Apache Cassandra

Topics

- Introduction
- Data Modeling
- Querying
- Administration
- DevOps

Introduction

- Overview
- Architecture
- Compare with Other Database

Apache Cassandra

- Distributed DBMS
  - Open-Source
  - Distributed
  - Decentralized
- Developed at Facebook
- Power the Facebook inbox search feature

Applications of Cassandra

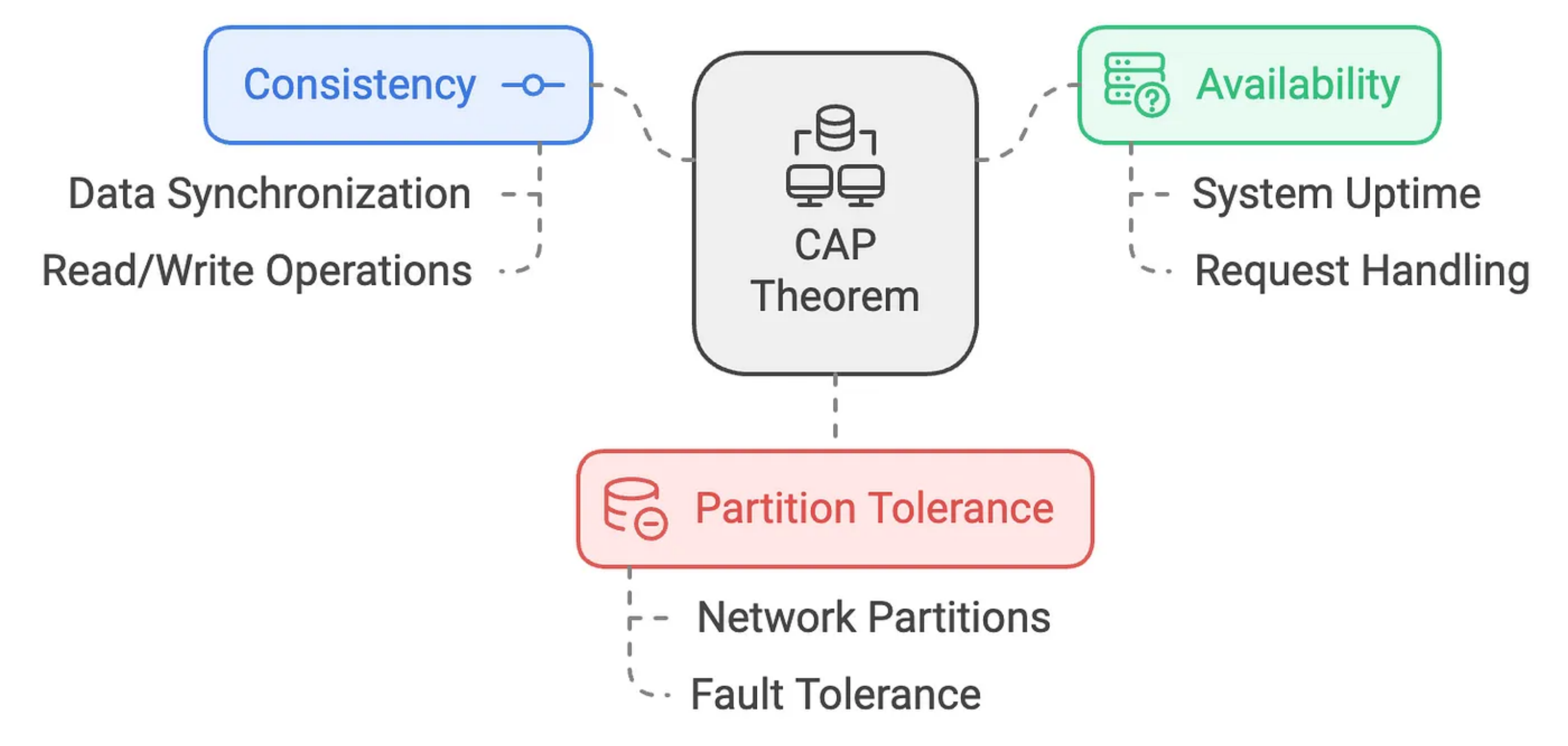
- Great Application where **Data** is collected at **High Speed** from **Different** kinds of **sources**
- Internet of Things
- Product & Retail apps
- Messaging
- Social Media Analytics
- Recommendation Engine

Traditional RDBMS vs Cassandra

Feature	Traditional RDBMS	Cassandra
Data Model	Relational	NoSQL, flexible column families
Architecture	Centralized/Master-Slave	Distributed, Peer-to-peer
Scalability	Vertical, complex horizontal	Horizontal, easy to scale out
Consistency	Strong (ACID)	Tunable, often eventually consistent
Transactions	Complex, nested	Simple
Use Cases	Structured data, strong consistency	High availability, large datasets

The CAP Theorem

- **Consistency**
    - all nodes see the same data at the same time
- **Availability**
    - every request receives a response, even if some nodes are down
- **Partition Tolerance**
    - the system continues to function despite communication breaks between nodes



CAP Theorem

Consistency

Data Synchronization

Read/Write Operations

Availability

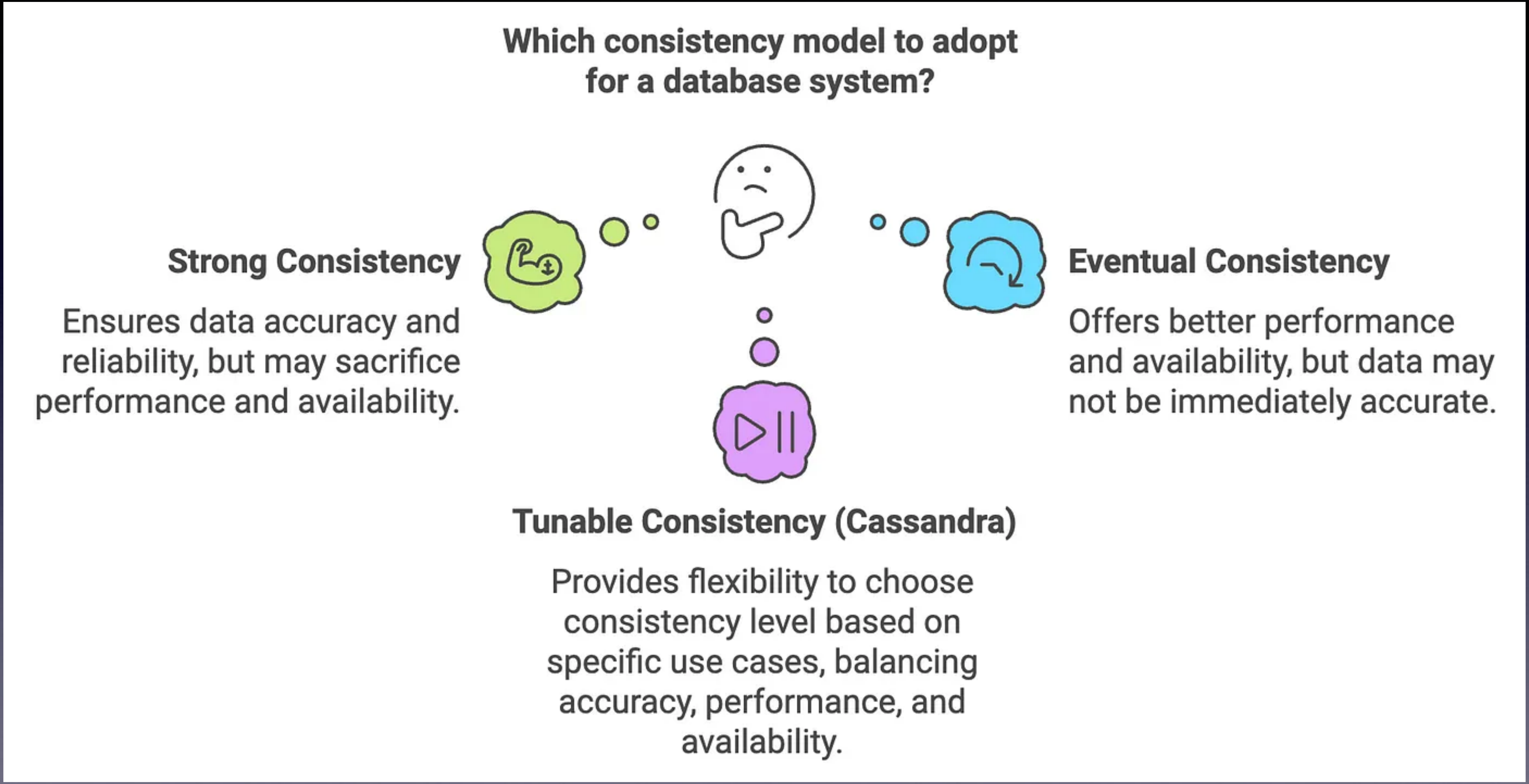
System Uptime

Request Handling

Partition Tolerance

Network Partitions

Fault Tolerance



Which consistency model to adopt for a database system?

Strong Consistency

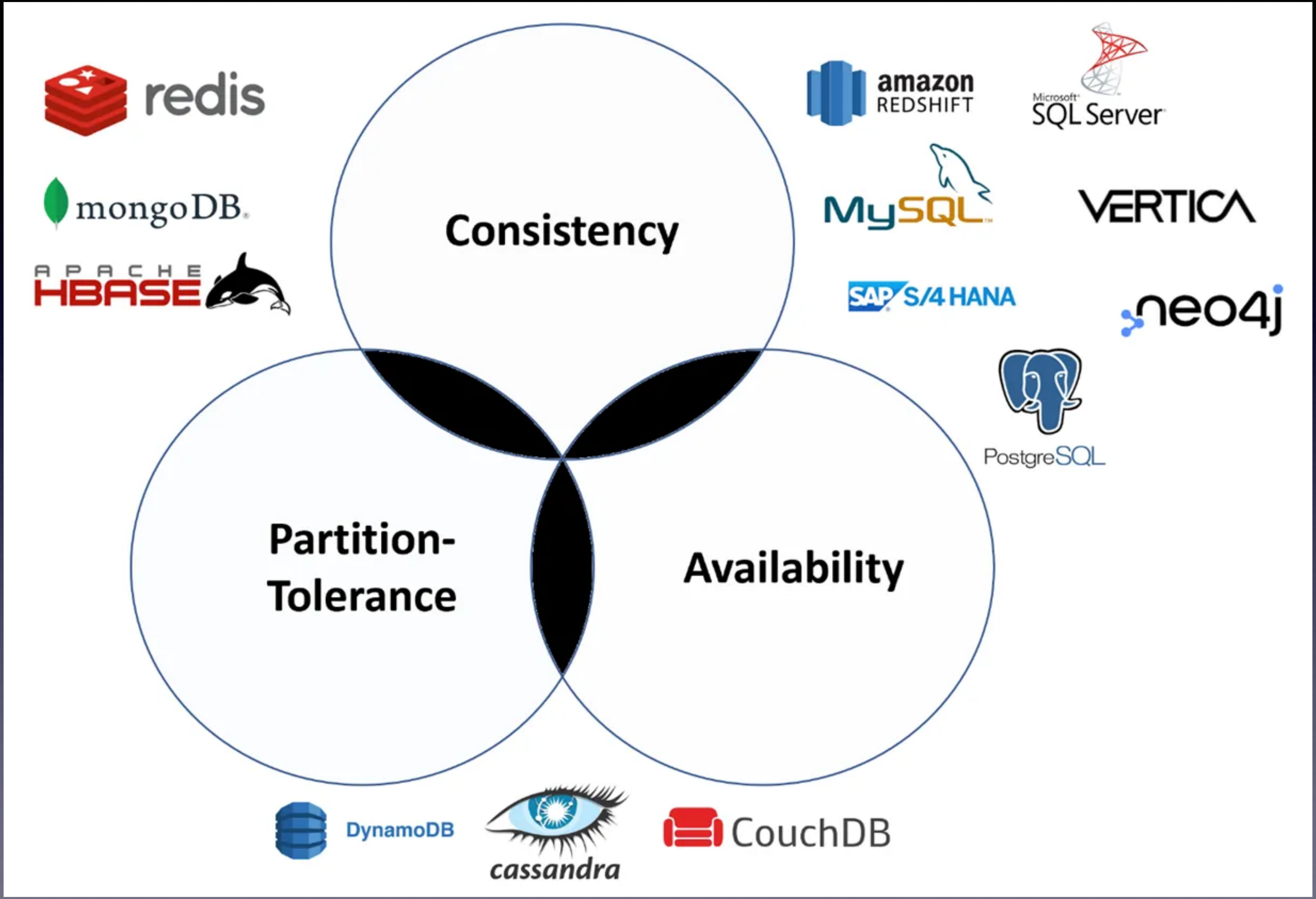
Ensures data accuracy and reliability, but may sacrifice performance and availability.

Eventual Consistency

Offers better performance and availability, but data may not be immediately accurate.

Tunable Consistency (Cassandra)

Provides flexibility to choose consistency level based on specific use cases, balancing accuracy, performance, and availability.



Consistency

Partition-Tolerance

Availability

CAP

- CA: The system prioritizes **consistency** and **availability** but doesn't handle network partitions well. In the case of a network partition, it might choose to become unavailable to maintain consistency.

- CP: The system prioritizes **consistency** and **partition tolerance** but sacrifices availability. In the face of a network partition, it will maintain consistency by refusing some requests.

- AP: The system prioritizes **availability** and **partition tolerance** but sacrifices strong consistency. It continues to operate and respond to requests even if it means returning stale data or data that is not consistent across all nodes.

BASE (NoSQL Philosophy)

- **[BASICALLY AVAILABLE]**
- System mostly works
- **[SOFT STATE]**
- May change over time
- **[EVENTUAL CONSISTENCY]**
- Gets fixed later

Key Terms in Cassandra

- Nodes
- Data Center / Cluster
- Commit Log
- SSTable
- MemTable
- Replication

Cassandra Node

- A node is a basic unit of Cassandra,
- It is a system that is part of a cluster.
- Node is the main area where the data is stored.
- The units of a node is represented as computer/server

Cassandra Data Center / Cluster

- A data center is a collection of Cassandra nodes.
- The data in a data center is stored in the form of a cluster
- The cluster is also referred to as a collection of nodes.

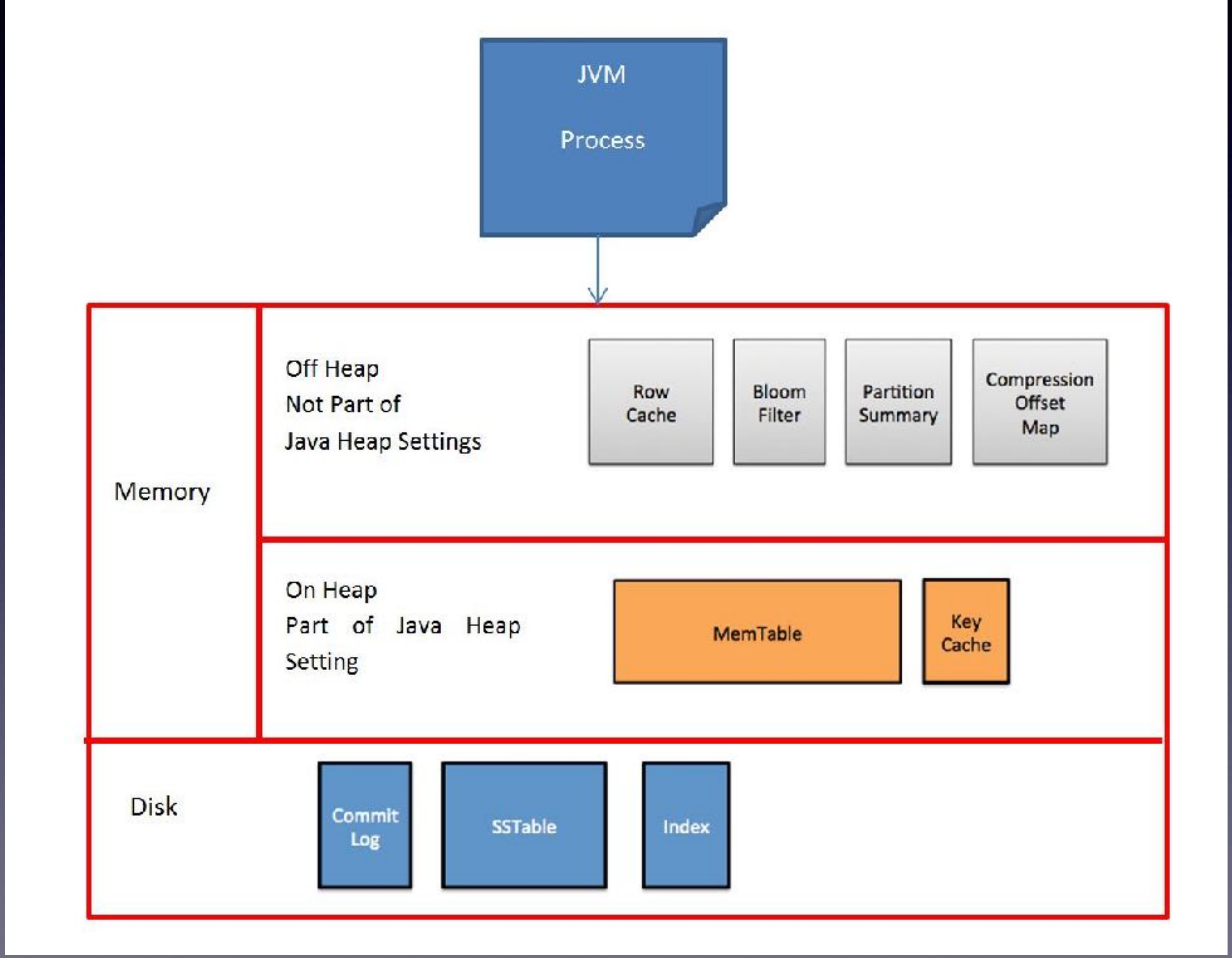
Cassandra MemTable

- MemTable is a location where data is written and stored temporarily.
- Data is written in memtable after the data is completed in the commit log.
- Memtable is a storage engine in Cassandra.
- Data in MemTable is classified into a key, and where the data is retrieved using the key as each column category has its own MemTable.
- When the write memory is full, it deletes the messages automatically.

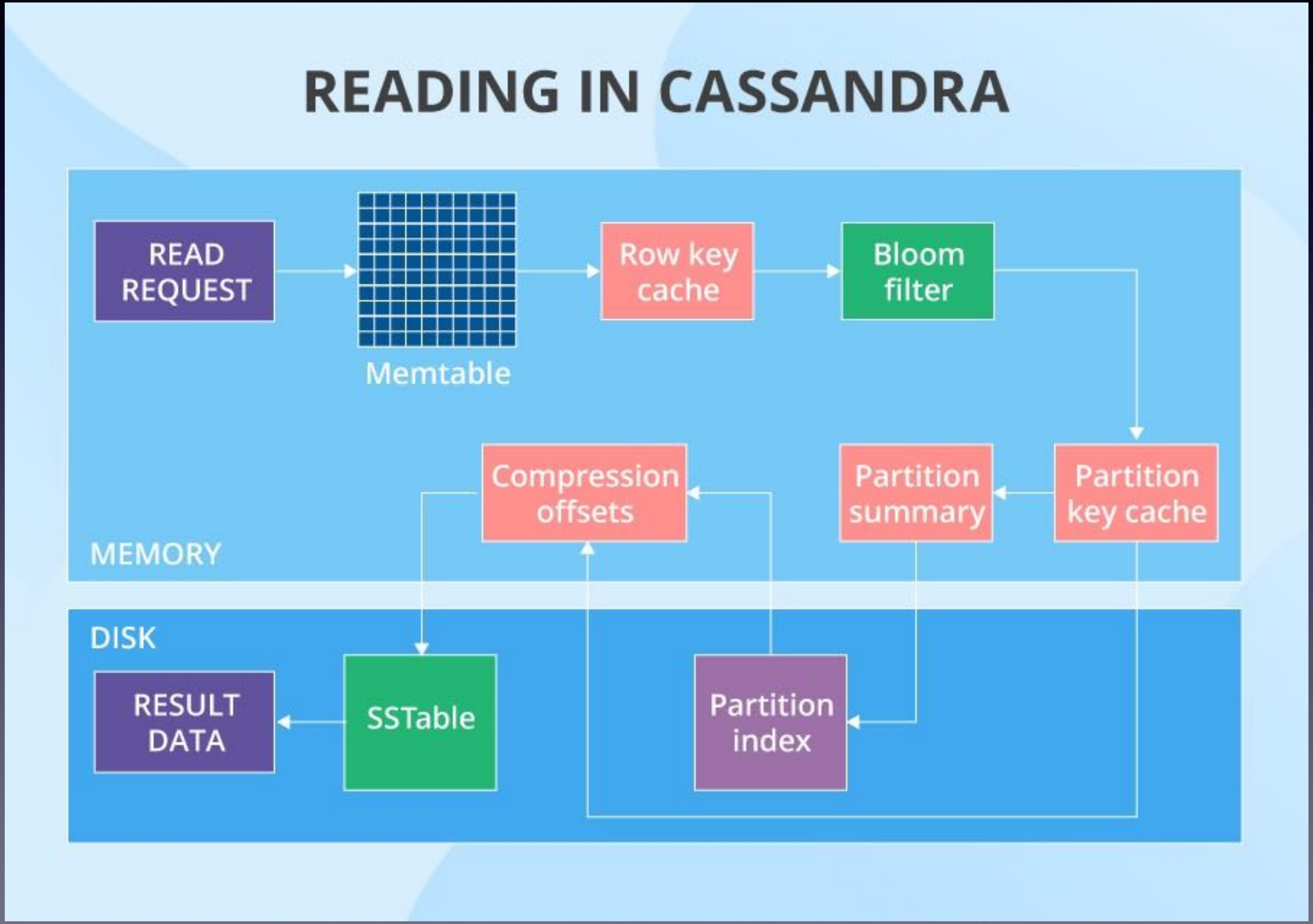
Cassandra SSTable

- SSTable also means 'Sorted String Table'.
- SSTable is a data file in Cassandra
- Its main function is to save data that is flushed from memtable.
- Unlike MemTable, SSTbale doesn't delete any data or lets any further addition once data is written.

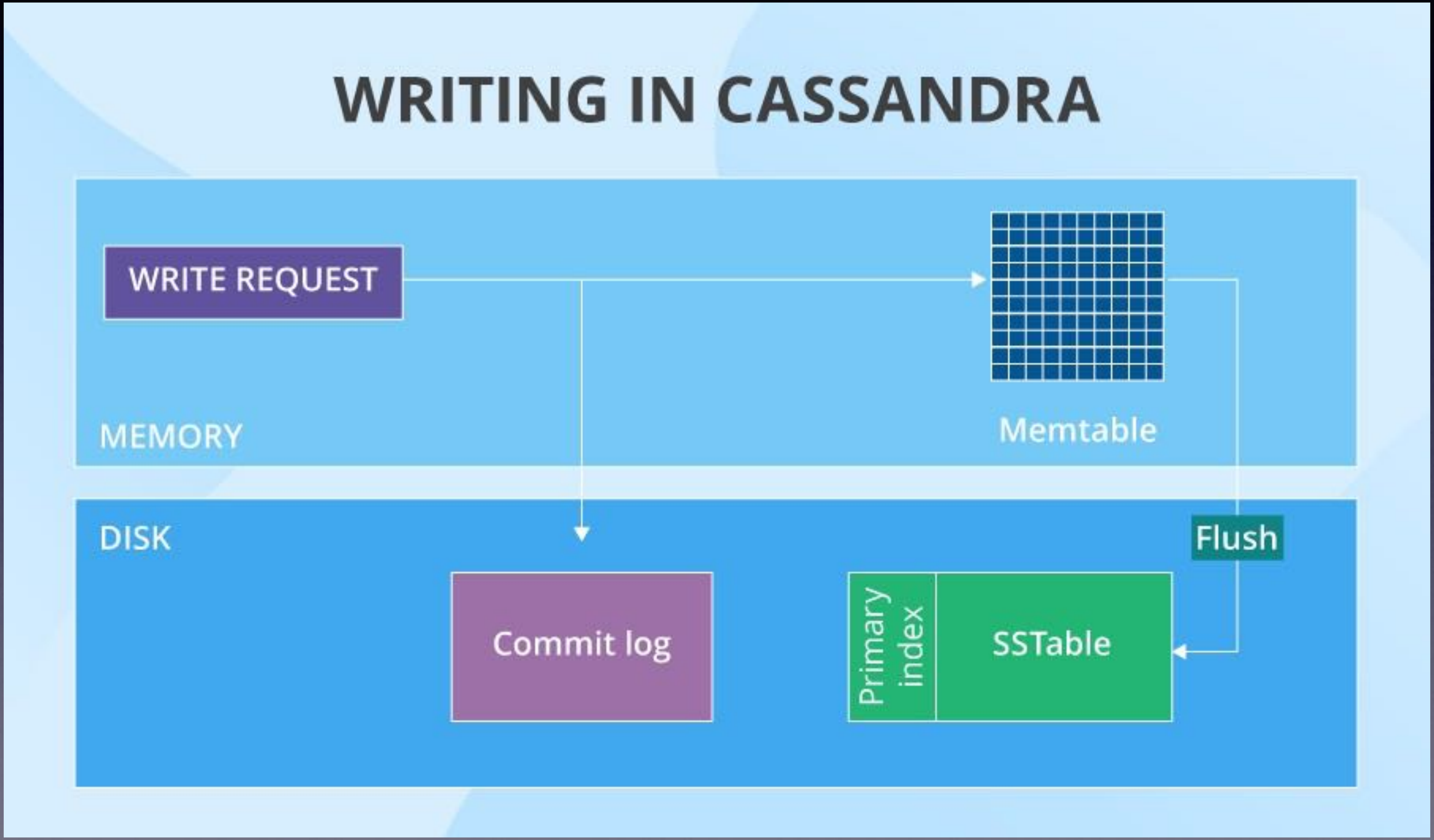
Architecture



Read Operation



Write Operation



Data Modeling

- Data Model
- Denormalization Strategies

Querying

- Cassandra Query language - CQL
- Query Optimization
- Advanced CQL Features

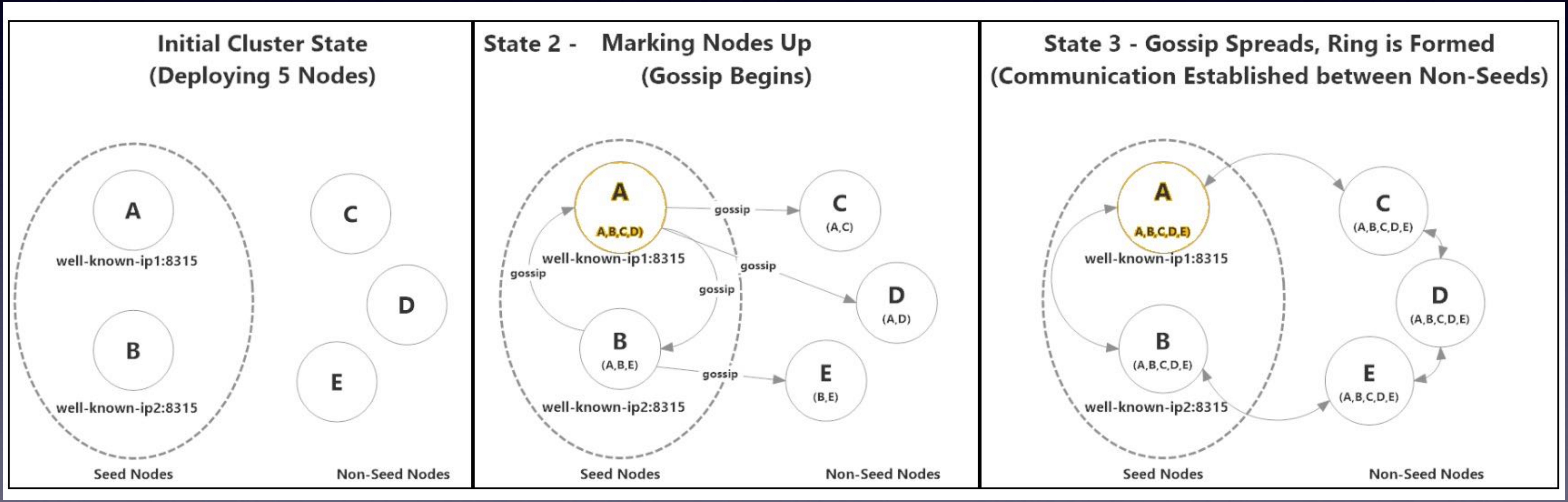
Administration

- Cluster Setup and Configuration
- Monitoring and Troubleshooting
- Backup and Recovery

Development

- Client Drivers
- Data Consistency and Durability
- Performance Optimization

High Availability



Direct vs Digest Requests

- Direct request involves the coordinator node directly contacting one replica for the data.
- Digest requests involve contacting multiple replicas and checking for data consistency by comparing digests (hashes) of the data

Direct vs Digest Requests

Feature	Direct Request	Digest Request
Primary Goal	Quickly retrieve data from one replica	Ensure data consistency across multiple replicas
Consistency	Limited consistency guarantees	Higher consistency guarantees
Performance	Potentially faster	Potentially slower due to additional network traffic and data comparison
Data Delivery	Full data retrieval	Digest (hash) of the data, not the full data
Consistency Verification	No direct verification	Uses digests to check for data consistency

Repair Request

- A repair request initiates the process of resolving data inconsistencies between replicas.
    - These inconsistencies can arise when nodes fail
    - When writes are not synchronized across all replicas.
- Repairs ensure data accuracy and consistency within the cluster, which is crucial for maintaining data integrity.

Read Repair vs Repair Request

- **Read Repair** is a process that occurs during a read operation to ensure data consistency across replicas if inconsistencies are detected.

- **Repair Request** is a broader term that refers to any request to reconcile data between replicas, whether it's during a read or as a separate maintenance task.

Repair Command

- ***nodetool repair***
  - Initiates an incremental repair.
- ***nodetool repair --full***
  - Initiates a full repair.
- ***nodetool repair [keyspace_name]***
  - Repairs a specific keyspace.
- ***nodetool repair [keyspace_name] [table1] [table2]:***
  - Repairs specific tables within a keyspace.

DevOps

- Integrating with DevOps Tools