Unit # 2

- Java Interface.
- Pig & Hive,
- Mongo DB , Dynamo,
- Cassendra,
- Voldemort.

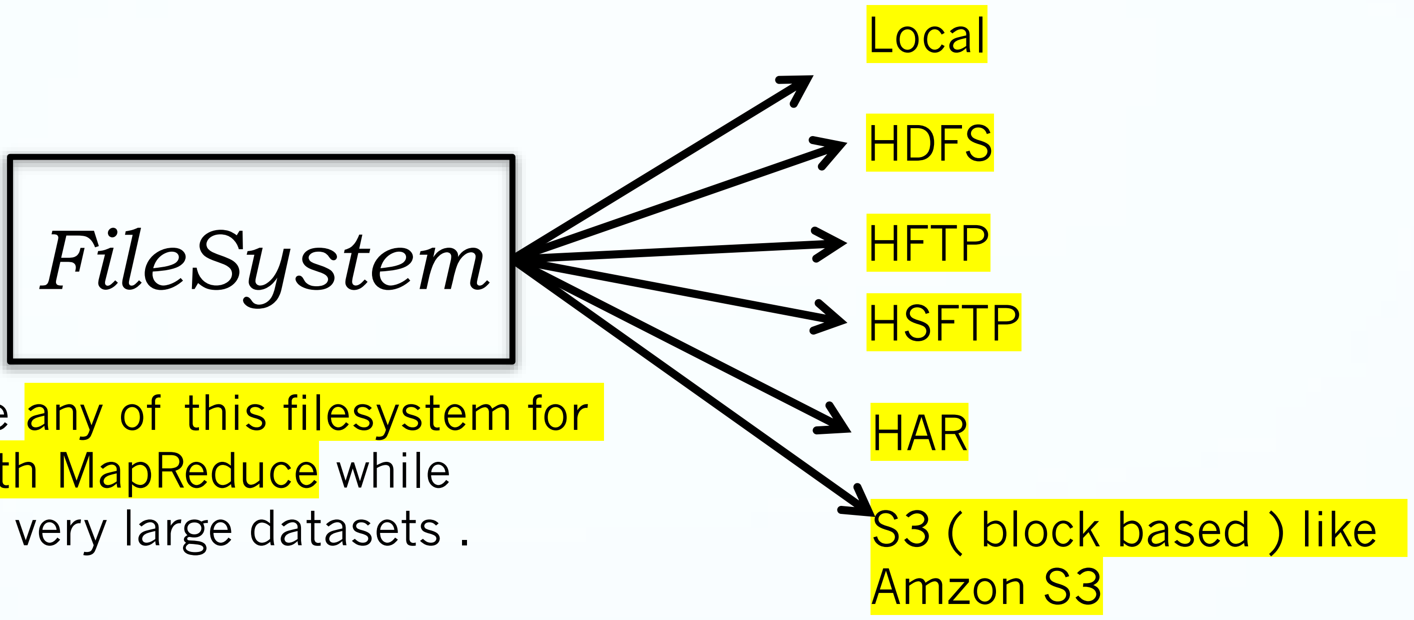
Java Interface

- In Java programming, an **interface** is a collection of abstract methods (methods without bodies) and constant fields (static and final). It serves as a blueprint for classes that implement it, allowing different classes to share common behavior.
- Hadoop itself is written in Java, so most Hadoop operations / interactions are mediated through the Java API to provide a modular, extensible framework for large-scale distributed data processing.
- It provide pluggable components that allow developers to customize data handling, serialization, I/O,reading data, writing data, processing key-value pairs , without altering the core architecture.

For Example : *FileSystem* : < org.apache.hadoop.fs.FileSystem >

- Hadoop is capable of running various file systems and HDFS is just one single implementation that out of all those file systems. The *FileSystem* abstract class in Hadoop provides a generic interface for Input/Output (I/O) across different file systems. It provides a unified API for working with various file systems regardless of the underlying file system (local, distributed, or cloud-based).
- Some of the common implementations include:
  - *Local File System* (LocalFileSystem): For working with files on locally connected disk with client-side
  - *HDFS* (DistributedFileSystem): For interacting with the Hadoop Distributed File System.
  - The *HFTP* filesystem provides read-only access to HDFS over HTTP.
  - The *HAR* file system reduces the memory usage of NameNode by registering files in Hadoop HDFS.

Java Interface (contd)



We can use any of this filesystem for working with MapReduce while processing very large datasets .

In Hadoop, many abstract classes and interfaces play a crucial role in providing a generic, pluggable framework that developers can extend to customize data processing. Some examples listed below that can be explored further.

2. Partitioner < org.apache.hadoop.mapreduce >

The Partitioner abstract class is responsible for controlling how Mapper outputs are distributed to Reducers. It defines the logic that determines which Reducer will process a given key.

Java Interface (contd)

3. Mapper <org.apache.hadoop.mapreduce>
The Mapper abstract class defines the map phase of a MapReduce job. It processes input key-value pairs and produces intermediate key-value pairs for the Reducer.

4. Reducer <org.apache.hadoop.mapreduce>
The Reducer abstract class defines the reduce phase of a MapReduce job. It processes intermediate key-value pairs generated by the Mapper and produces final output key-value pairs.

5. Writable < org.apache.hadoop.io >
The Writable interface is the core serialization mechanism in Hadoop. All custom data types that need to be serialized across the network or stored in HDFS must implement this interface.

6. WritableComparable < org.apache.hadoop.io >
WritableComparable is an extension of the Writable interface that adds the ability to compare objects. This interface is used for keys in Hadoop because keys need to be compared during sorting in the MapReduce framework.

7. Combiner < org.apache.hadoop.mapreduce >
The Combiner class works like a mini-Reducer and is optional in a MapReduce job. It processes the Mapper output before sending it to the Reducer, reducing the amount of data transferred between Mappers and Reducers.

Apache PIG

- One limitation of MapReduce is that the development cycle is very long. Writing the reducer and mapper, compiling packaging the code, submitting the job and retrieving the output is a time-consuming task. Apache Pig reduces the time of development.
- 200 lines of Java code can be written in only 10 lines using the Pig Latin language. Programmers who have SQL knowledge needed less effort to learn Pig Latin and its also helpful to those who do not have JAVA language expertise.
- Apache Pig was developed way back in 2006, by Yahoo's researchers. At that time, the main idea to develop Pig was to execute the MapReduce jobs on extremely large datasets.

Features of Apache PIG

✓ Easy to learn, read and write. Especially for SQL-programmer, Apache Pig is a boon.
✓ We can make our own process and user-defined functions(UDFs) written in python, java or other programming languages .
✓ It very easily integrates with Apache HIVE , Apache Hadoop and Apache Zookeeper thus allowing to take advantage of the ecosystem.
✓ Its datas structure is rich , multi valued and is also nested.
✓ It can handle analysis of both structured and unstructured data

Apache PIG (contd)

Key Components of Apache Pig

1. Pig Latin

Latin is the core language of Apache Pig. It is a dataflow language that allows users to describe Extract, Transform, Load (ETL) operations over large datasets. Few Key commands to get type of operations possible through Apache PIG are decribed below :

a) *LOAD:* Load data from a file system like HDFS
b) *FOREACH:* Iterate over each record and apply transformations.
c) *FILTER:* Filter records based on a condition.
d) *GROUP* *:* Group data based on a key.
e) *JOIN:* Join two datasets on a common key.
f) *DUMP:* Display the output on the console.
g) *STORE:* Write the results back to a file system like HDFS

2. GRUNT Shell

The Grunt Shell is Pig's interactive command-line interface (CLI) where users can write Pig Latin commands interactively and immediately execute them. It is useful for exploring datasets and debugging Pig scripts.

3. Pig Execution Modes

There are two modes of execution in Pig. They are

Apache PIG (contd)

There are two modes of execution in Pig. They are

Local Mode :

In this mode, Pig runs on a **local machine** and accesses the local file system. This mode is useful for small datasets or testing Pig scripts locally.

MapReduce mode:

In this mode, Pig runs on a **Hadoop cluster**, utilizing HDFS for storage and executing MapReduce jobs for data processing. This mode is used for large-scale data processing in production environments.

4. User Defined Function

Pig allows users to define their own custom functions called User Defined Functions (UDFs) in languages like Java, Python, or JavaScript. UDFs can be used to perform complex transformations, filtering, or aggregations that are not available in built-in Pig functions.

5. Joining Datasets

In Hadoop data analysis we have a very frequent used operation to combine two or more datasets based on a common key. combine two or more datasets based on a common key. Joining data can be quite complex and time-consuming in MapReduce. Apache Pig simplifies the process through high-level operations under the hood by converting the Pig Latin join statements into efficient MapReduce jobs.

Apache HIVE

Apache Hive is a data warehousing and SQL-like query engine built on top of the Hadoop ecosystem. It allows users to easily query, analyze, and manage large datasets stored in Hadoop's HDFS (Hadoop Distributed File System) using a language called HiveQL (Hive Query Language), which is similar to SQL.

- There are two goals of HIVE:-
  - An SQL-based declarative language that also allows to plug in their user defined scripts and programs in the SQL query.
  - Second, to provide a centralized metadata store (Hadoop based) of all the datasets in the organization.
- It provides functionalities like string manipulation, date manipulation, type conversion, conditional operators, mathematical functions, and others

- Apache Hive supports the analysis of large datasets stored in Hadoop's HDFS and compatible file systems such as Amazon S3, Azure Blob Storage, Azure Data Lake Storage, Google Cloud Storage, and Alluxio.
- The key components of the Apache Hive architecture are the
  - ✓ Hive Server ,
  - ✓ Hive Query Language (HQL),
  - ✓ External Apache Hive Metastore,
  - ✓ Hive Beeline Shell

1. HIVE server :

MapReduce mode:

In this mode, Pig runs on a Hadoop cluster, utilizing HDFS for storage and executing

Apache HIVE (contd)

The server also supports the Hive optimizer and Hive compiler to streamline data extraction and processing.

2. HIVE Query Language (HQL) :

Hive provides a SQL-like querying language but with extensions to support large-scale data processing on Hadoop for querying and managing data. This makes it easy for users who are already familiar with SQL to interact with big data stored in HDFS or other Hadoop-supported file systems. It supports most of the common SQL features such as SELECT, JOIN, GROUP BY, and ORDER BY.

3. HIVE MetaStore :

It is the central repository of the Apache Hive infrastructure, where all of the Hive's metadata is stored about the databases, tables, partitions, columns, and data types in Hive. It also keeps track of where the actual data is stored in HDFS.

3. HIVE Beeline Shell :

Hive has its own built-in command-line interface where users can run HQL statements. The shell also runs Hive JDBC and ODBC drivers and so can conduct queries from an Open Database Connectivity or Java Database Connectivity application.

Mongo DB

Structured Data	UnStructured Data
Can be displayed in to rows and columns in DBMS or a RDBMS	Cannot be displayed strictly as rows and columns
Numbers , Chars , Dates and Strings	Images , Audio clips , Video clips , word processing files , E-mails , Spreadsheets and many more …
Estimated ~ 20 % of Enterprise data (Gartner)	Estimated 80% of Enterprise Data (Gartner)
Requires less Storage	Requires ever increasing huge storage
Easier to manage and Protect with legacy solutions	More difficult to manage and Protect.
Can be queried using SQL (Structured Query Language)	Searched with specialized tools (e.g., full-text search, data mining algorithms)
Schema is rigid and must be defined before data entry	Schema cannot be frozen and in fact anything can be stored as first go and schema applied later (schema-on-read)

Mongo DB (contd)

Hall mark of Unstructured Data :

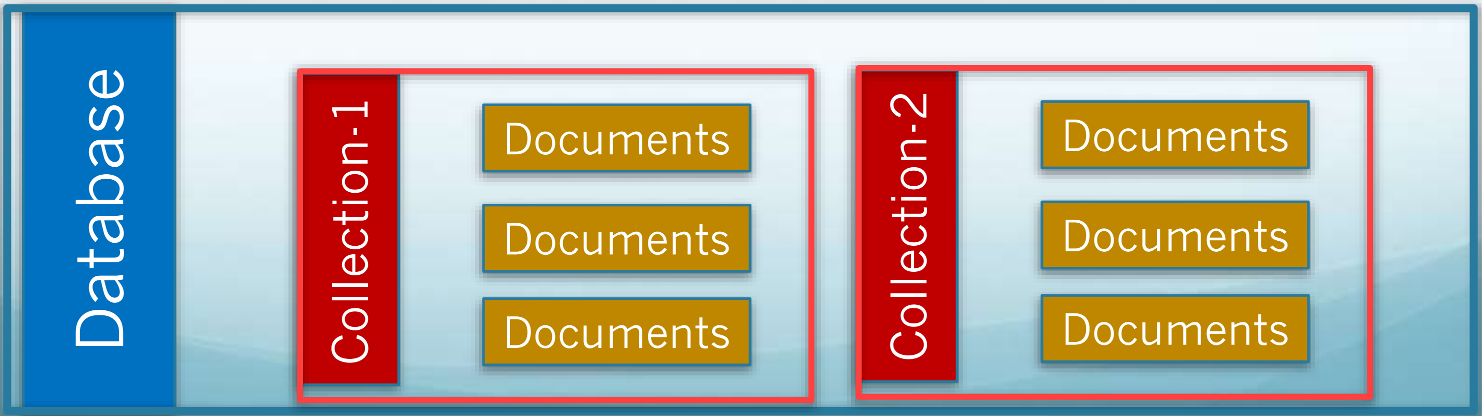
Unstructured data doesn't have consistent structure or schema and can be found in a variety of formats.

These formats can include everything from videos, images, and audio files to documents, web logs, sensor data, and binary data.

Significant amounts of unstructured data contain natural language text that may include misspellings, abbreviations, grammatical errors, and slang as it isn't standardized before ingestion.

Mongo-DB ?

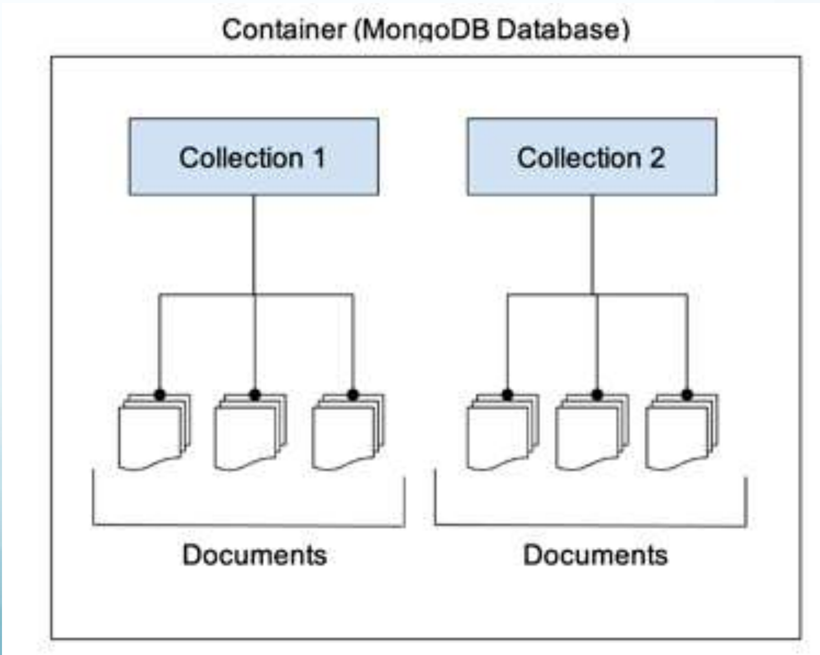
MongoDB is an open-source document-oriented database that is designed to store a large scale of **Unstructured data**. It is categorized hence as a NoSQL (Not only SQL) database because the storage and retrieval of data in the MongoDB are not in the form of tables. The MongoDB database we are allowed to create multiple databases and multiple collections.



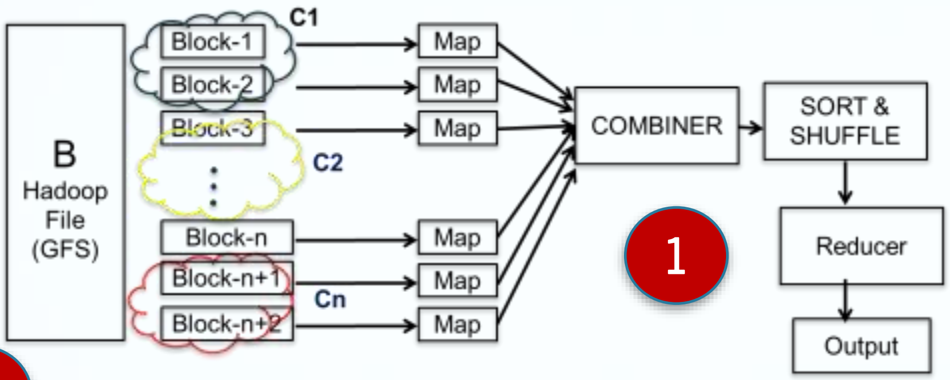
Mongo DB (contd)

- Once the MongoDB server is started we can have create multiple databases and the data is stored in these databases. There are no rows and tables in its databases but the data is stored as the collections and documents.
- The documents are created using the fields. Fields are key-value pairs in the documents, it is just like columns in the relation database. The value of the fields can be of any data types like double, string, boolean, etc.
- The data stored in the MongoDB is in the format of BSON documents. BSON stands for Binary representation of JSON documents. MongoDB server converts the JSON data into a binary form in to BSON.
- A single collection can contain multiple documents and since they are schema-less it means it is not necessary that one document is similar to another.
- The maximum size of the BSON document is 16MB

- The MongoDB database is developed and managed by MongoDB Inc and the company provides official driver support for all the popular languages like C, C++, C#, and .Net, Go, Java, Node.js, Perl, PHP, Python, Motor, Ruby, Scala, Swift, Mongoid. So we can create an application using any of these languages.
- It has become one of the most popular database and used by Facebook, Nokia, eBay, Adobe, Google.



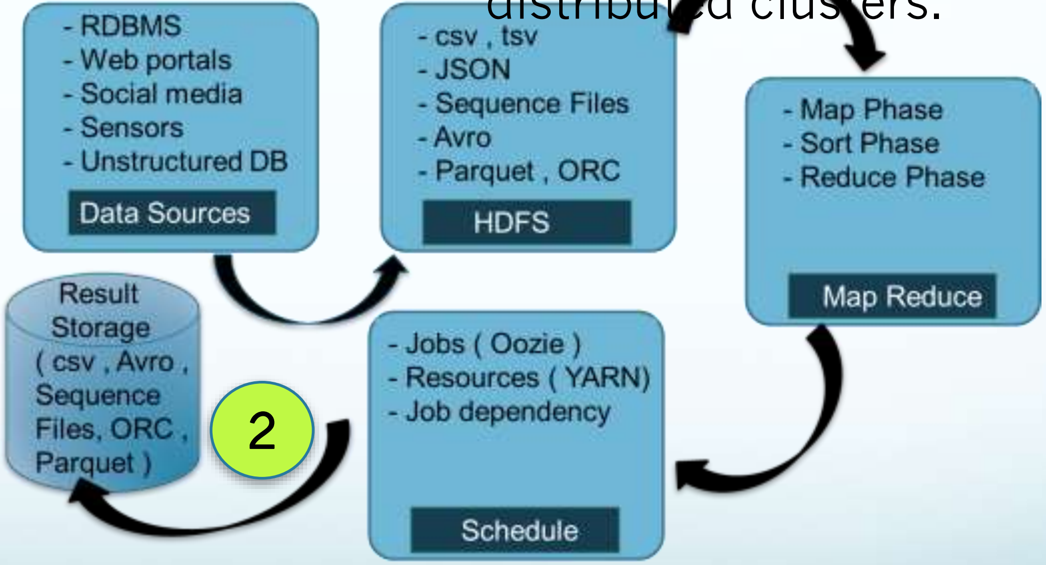
Mongo DB in Big Data Analysis Lifecycle



MongoDB is not optimal for large-scale batch processing, which requires distributed computation across potentially vast datasets. Hadoop is better suited for these tasks because they are designed to process terabytes to petabytes of data using distributed clusters.

1

MongoDB can act as an ingestion layer, especially for real-time, high-velocity data such as logs, social media feeds, and sensor data. MongoDB can also store the ingested data without requiring a predefined schema



2 MongoDB is optimized for low-latency, real-time reads and writes, making it ideal after running **MapReduce** jobs. From here the business applications can quickly query MongoDB to build **real-time dashboards** or serve customer-facing applications where speed and flexibility are essential.

Dynamo

- DynamoDB is a cloud-native NoSQL primarily key-value database developed at Amazon. It is a cloud-native in that it does not run on-premises or even in a hybrid cloud; it only runs on Amazon Web Services (AWS).
- It is NoSQL in that it does not support ANSI Structured Query Language (SQL). Instead, it uses a proprietary API based on JavaScript Object Notation (JSON). This API is invoked through AWS Software Developer Kits (SDKs) for DynamoDB .
- It does not support relational database management systems (RDBMS) methods to join tables through foreign keys.
- Outside of Amazon, the world doesn't know much about the exact nature of this database. The original **Dynamo paper** (released by Amazon in 2007) details the concept. This white paper cited and also contrasted itself from Google's Bigtable (2006).
- On performance front , for example, DynamoDB delivers consistent single-digit millisecond performance for a shopping cart use case, whether there is 10 or 100 million users.

Characteristics of DynamoDB

1. Serverless

With DynamoDB, you don't need to provision any servers, or patch, manage, install, maintain, or operate any software. DynamoDB provides zero downtime maintenance. It has no versions (major, minor, or patch), and there are no maintenance windows.

Amazon manages all the underlying infrastructure.

DynamdB(contd)

2. NoSQL

DynamodB is a NoSQL database. It is purpose-built to deliver improved performance, scalability, manageability, and flexibility compared to traditional relational databases. But it still provides strong read consistency and ACID transactions to build enterprise-grade applications.

3. Fully Managed

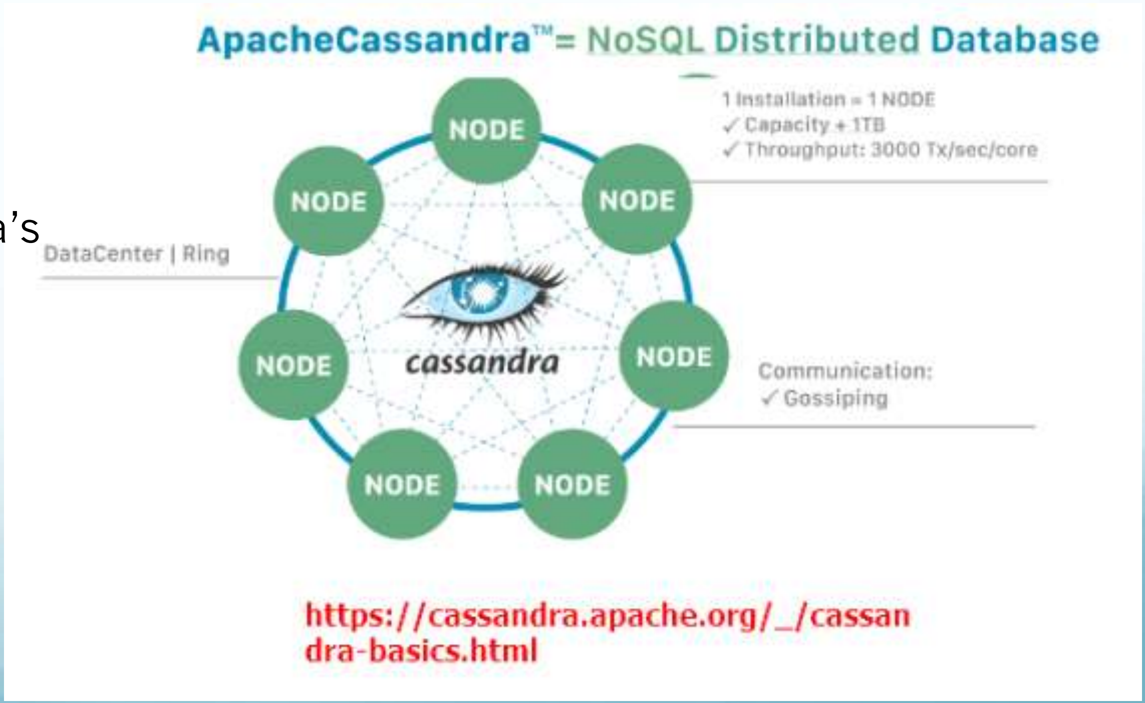
It handles setup, configurations, maintenance, high availability, hardware provisioning, security, backups, monitoring, and more. This ensures that when you create a DynamoDB table, it's instantly ready for production workloads. DynamoDB constantly improves its availability, reliability, performance, security, and functionality without requiring upgrades or downtime.

4. Security

DynamoDB utilizes secure full control access to DynamoDB resources. It can centrally manage permissions that control which DynamoDB users can access resources. By default, DynamoDB encrypts all customer data in its database and decrypt transparently when demanded. Even with this overhead its performance of single digit latency is maintained.

Cassandra

- Apache Cassandra is a highly scalable, distributed NoSQL database developed at Facebook and designed for handling large volumes of data across many commodity servers without a single point of failure. It was developed to power Facebook inbox search, Cassandra has become a popular choice for organizations looking to manage massive amounts of data with high availability and no downtime.
- By design, it is lightweight, open-source, non-relational, and largely distributed. Counted among their strengths are horizontal scalability, distributed architectures, and a flexible approach to schema definition.
- Cassandra can run on multiple machines since it is a distributed database but to user it appears as a unified single machine.It runs as a masterless architecture – any node in the database can provide the exact same functionality as any other node – contributing to Cassandra's robustness and resilience.


ApacheCassandra™= NoSQL Distributed Database
1 Installation = 1 NODE
✓ Capacity + 1TB
✓ Throughput: 3000 Tx/sec/core
DataCenter | Ring
NODE
cassandra
Communication:
✓ Gossiping
https://cassandra.apache.org/_/cassandra-basics.html

- The nodes communicate with one another through a protocol called gossip, which is a process of computer peer-to-peer communication.

Cassandra (contd)

Key Features of Cassandra

1. Distributed Architecture:

Unlike traditional databases that follow a master-slave architecture, Cassandra uses a peer-to-peer architecture. All nodes are equal, meaning there's no single master node. This setup eliminates single points of failure and ensures high availability. Unlike traditional databases that follow a master-slave architecture, Cassandra uses a peer-to-peer architecture. All nodes are equal, meaning there's no single master node. This setup eliminates single points of failure and ensures high availability.

2. Scalability :

Cassandra scales horizontally by adding more nodes to the cluster. As nodes are added, Cassandra maintains performance because of its linear scaling properties. New nodes can be added dynamically without stopping the system, ensuring uninterrupted service availability.

3. High Availability and Fault Tolerance:

Cassandra uses a configurable replication strategy to replicate data across multiple nodes, ensuring that even if a node or several nodes fail, data remains accessible.

4. Consistency and Availability Trade-offs::

Cassandra offers tunable consistency, allowing developers to choose between strong

Cassandra (contd)

to medium to low i.e that governed by Consistency Levels, which can be set at the query level. Thus Consistency Vs Availability can be bargained as per CAP theorem.

Application areas :
Cassandra's architecture makes it ideal for use cases that demand high availability, scalability, and fast read/write operations:

Time-Series Data: Applications that involve time-series data, such as monitoring tools (e.g., monitoring server health, IoT data) and logging systems, benefit from Cassandra's performance and ability to handle large amounts of sequentially growing data efficiently.

Real-Time Data and Analytics: For applications requiring real-time data access, like social media platforms, e-commerce personalization engines, or recommendation systems, Cassandra provides fast read and write capabilities.

IoT Applications: IoT systems generate massive amounts of data from devices in various geographic locations. Cassandra's multi-data center support ensures data is available quickly from anywhere.

Messaging and Event Logging: Systems that require logging of events, messages, or user actions in real-time use Cassandra due to its ability to handle high-velocity writes without sacrificing performance.

Voldemart

Voldemort is an open-source , NoSQL distributed key-value storage system developed by LinkedIn. It is designed for scalability and performance and inspired by Amazon's Dynamo. It is used to manage large-scale distributed data while ensuring high availability and fault tolerance. Voldemort is suitable for applications that require real-time, low-latency access to large volumes of data.

Key Features of Voldemart

- Data is automatically replicated over multiple servers.
- Data is partitioned using consistent hashing, distributing it evenly across all nodes in the cluster. This ensures balance and even load distribution, which is critical for performance and scalability.
- It supports an eventual consistency model, meaning data updates propagate through the cluster over time, eventually reaching consistency. This is beneficial for applications that prioritize availability and partition tolerance over immediate consistency.
- Conflicts are managed using vector clocks, a versioning mechanism that tracks multiple versions of data, allowing the system to resolve conflicts when data nodes diverge.
- Voldemort supports In-Memory storage for high-speed access . It supports REST API for accessing and manipulating data, making it easy for developers to interact with the database using standard web protocols.

Voldemart (contd)

Application areas :

Content Delivery : Suitable for caching and delivering content quickly, Voldemort is often used for user profiles, recommendation systems, and session data, where fast read and write operations are crucial.

Global Scale Applications: With support for multi-data center replication, Voldemort is effective for applications needing global data distribution and low latency.

- Its limitation is that it has got limited support for complex queries compared to traditional databases or more sophisticated NoSQL solutions like Cassandra or MongoDB.
- Like Cassendra , Its tunable consistency can add complexity in application design and conflict resolution.

Thanks