NoSQL Column Store Databases

Technologies for Big Data Analytics - Fall Semester 23/24

Eleni Mandana & Dimitrios Markou // {emandan, dmarkou}@csd.auth.gr

Agenda

Databases

Relational databases are defined by *ACID* properties. *ACID*, stands for Atomicity, Consistency, Isolation and Durability. The reverse of *ACID* is *BASE*, which stands for Basically, Available, Soft state, and Eventual consistency.

In the era of *Big Data* the aforementioned properties conflict with the need for massive read/write requests and availability of data.

To tackle these needs, *NoSQL is introduced. It* a class of non-relational **distributed** databases, that was introduced providing better scaling capabilities and vary in between the road from *ACID* to *BASE*.

NoSQL databases are key-value, column, document or graph-based, but it is also possible to have a combination of these.

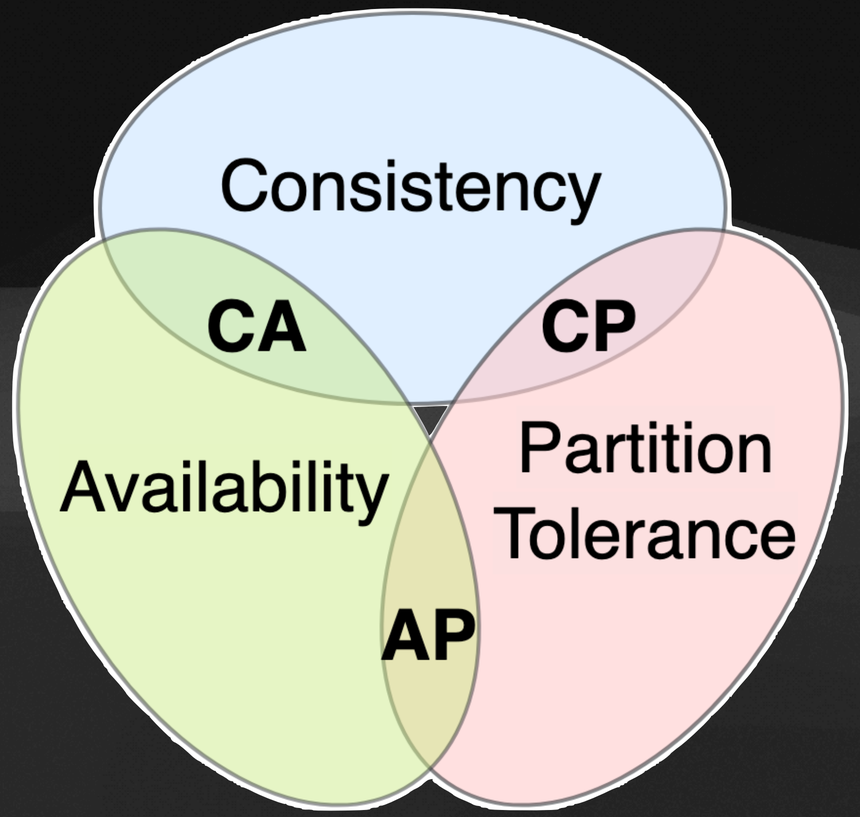
CAP Theorem

Any distributed data store can provide only two of the following three guarantees:

Consistency — Every read receives the most recent write or an error.

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

Partition tolerance — The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes.



https://en.wikipedia.org/wiki/CAP_theorem#/media/File:CAP_Theorem_Venn_Diagram.png

What are Column Store Databases?

Column-store databases are a type of database management system (DBMS) that organizes and stores data in columns rather than rows.

In a traditional row-store database, data for a single record or entry is stored in a contiguous block of memory. In a column-store database, the data for a single column is stored together.

Row-oriented

ID	Name	Grade	GPA
001	John	Senior	4.00
002	Karen	Freshman	3.67
003	Bill	Junior	3.33

Column-oriented

Name	ID
John	001
Karen	002
Bill	003

Grade	ID
Senior	001
Freshman	002
Junior	003

GPA	ID
4.00	001
3.67	002
3.33	003

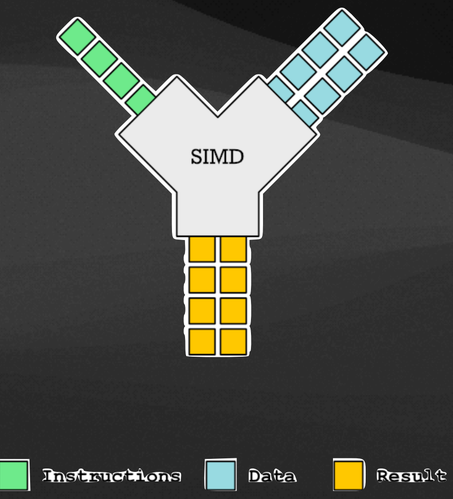
https://www.kdnuggets.com/2021/02/understanding-nosql-database-types-column-oriented-databases.html

Key characteristics of Column Store Databases

Compression — Data in columns often exhibit high compressibility because they contain similar or repetitive values compared to traditional row-based databases (1:10 vs 1:3).

Parallel Processing — Data in columns facilitate vectorized processing. This involves applying operations to multiple values simultaneously through SIMD (Single Instruction, Multiple Data) instructions, enabling parallel processing of diverse values within a column.

Scalability — Their distributed architecture allows for data partitioning at the column level, enabling parallel processing and horizontal scalability.

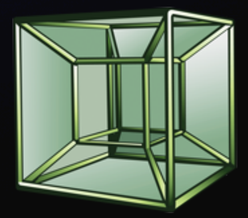


https://www.google.com/url?sa=i&url=https://johnnysswlab.com/crash-course-introduction-to-parallelism-simd-parallelism/&psig=AOvVaw33LuCNdkd2C2t1wwYchrQ7&ust=1704920305777000&source=images&cd=vfe&opi=89978449&ved=0CBMQjhxqFwoTCliGq5yZ0YMDFQAAAAAdAAAAABAD

Some NoSQL column store databases

APACHE HBASE

druid



HYPERTABLE INC

MariaDB Foundation

Apache Cassandra

Apache Cassandra

What is Cassandra?

Cassandra was originally created by *Avinash Lakshman* and *Prashant Malik*, for Facebook's inbox search feature and in July 2008 it was released as an open source program. It later became part of the Apache foundation.

Cassandra is an open-source,

- peer-to-peer distributed,
- easy scalable,
- fault tolerant,
- highly available and
- schema free wide-column database.

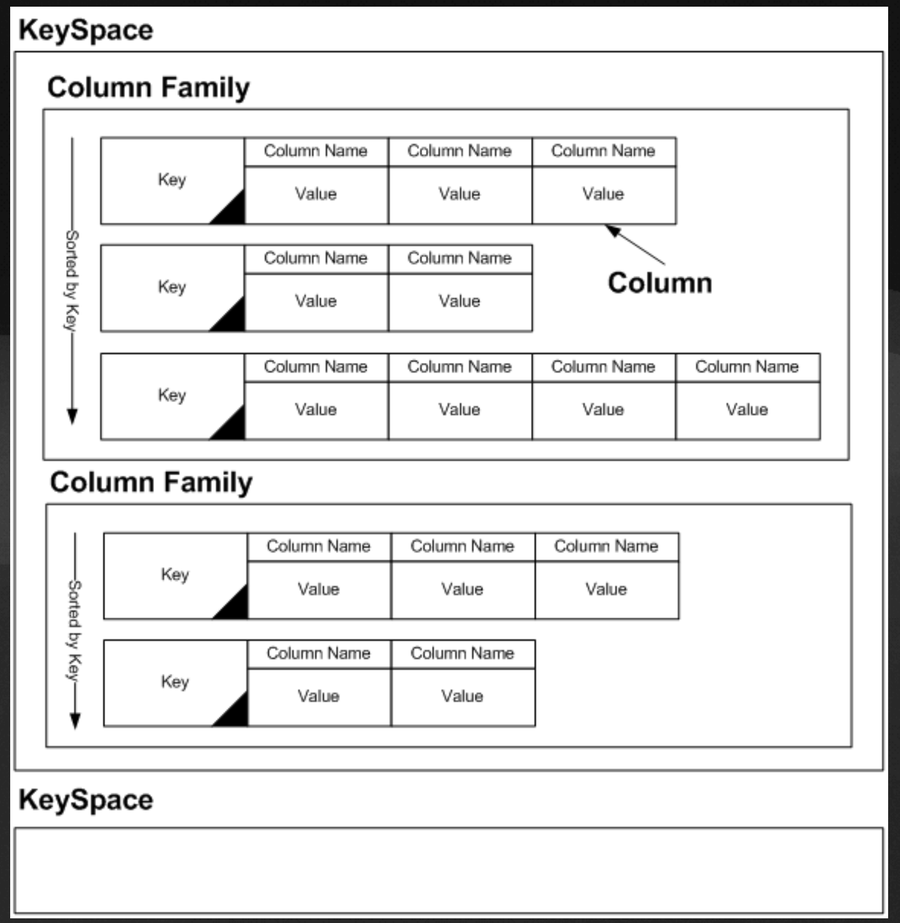
LinkedIn

Cassandra’s Data Model

Keyspaces and Column Families

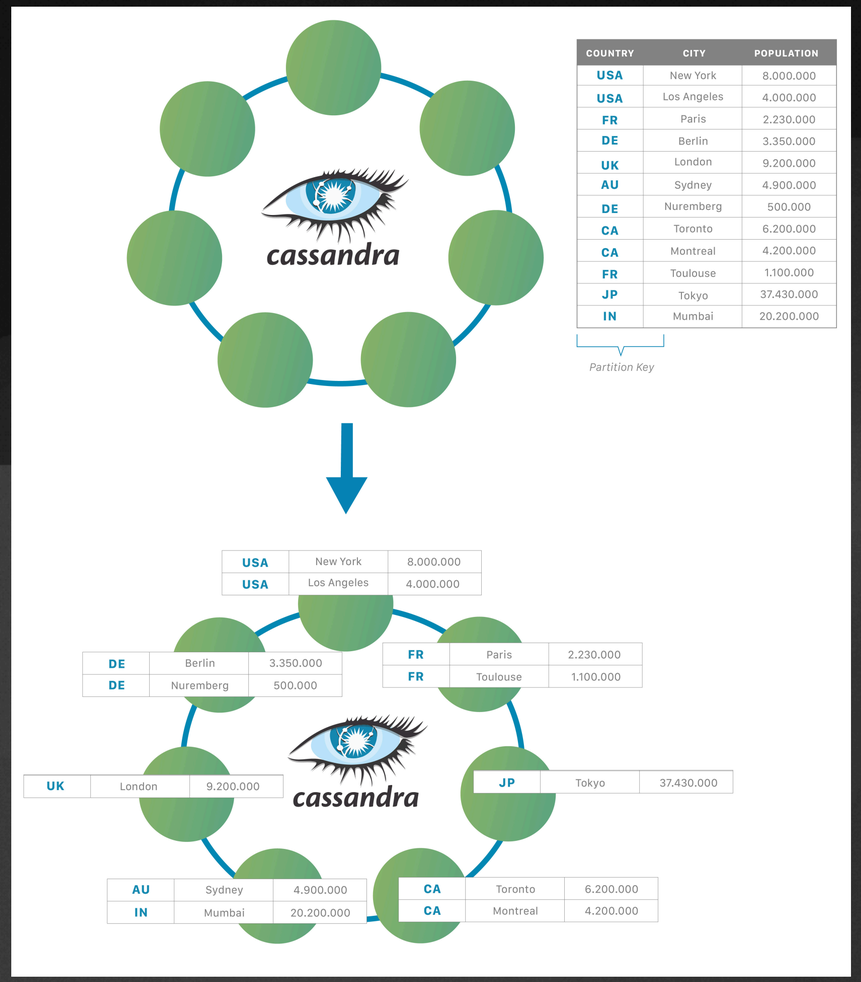
Keyspaces are the outermost container for at least one column family. In Keyspaces one can define the replication factor, the strategy to place replicas in the ring and column families.

Column families are containers that consist of ordered collections of rows, where each row is itself an ordered collection of columns. Although the column families are defined, the columns are not.



https://javamaster.wordpress.com/tag/apache-cassandra/

Cassandra
Architecture



https://cassandra.apache.org/_/cassandra-basics.html

A node represents a single instance of Cassandra, and stores data. Nodes are organized into a “ring”. The communication protocol between nodes is done through a peer-to-peer protocol called gossip.

Cassandra’s architecture is masterless and due to that, nodes provide the same functionality.

A collection of related nodes is called a data center, and a cluster contains one or more data centers.

Cassandra’s Main Attributes

Highly Available, Fault Tolerant and Scalable

Cassandra uses partitioning to distribute data across nodes. Each node is assigned specific tokens to manage data, determined by the partition key and hash function.

Any node can be the coordinator, assigning data to a partition. Through gossiping, nodes establish which node owns specific token ranges.

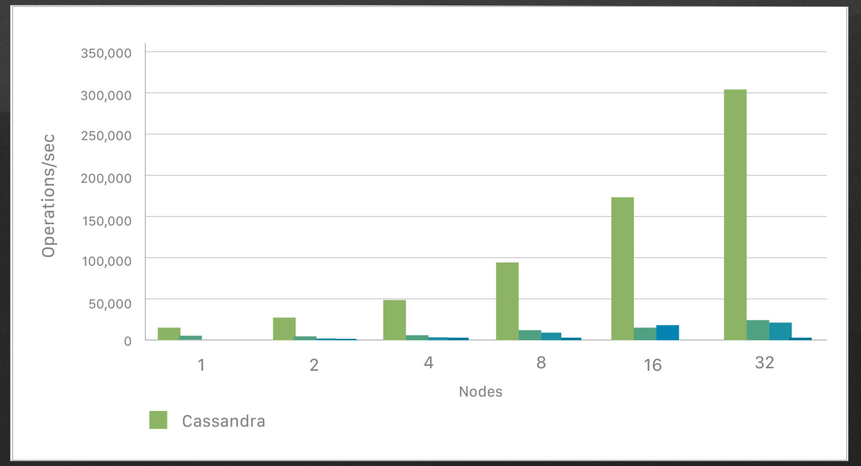
The coordinator node is dynamically assigned, preventing system fragility. Nodes continuously communicate about partition ownership, optimizing data distribution. This decentralized coordination enhances Cassandra's robustness and resilience.

Cassandra’s Main Attributes

Highly Available, Fault Tolerant and Scalable

Cassandra provides different policies for data replication in the cluster to ensure fault tolerance. Each piece of data is typically replicated to multiple nodes (N replicas), and these replicas are placed on (different) physical servers and racks, depending on the specified policy.

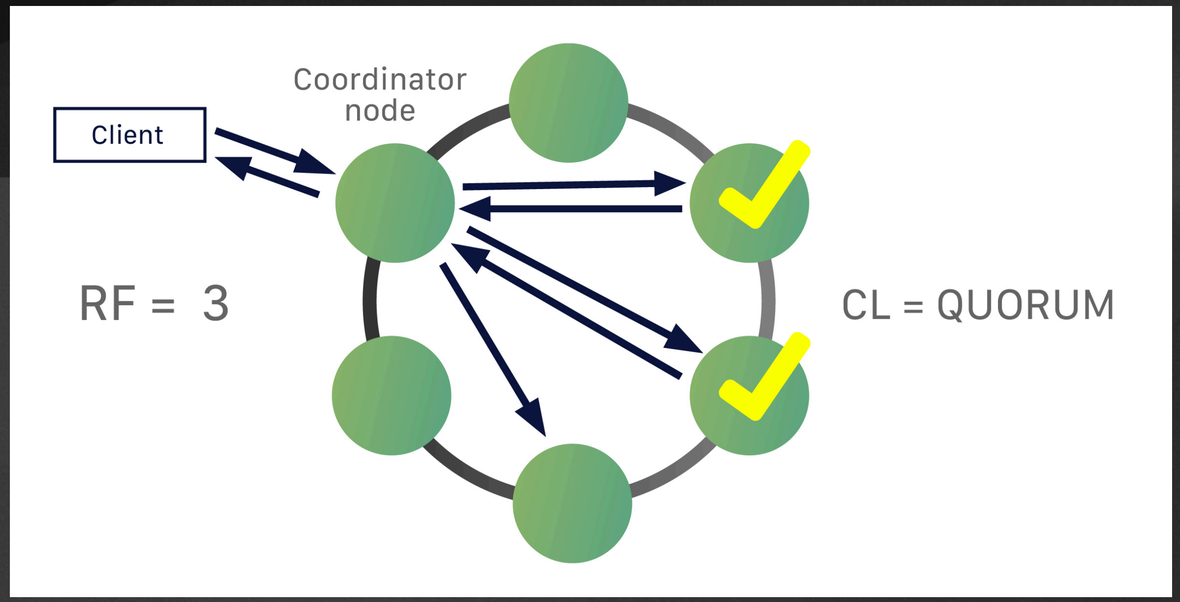
Cassandra is extremely scalable because it is based on nodes. This allows for horizontal scaling, and it ensures that each node is responsible for a specific range of data. Horizontal scaling means increasing performance or capacity by adding more machines or nodes to a system. This linear scalability is one of Cassandra’s keys strengths.



https://cassandra.apache.org/_/cassandra-basics.html

Cassandra
CAP theorem

Cassandra is by default AP (Available and Partition Tolerant), but its consistency can be configured. The consistency level is selected based on the replication factor.



Data is replicated out to three nodes. The consistency level is QUORUM, meaning that the majority replicas (replication factor/2 + 1) must acknowledge back to the coordinator in order for the query to be considered success.

https://cassandra.apache.org/_/cassandra-basics.html

Apache HBase

HBase

What is Base?

HBase was modeled after Google's Bigtable and is developed as a part of Apache Software Foundation's Apache Hadoop. It is a java implementation of Bigtable.

Apache HBase is an open source,

- non-relational,

-persistent,

-strictly consistent

-fault tolerant

-distributed database that runs on top of HDFS.

THE APACHE® SOFTWARE FOUNDATION

HBase

Key Components

It is the Hadoop database that means it has the advantages of Hadoop’s distributed file system and MapReduce model by default.

The store files are typically saved in the HDFS, which provides a scalable, persistent, replicated storage layer for HBase.

Since HBase uses Hadoop's HDFS for storage, and HDFS itself replicates data across multiple nodes, it is fault tolerant. Additionally, HBase has its own mechanism for replicating data between clusters.

HBase

Data Model

In HBase the table schema is only defining column-families which are key-value pairs.

Tables are a collections of rows, rows are a collections of column families, a column family is the collections of columns and these are collections of key-value pairs.

A table can have many column families, and each column family can have any number of columns. Column values are stores contiguously on the disk.

Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

https://www.tutorialspoint.com/hbase/hbase_overview.htm

HBase

Architecture

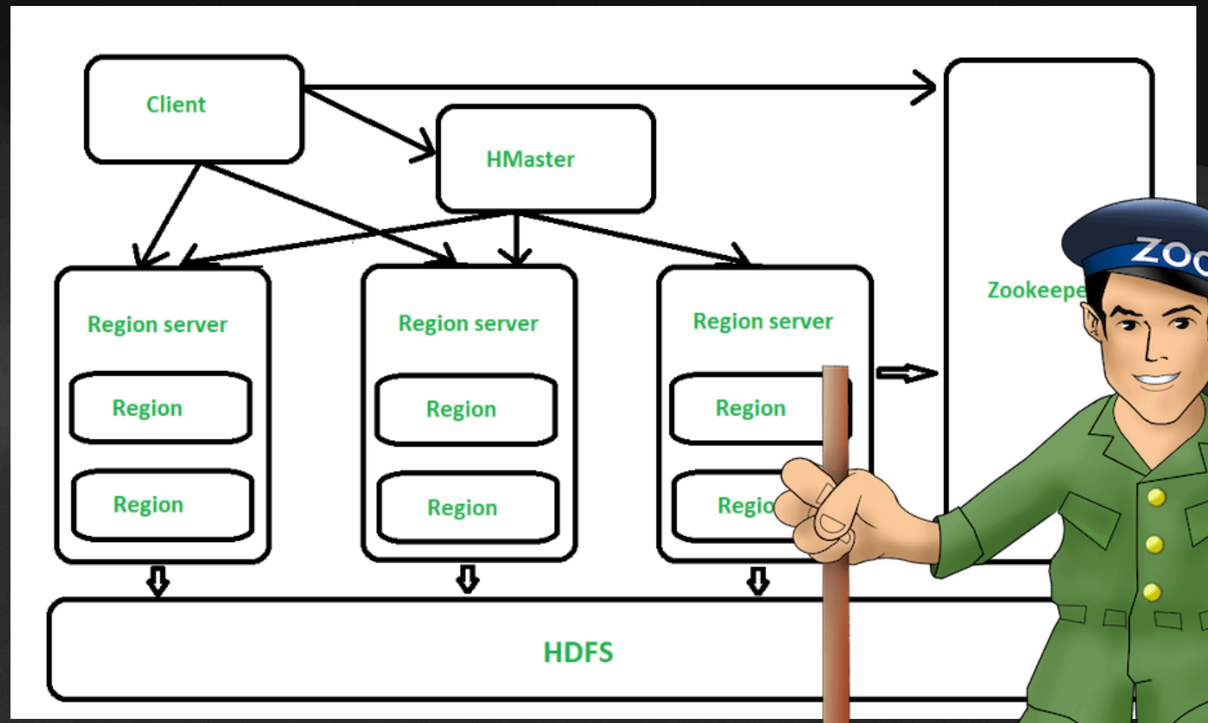
The **HMaster** or **Master Server** in HBase is responsible for assigning regions to region servers, ensuring load balancing, managing schema changes, creating tables, and overseeing Hadoop cluster operations, including failover handling.

The **Region Server** in HBase manages tables distributed across its regions, communicates with clients, and executes CRUD operations. It comprises components like Block cache, MemStore, WAL, and HFiles, each serving a specific role in handling read and write processes and storing data persistently.

HBase

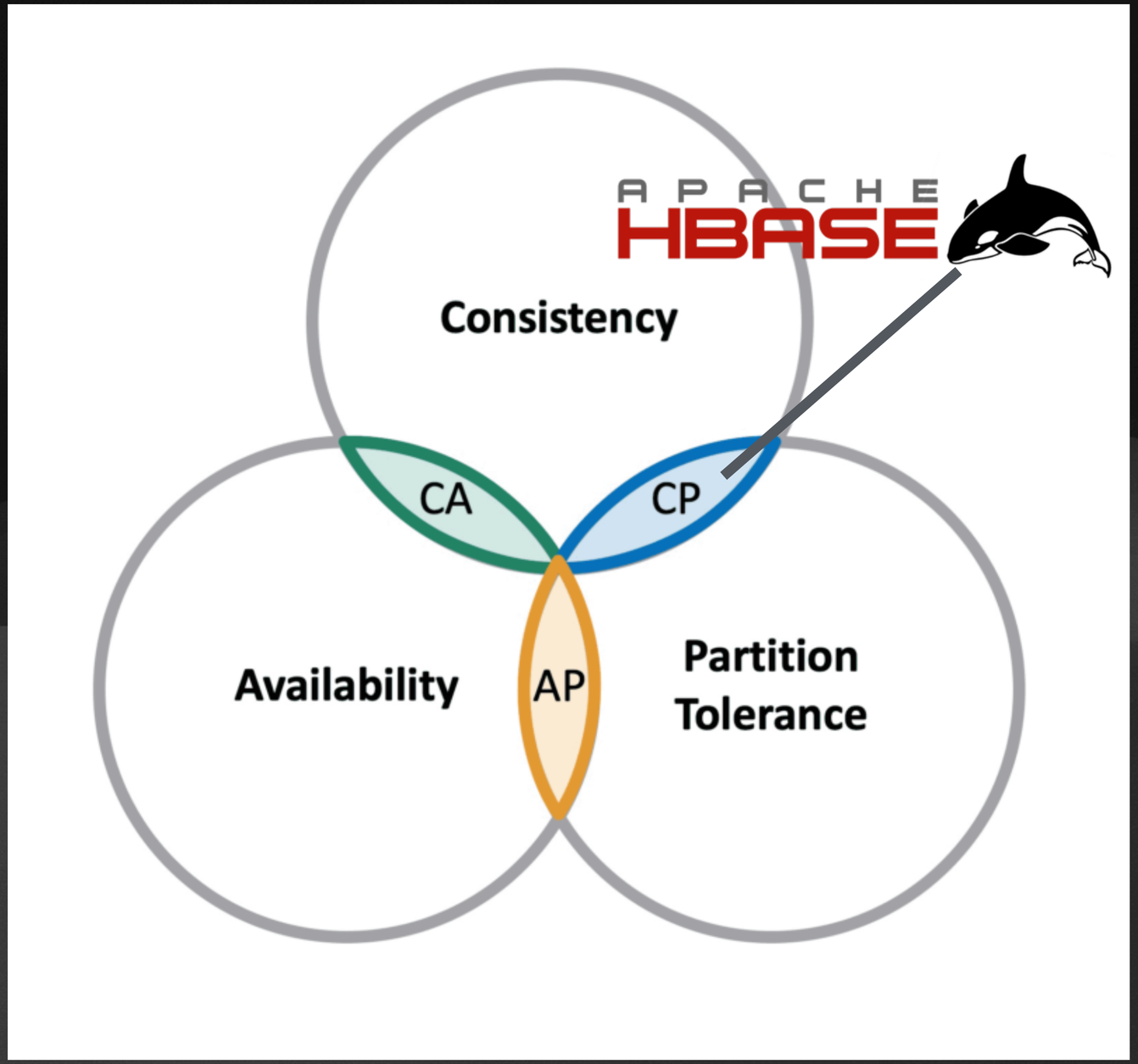
Architecture

Zookeeper plays a crucial role in HBase by serving as a coordination link between clients and HMaster. It helps recover from region server crashes, maintains information structure, acts as a contact point for client requests, and serves as a coordinator, tracking server status and managing composition information.



https://www.geeksforgeeks.org/architecture-of-hbase/

HBase
CAP theorem



Hbase follows strict consistency model, writes are written to single master, on CAP theorm Hbase focuses on Consistency and partition tolerance, offering strict consistency model for optimized reads.

https://www.google.com/url?sa=i&url=https://hazelcast.com/glossary/cap-theorem/&psig=AOvVaw1l1hzUFz1GPgqI4O8m4pID&ust=1704927883705000&source=images&cd=vfe&opi=89978449&ved=0CBMQjhxqFwoTCPDQj7m10YMDFQAAAAAdAAAAABAD

Apache Cassandra
vs
Apache HBase

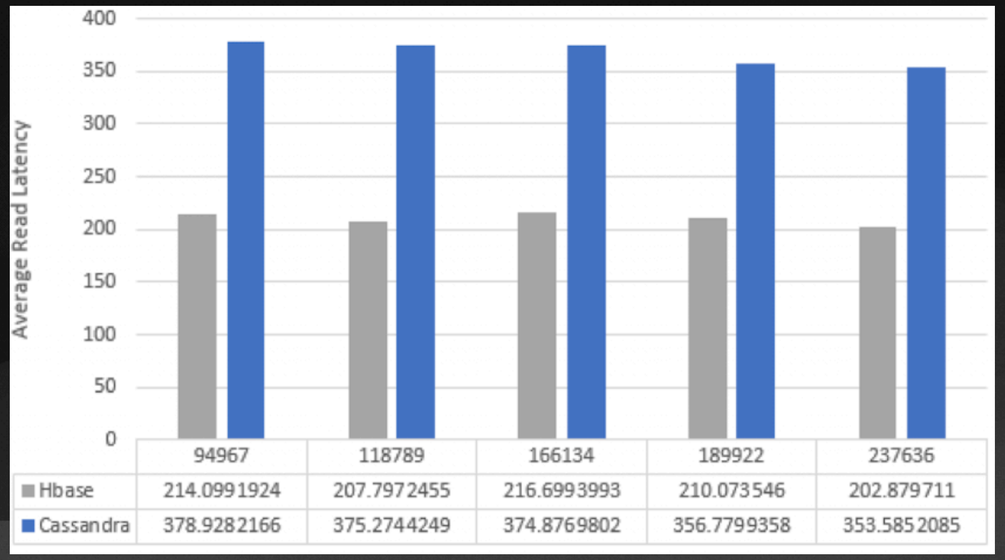
HBase vs cassandra

	Cassandra	HBase
Architecture	Peer-to-peer	Master/Slave
Consistency	Tunable	Strict
Availability	Very High	Failover Clustering*
Partitioning	Supports	Through Regions
Read	Depends on CL and RF	Fast

	Cassandra	HBase
Replication	The strategy can be defined	HDFS replication
Fault Tolerance	No single point of failure	Single point of failure
Cluster Communication	Gossip protocol	Apache Zookeeper
Map/Reduce	Can support with Haddop	Yes
Write	Fast	Slower

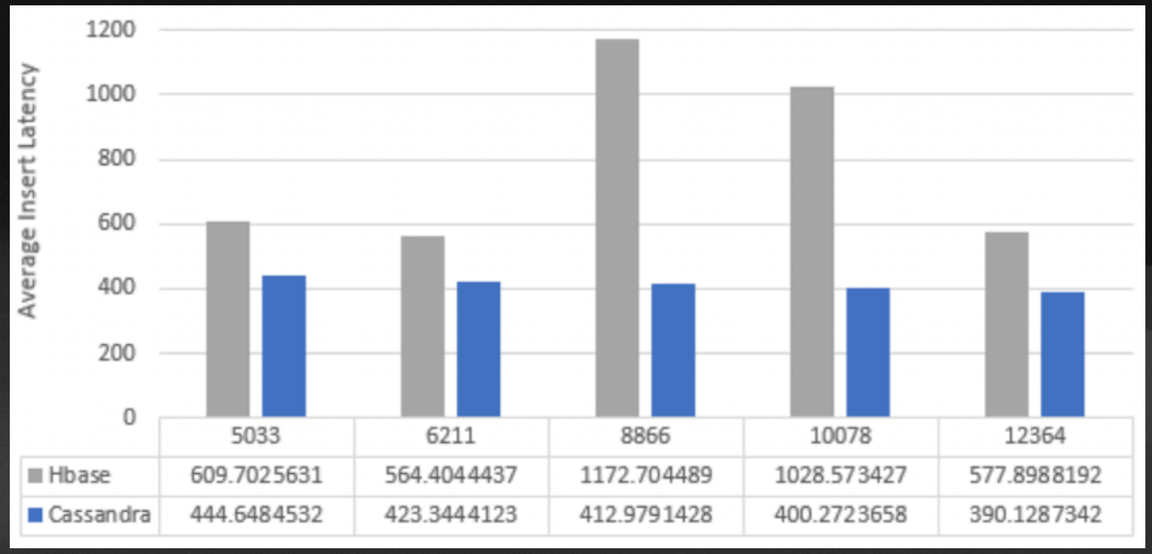
* another node takes over the responsibilities of the failed master to maintain continuous operations.

HBase vs Cassandra



	94967	118789	166134	189922	237636
Hbase	214.0991924	207.7972455	216.6993993	210.073546	202.879711
Cassandra	378.9282166	375.2744249	374.8769802	356.7799358	353.5852085

Read Operations



	5033	6211	8866	10078	12364
Hbase	609.7025631	564.4044437	1172.704489	1028.573427	577.8988192
Cassandra	444.6484532	423.3444123	412.9791428	400.2723658	390.1287342

Insert Operations

In a heavy reading workload with 95% reading operations and 5% insert operations executed in several iterations, Cassandra has higher (average) read latency compared to HBase (left figure).

Although Cassandra's performance on insert operations is consistently across different counts of inserts.

Conclusions

When are NoSQL column store databases a good choice?

Column-family databases are best suited for **data warehousing** applications. These applications require **analyzing** large amounts of data for **business intelligence** with a high write throughput, and column-family databases completely take charge of it.

Cassandra or HBase?

Cassandra can be used for applications requiring **faster writes** and **high availability**. Cassandra is easier to set up since its a **standalone** product.

Hbase suits the scenarios where hadoop **map reduce** is useful for bulk read and load operations hbase offers optimized read performance with hadoop platform. Additionally, HBase is suitable in applications that require data **consistency**. HBase relies on several Hadoop components to run and is a better option if someone already uses **Hadoop**, than Cassandra.

References

- Daniel, M. S., Abadi, D. J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Rasin, A., Tran, N., Zdonik, S. & Ma, W. (2005). C-Store: A Column-oriented DBMS
- Lakshman, A., & Malik, P. (2010). Cassandra: a decentralized structured storage system. *ACM SIGOPS operating systems review*, *44*(2), 35-40.
- Nayak, A., Poriya, A., & Poojary, D. (2013). Type of NOSQL databases and its comparison with relational databases. *International Journal of Applied Information Systems*, *5*(4), 16-19.
- Sharma, V., & Dave, M. (2012). Sql and nosql databases. International Journal of Advanced Research in Computer Science and Software Engineering, 2(8).
- Jakkula, P. (2020). HBase or Cassandra? A comparative study of nosql database performance. International Journal of Scientific and Research Publications, 10(3), 808-820.

References

- https://www.techtarget.com/searchdatamanagement/tip/NoSQL-database-types-explained-Column-oriented-databases
- https://cassandra.apache.org/_/cassandra-basics.html
- https://airbyte.com/data-engineering-resources/columnar-storage
- https://www.tutorialspoint.com/hbase/hbase_architecture.html
- https://aws.amazon.com/compare/the-difference-between-cassandra-and-hbase/
- https://www.kdnuggets.com/2023/03/nosql-databases-cases.html

Thank you