C syncs from A+B.
- Doubt: “If someone writes to A or B while C is cut off, won’t that break consistency?”
  - No, because C rejects writes. Only A+B accept. After healing, C catches up → consistency preserved.

3.3 AP (Availability + Partition Tolerance)

- Guarantees:
  - Always available
  - Survives partitions
- Sacrifices: Consistency (may diverge → eventual consistency).
- Examples: DynamoDB, Cassandra
- Behavior:
  - If C is partitioned away, C′ (replica) may take over.
  - Both sides accept writes (A+B vs C′).
  - After healing, conflicts resolved via:
    - Last-write-wins (timestamps)
    - Vector clocks (causality tracking)
    - CRDTs (auto-merge data structures)
- Doubt: “Why can’t A+B sync with C′ during partition?”
  - Because partition means no communication path exists. Only after healing can they exchange logs and resolve.
- Doubt: “After healing, how do A+B decide whether to sync with C or C′?”
  - Cluster consensus protocol (Raft/Paxos, Gossip, etc.) decides:
    - If A+B had majority → C/C′ syncs from A+B.
    - If C′ was promoted → A+B sync with C′.
    - Conflict resolution may merge histories.

---

4. Timeline Example

- Partition Event:
  - T1 → A+B and C′ lose contact.
  - T2 → A+B continue writes. C′ may also accept writes.
- Healing Event:
  - T3 → Network restored.
  - T4 → Sync happens:

- CP system → C′ discards its writes, catches up from A+B.
- AP system → Conflicts reconciled (timestamps/vector clocks).

---

5. Real-World Examples

- CP Systems: HDFS, Zookeeper, Google Spanner (strong consistency).
- AP Systems: DynamoDB, Cassandra, CouchDB (eventual consistency).
- CA Systems (rare, mostly theoretical at large scale): Relational DBs in a single-node cluster.

✅ Final Takeaway

- You cannot avoid partitions in real networks.
- So every real distributed system is either CP or AP.
- Choice depends on business needs:
  - Banking → CP (better to reject a transaction than show wrong balance).
  - E-commerce → AP (better to show slightly stale inventory than reject orders).

Quick Comparison Table

Property	CA	CP	AP
Consistency	✅ (only without partitions)	✅	❌ (eventual)
Availability	✅ (only without partitions)	❌ (may reject)	✅
Partition Tolerance	❌	✅	✅
Typical Use Cases	Single-node RDBMS	Metadata stores, coordination (ZooKeeper, Spanner)	High-throughput, user-facing (Cassandra, DynamoDB)

CP vs AP Decision Checklist

☐ Strong invariants required at all times? Choose **CP**

- [ ] Okay to return slightly stale data if system stays up? Choose **AP**
- [ ] Can clients retry on write failures? **CP** is viable
- [ ] Need write availability during partitions? Prefer **AP**
- [ ] Human-facing latency critical over perfect freshness? **AP**
- [ ] Cross-region, globally distributed with strict ordering needs? Likely **CP** with consensus
- [ ] Conflict-free data types or easy merge semantics available? **AP** gets easier
- [ ] Regulatory or financial correctness constraints? Bias to **CP**

Flashcards: CAP Theorem

▼ What does CAP stand for?

- Consistency, Availability, Partition Tolerance

▼ Who proposed the CAP theorem and when?

- Eric Brewer, 2000

▼ What is the core statement of CAP?

- In the presence of a network partition, a system can provide at most two of Consistency, Availability, and Partition Tolerance.

▼ Define Consistency in CAP.

- Every read sees the most recent write, as if from a single up-to-date copy.

▼ Define Availability in CAP.

- Every request receives a non-error response, without guarantee that it contains the most recent write.

▼ Define Partition Tolerance in CAP.

- The system continues operating despite network faults that prevent some nodes from communicating.

▼ How do Availability and Partition Tolerance differ?

- Availability concerns serving requests despite node failures. Partition tolerance concerns continuing operation despite network splits

between groups of nodes.

▼ In a partition, what must a real distributed system choose between?

- Consistency and Availability

▼ What does a CP system sacrifice during partitions?

- Availability, by rejecting or delaying some requests.

▼ What does an AP system sacrifice during partitions?

- Immediate consistency, allowing temporary divergence.

▼ Give two examples of CP systems.

- ZooKeeper, Google Spanner

▼ Give two examples of AP systems.

- Cassandra, DynamoDB

▼ What is eventual consistency?

- Replicas may diverge temporarily but converge to the same state after no new updates and successful communication.

▼ Name three conflict resolution strategies in AP systems.

- Last-write-wins, vector clocks, CRDTs

▼ In CP, what happens to writes on a minority partition?

- They are rejected; clients are asked to retry later.

▼ In AP, why can’t partitions synchronize during the split?

- There is no communication path; sync occurs only after healing.

▼ How does a cluster decide which side to follow after healing?

- Consensus or membership protocols determine leadership or majority; others reconcile to it.

▼ When might you choose CP over AP?

- When strong invariants and correctness are critical, such as financial systems.

▼ When might you choose AP over CP?

- When high availability and low latency are more important than immediate consistency, such as user-facing catalogs.