In-Memory Computing SUMMIT | NORTH AMERICA 2018

Replication Distilled: Hazelcast Deep Dive

Ensar Basri Kahveci
Hazelcast

Hazelcast

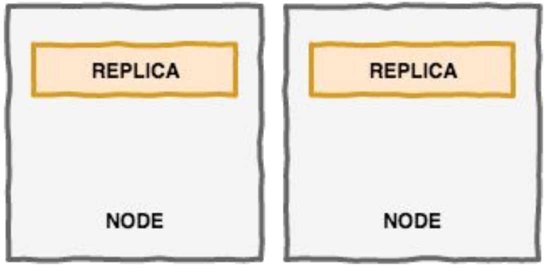
- The leading open source Java IMDG
- Distributed Java collections, concurrency primitives, ...
- Distributed computations, messaging, ...

In-Memory Data Grids

- Distributed caching
- Keeping data in local JVM for fast access & processing
- Elasticity, availability, high throughput, and low latency
- Multiple copies of data to tolerate failures

Replication

- Putting a data set into multiple nodes
- Fault tolerance
- Latency
- Throughput

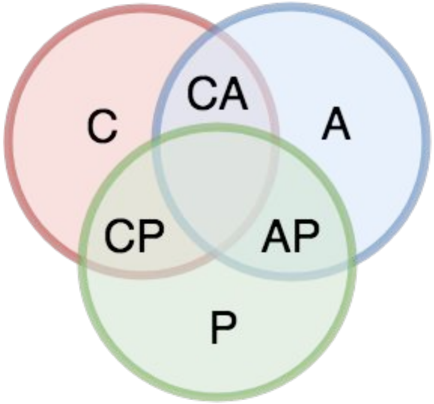


Challenges

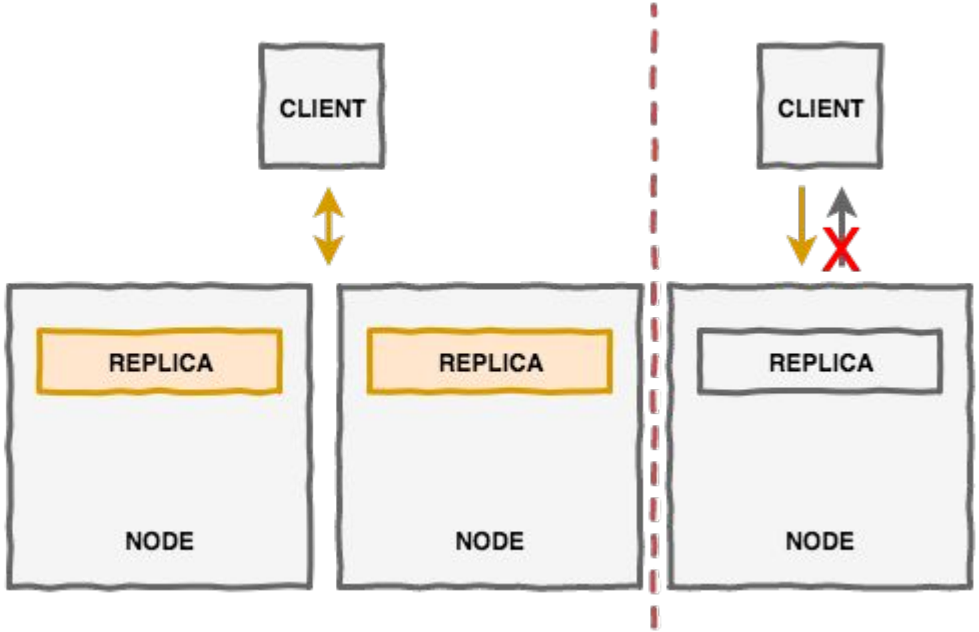
- Where to perform reads & writes?
- How to keep replicas sync?
- How to handle concurrent reads & writes?
- How to handle failures?

CAP Principle

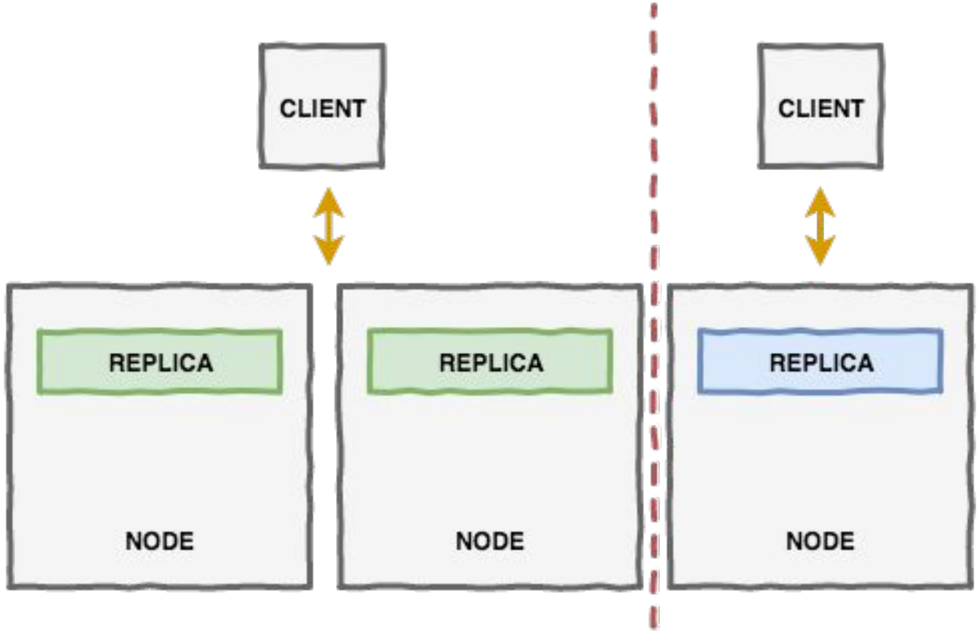
- Pick two of **C**, **A**, and **P**
- **CP** versus **AP**



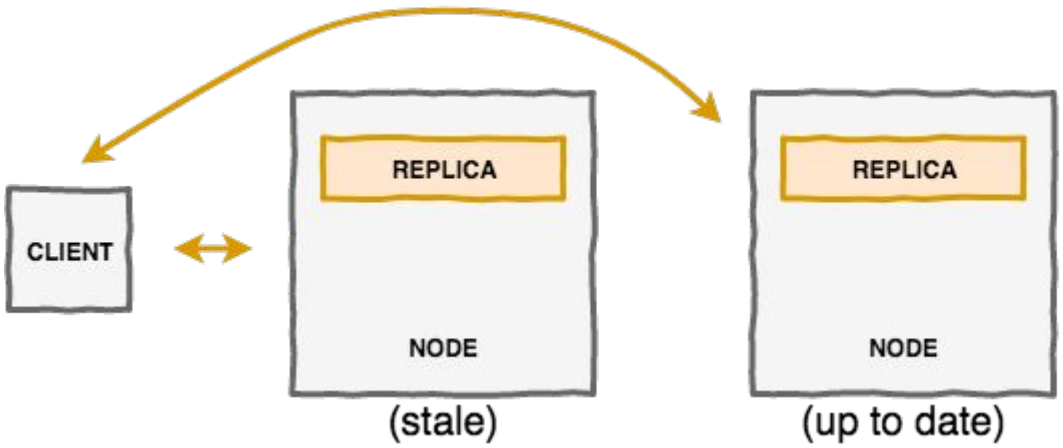
CP



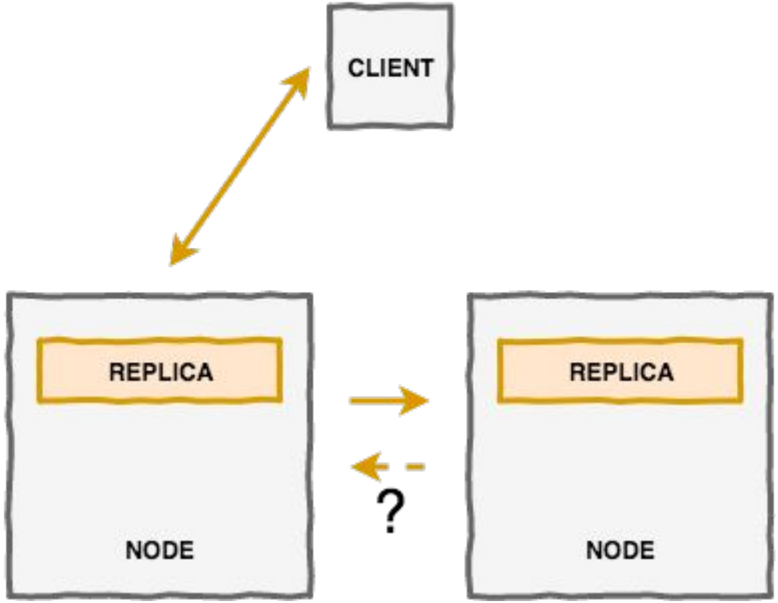
AP



Consistency/Latency Trade-off



Consistency/Latency Trade-off



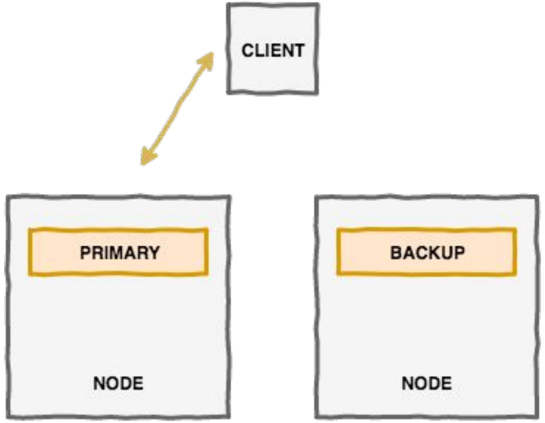
PACELC Principle

- If there is a network partition **(P)**, **we have to choose** between availability and consistency **(AC)**.
- Else **(E)**, during normal operation, **we can choose** between latency and consistency **(LC)**.

Let’s build
the core replication protocol
of Hazelcast

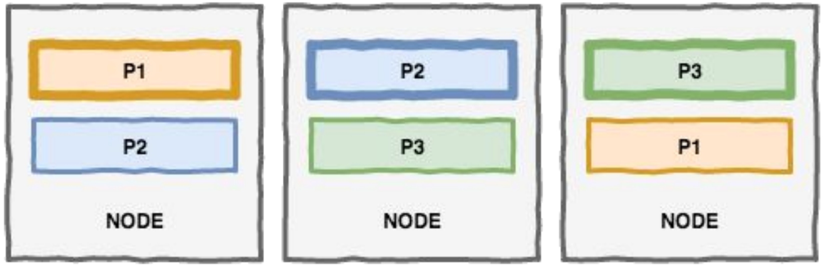
Primary Copy

- Operations are sent to primary replicas.
- Strong consistency when the primary is reachable.

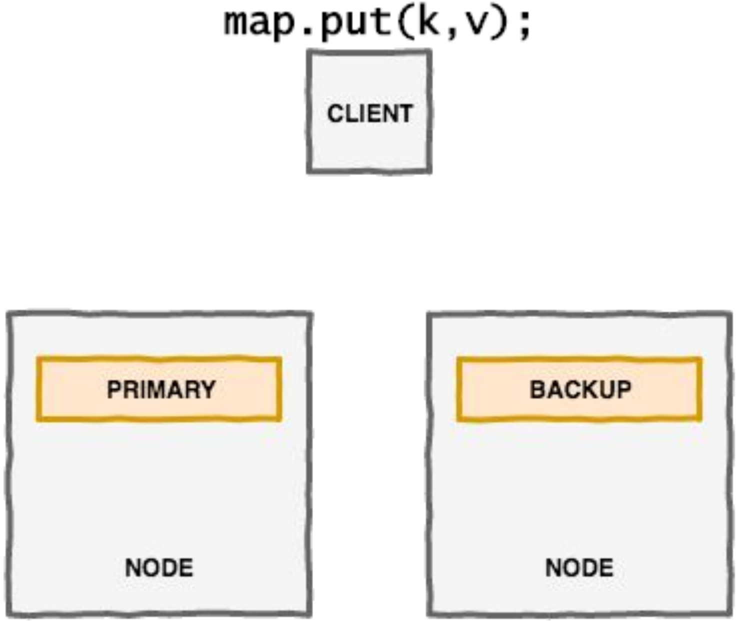


Partitioning (Sharding)

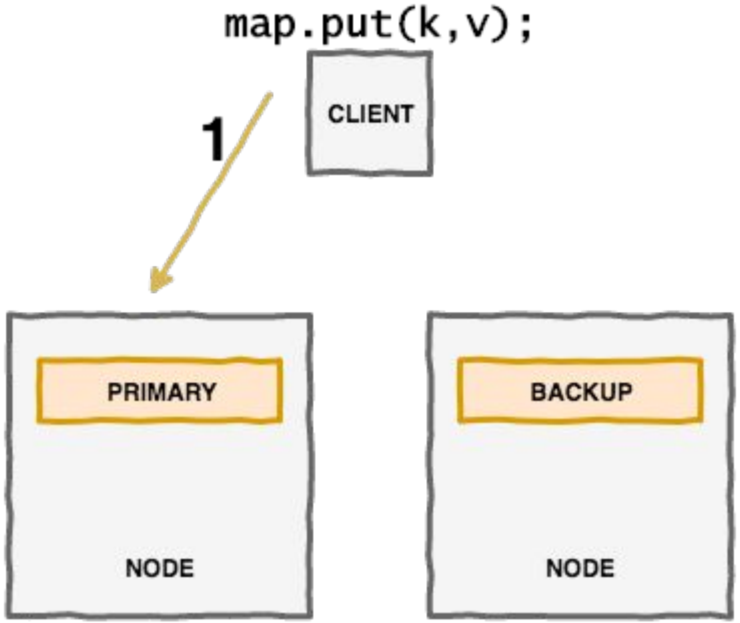
- Partitioning helps to scale primaries.
- A primary replica is elected for each partition.



Updating Replicas

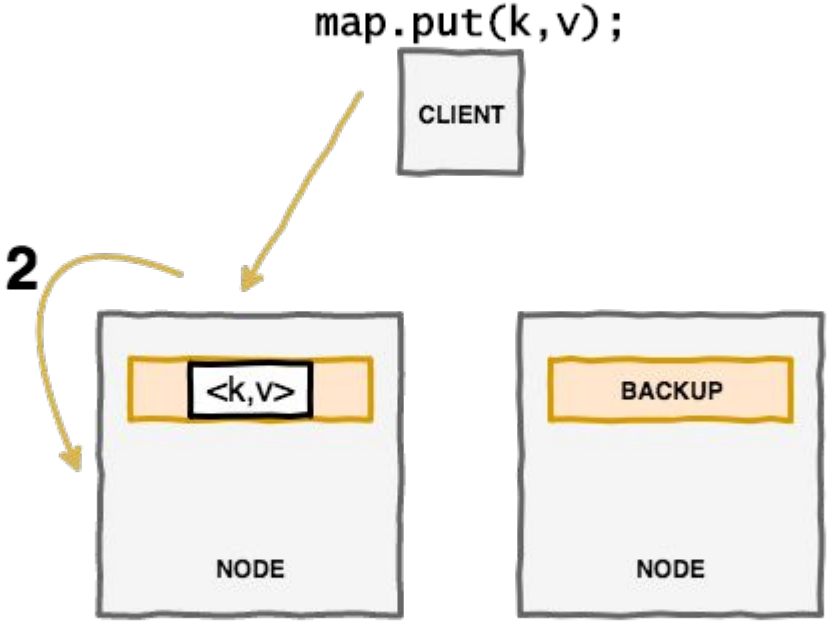


Updating Replicas

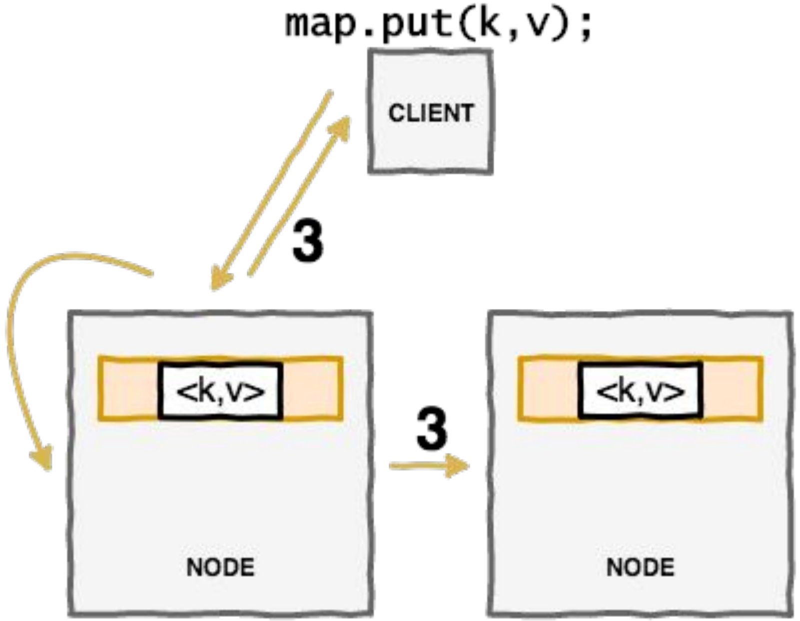


partition id = hash(serialize(key)) % partition count

Updating Replicas



Updating Replicas

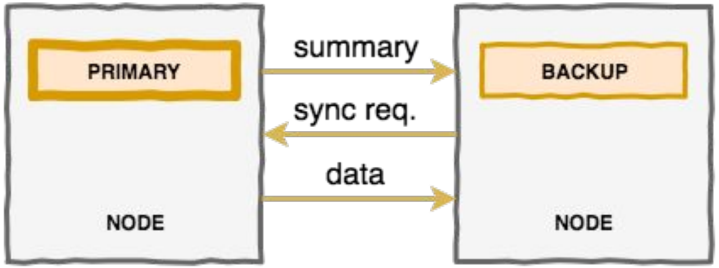


Async Replication

- Each replica is updated separately.
- High throughput and availability

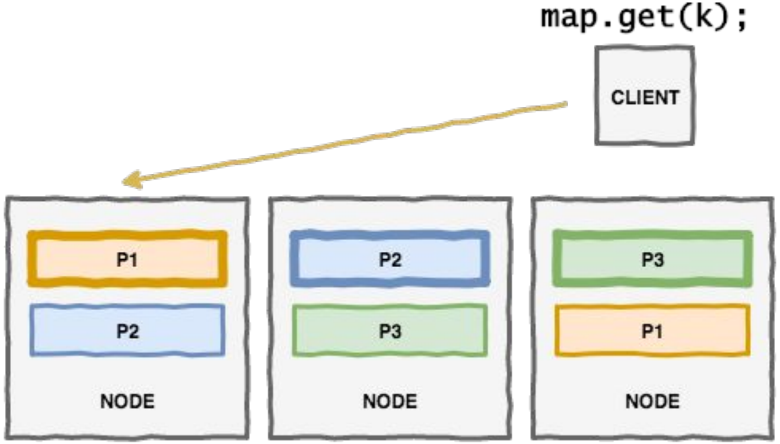
Anti-Entropy

- Backup replicas can fall behind the primary.
- Non-sync backups are fixed with an active anti-entropy mechanism.



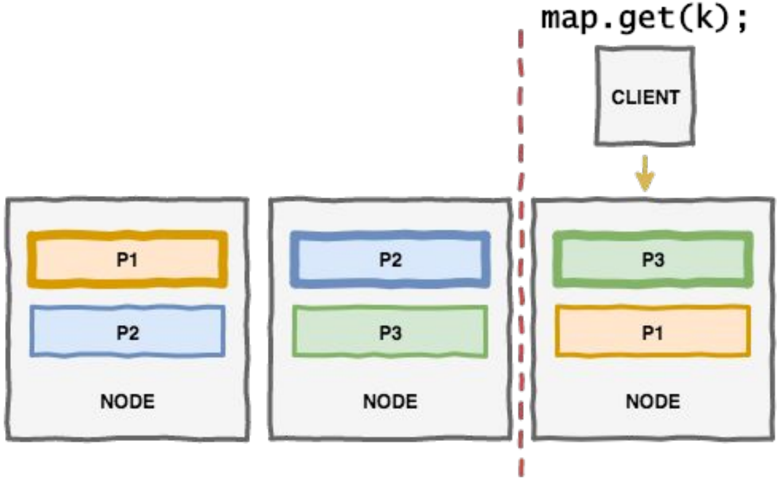
Replicas are not sync

- The client reads a key from the current primary replica.



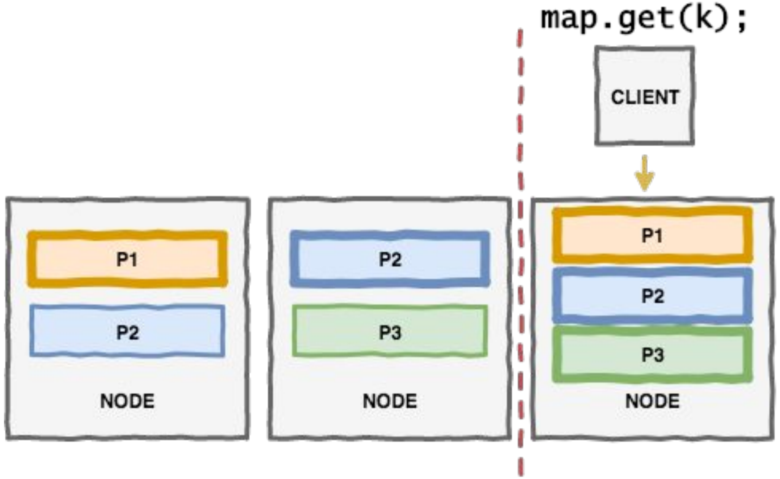
Network Partitioning

- The client reads the same key.



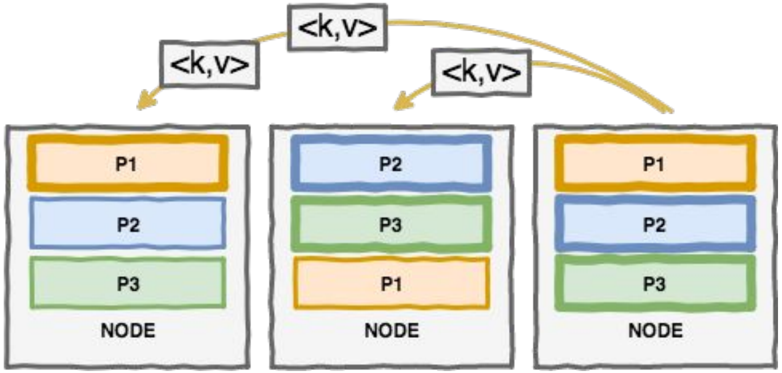
Split-Brain

- Strong consistency is lost.



Resolving the Divergence

- Merge policies: higher hits, latest update / access, ...
- Merging may cause lost updates.



Let’s classify this protocol with PACELC

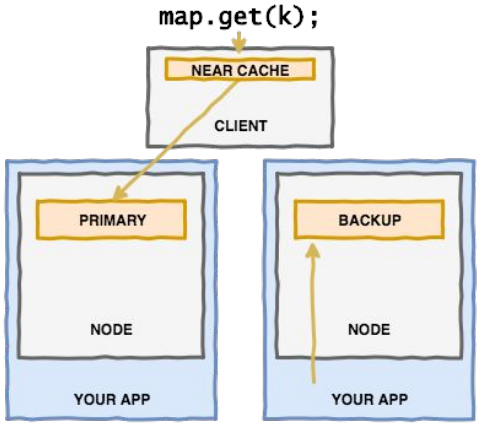
Hazelcast is PA/EC

- Consistency is usually traded to availability and latency together.
- Hazelcast works in memory and mostly used in a single computing cluster.
- Consistency - latency trade-off is minimal.
- PA/EC works fine for distributed caching.

Favoring Latency (PA/EL)

Scaling Reads

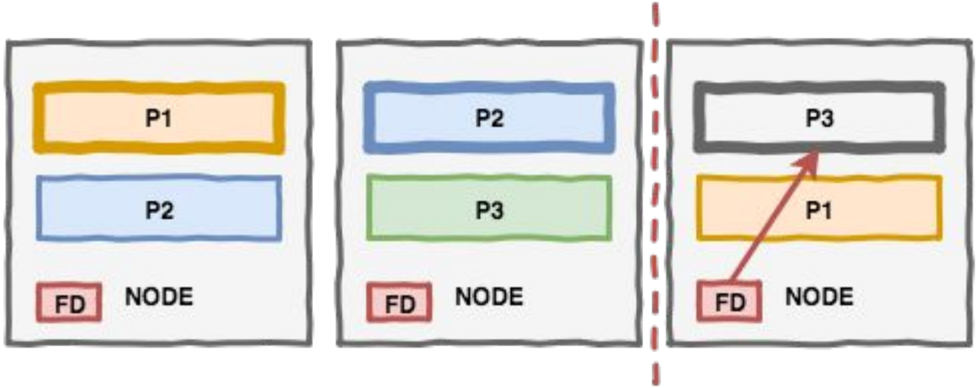
- Reads can be served locally from near caches and backup replicas.



Favoring Consistency (PC/EC)

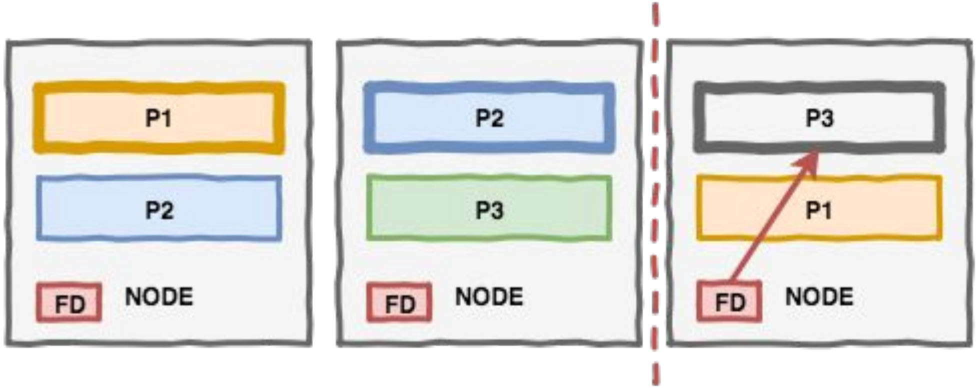
Failure Detectors

- Local failure detectors rely on timeouts.
- Operations are blocked after the cluster size falls below a threshold.



Failure Detectors

- It takes some time to detect an unresponsive node.
- Minimizes divergence and maintains the baseline consistency.



Isolated Failure Detectors

- Configure failure detectors independently for data structures
- Phi-Accrual Failure Detector

CP Data Structures

- IDGenerator
- Distributed impls of **`java.util.concurrent.*`**
- PA/EC is not the perfect fit for CP data structures.

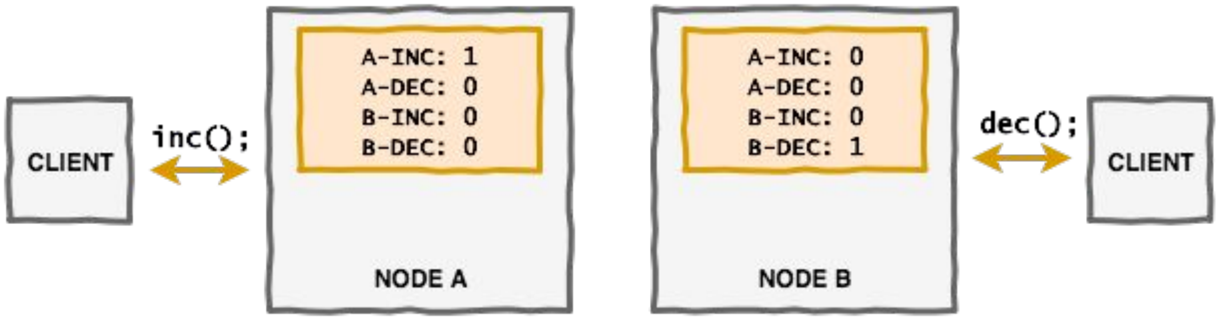
Flake IDs

- Local unique id generation
- Nodes get a unique node id during join.
- K-ordered IDs

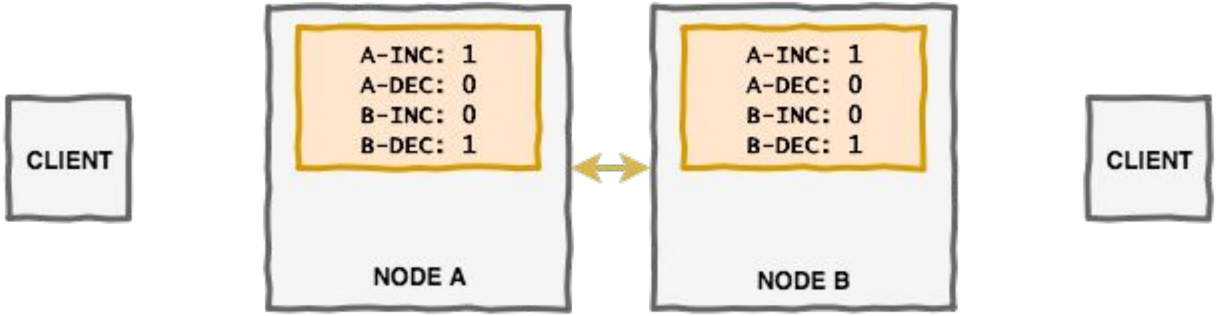
CRDTs

- CRDTs: Conflict-free Replicated Data Types
- Replicas are updated concurrently without coordination.
- Strong eventual consistency
- Counters, sets, maps, graphs, …

PN-Counter


CLIENT
inc();
A-INC: 1
A-DEC: 0
B-INC: 0
B-DEC: 0
NODE A
A-INC: 0
A-DEC: 0
B-INC: 0
B-DEC: 1
NODE B
dec();
CLIENT

PN-Counter


CLIENT
A-INC: 1
A-DEC: 0
B-INC: 0
B-DEC: 1
NODE A
A-INC: 1
A-DEC: 0
B-INC: 0
B-DEC: 1
NODE B
CLIENT

Sync Replication

- Concurrency primitives imply the true CP behavior.
- Paxos, Raft, ZAB, VR
- Re-implementing Hazelcast concurrency primitives with Raft

Recap

- http://bit.ly/hazelcast-replication-consistency
- http://bit.ly/hazelcast-network-partitions
- http://dbmsmusings.blogspot.com/2017/10/hazelcast-and-mythical-paec-system.html

Thanks!

You can find me at

- @metanet
- ebkahveci@gmail.com