Chapter 3: HDFS, Hive and HiveQL, HBase (part 1)

—

By Ms. Tina D’abreo

Content

- HDFS-Overview, Installation and Shell, Java API
- Hive Architecture and Installation, Comparison with Traditional Database,
- HiveQL Querying Data, Sorting And Aggregating,
- Map Reduce Scripts, Joins & Sub queries
- HBase concepts, Advanced Usage, Schema Design, Advance Indexing, PIGGrunt – pig data model – Pig Latin – developing and testing Pig Latin scripts
- Zookeeper , how it helps in monitoring a cluster
- Build Applications with Zookeeper and HBase

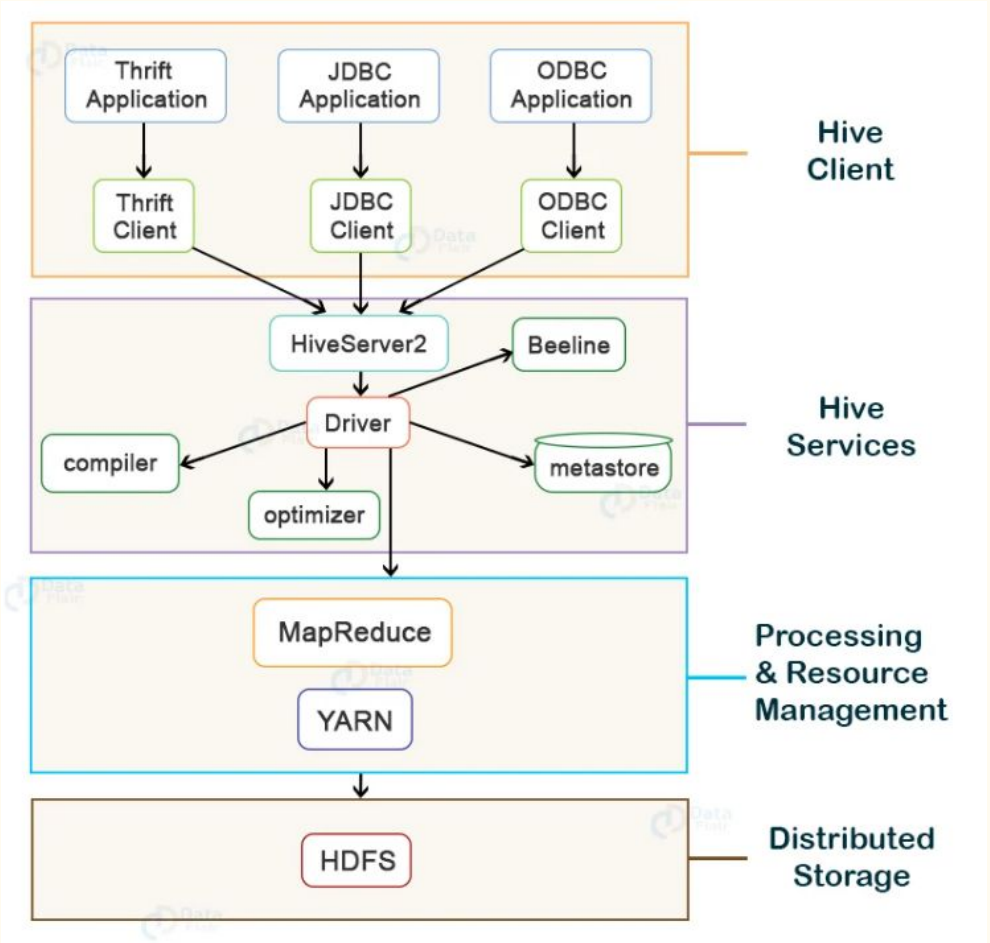
Hive

- Hive is a data warehouse infrastructure tool to process structured data in Hadoop.
- It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.
- Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive.
- Features:
    - It is familiar, fast, scalable and extensible
    - It stores database schema; the processed data is stored into HDFS
    - It provides SQL- type language for querying called HiveQL or HQL
- The Facebook open-source data warehousing tool Apache Hive was designed to eliminate the job of writing the MapReduce Java program.
- Facebook developed it to decrease the amount of code it requires.
- Apache Hive is an open-source data warehousing tool for performing distributed processing and data analysis. It was developed by Facebook to reduce the work of writing the Java MapReduce program.
- Apache Hive uses a Hive Query language, which is a declarative language similar to SQL. Hive translates the hive queries into MapReduce programs.

Hive

- Hive is a data warehouse infrastructure tool to process structured data in Hadoop.
- It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy.
- Initially Hive was developed by Facebook, later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive.
- Features:
    - It is familiar, fast, scalable and extensible
    - It stores database schema; the processed data is stored into HDFS
    - It provides SQL- type language for querying called HiveQL or HQL

Hive - Architecture



Hive - Architecture - Hive Client

- Hive supports applications written in any language like Python, Java, C++, Ruby, etc. using JDBC, ODBC, and Thrift drivers, for performing queries on the Hive.
- Hive client application can be written in any language of his own choice.
- Hive clients are categorized into three types:
  - ***Thrift Clients***
    - The Hive server is based on Apache Thrift so that it can serve the request from a thrift client.
  - ***JDBC client***
    - Hive allows for the Java applications to connect to it using the JDBC driver. JDBC driver uses Thrift to communicate with the Hive Server.
  - ***ODBC client***
    - Hive ODBC driver allows applications based on the ODBC protocol to connect to Hive. Similar to the JDBC driver, the ODBC driver uses Thrift to communicate with the Hive Server.

Hive - Architecture - Hive Services

- ***Beeline -***
  - The Beeline is a command shell supported by HiveServer2, where the user can submit its queries and command to the system.
  - It is a JDBC client that is based on SQLLINE CLI (pure Java-console-based utility for connecting with relational databases and executing SQL queries).
- ***Hive Server2 -***
  - HiveServer2 is the successor of HiveServer1. HiveServer2 enables clients to execute queries against the Hive.
  - It allows multiple clients to submit requests to Hive and retrieve the final results. It is basically designed to provide the best support for open API clients like JDBC and ODBC.

Hive - Architecture - Hive Services

- ***Hive Driver -***
  - The Hive driver receives the HiveQL statements submitted by the user through the command shell.
  - It creates the session handles for the query and sends the query to the compiler.
- ***Hive Compiler -***
  - Hive compiler parses the query.
  - It performs semantic analysis and type-checking on the different query blocks and query expressions by using the metadata stored in metastore and generates an execution plan.
  - The execution plan created by the compiler is the DAG(Directed Acyclic Graph), where each stage is a map/reduce job, operation on HDFS, a metadata operation.
- ***Optimizer -***
  - Optimizer performs the transformation operations on the execution plan and splits the task to improve efficiency and scalability.

Hive - Architecture - Hive Services

- ***Execution Engine-***
  - The Execution engine, after the compilation and optimization steps, executes the execution plan created by the compiler in order of their dependencies using Hadoop.
- ***Metastore -***
  - Metastore is a central repository that stores the metadata information about the structure of tables and partitions, including column and column type information.
  - It also stores information of serializer and deserializer, required for the read/write operation, and HDFS files where data is stored. This metastore is generally a relational database.
  - Metastore provides a Thrift interface for querying and manipulating Hive metadata.
  - We can configure metastore in any of the ***two*** modes:
    - Remote: In remote mode, metastore is a Thrift service and is useful for non-Java applications.
    - Embedded: In embedded mode, the client can directly interact with the metastore using JDBC.

Hive - Architecture - Hive Services

- ***HCatalog -***
  - HCatalog is the table and storage management layer for Hadoop.
  - It enables users with different data processing tools such as Pig, MapReduce, etc. to easily read and write data on the grid.
  - It is built on the top of Hive metastore and exposes the tabular data of Hive metastore to other data processing tools.
- ***WebHCat -***
  - WebHCat is the REST API for HCatalog.
  - It is an HTTP interface to perform Hive metadata operations.
  - It provides a service to the user for running Hadoop MapReduce (or YARN), Pig, Hive jobs.

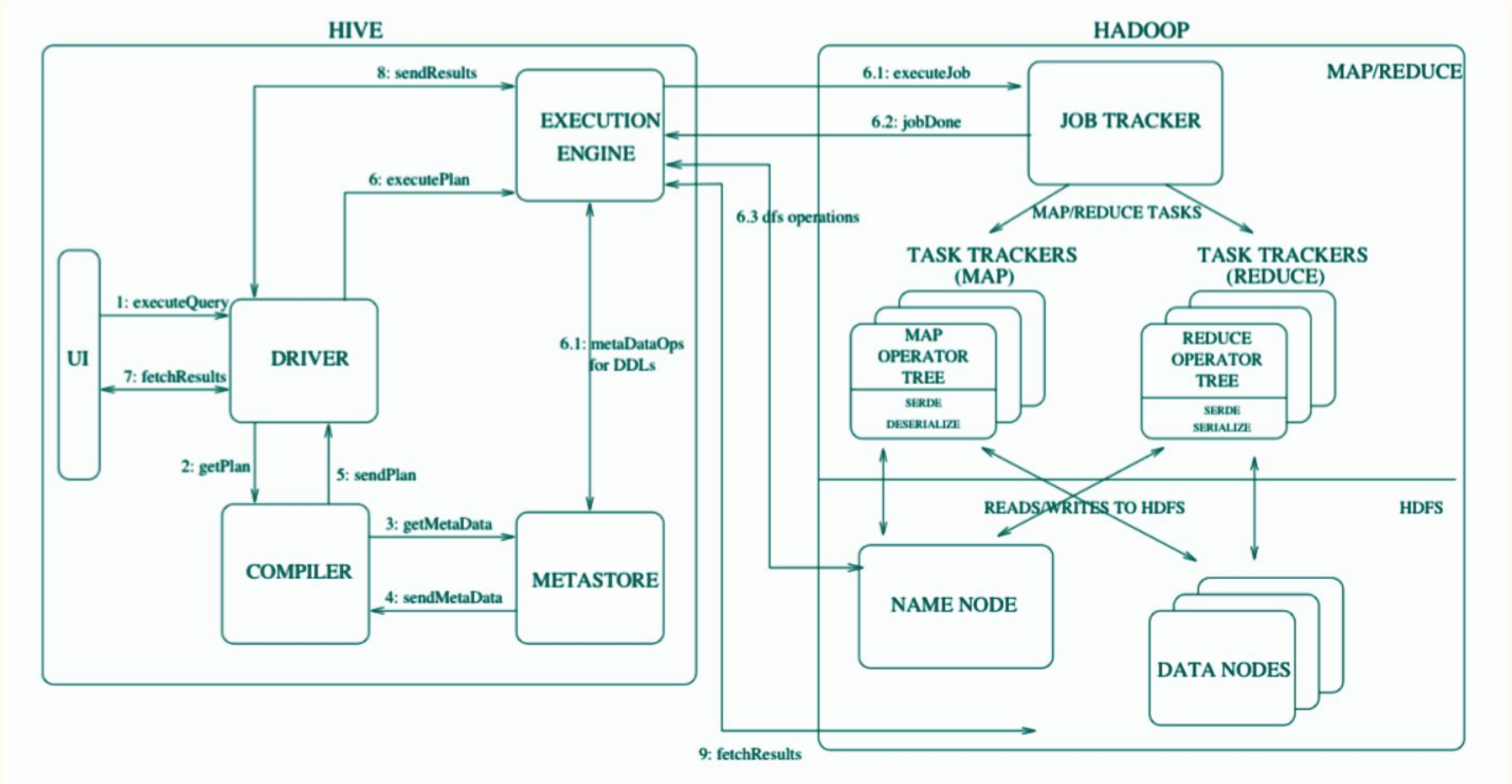
Hive - Architecture - Processing & Resource Management

- Hive internally uses a MapReduce framework as a defacto engine for executing the queries.
- MapReduce is a software framework for writing those applications that process a massive amount of data in parallel on the large clusters of commodity hardware.
- MapReduce job works by splitting data into chunks, which are processed by map-reduce tasks.

Hive - Architecture - Distributed Storage

- Hive is built on top of Hadoop, so it uses the underlying Hadoop Distributed File System for the distributed storage.

Hive - Execution of Job between Hive and Hadoop



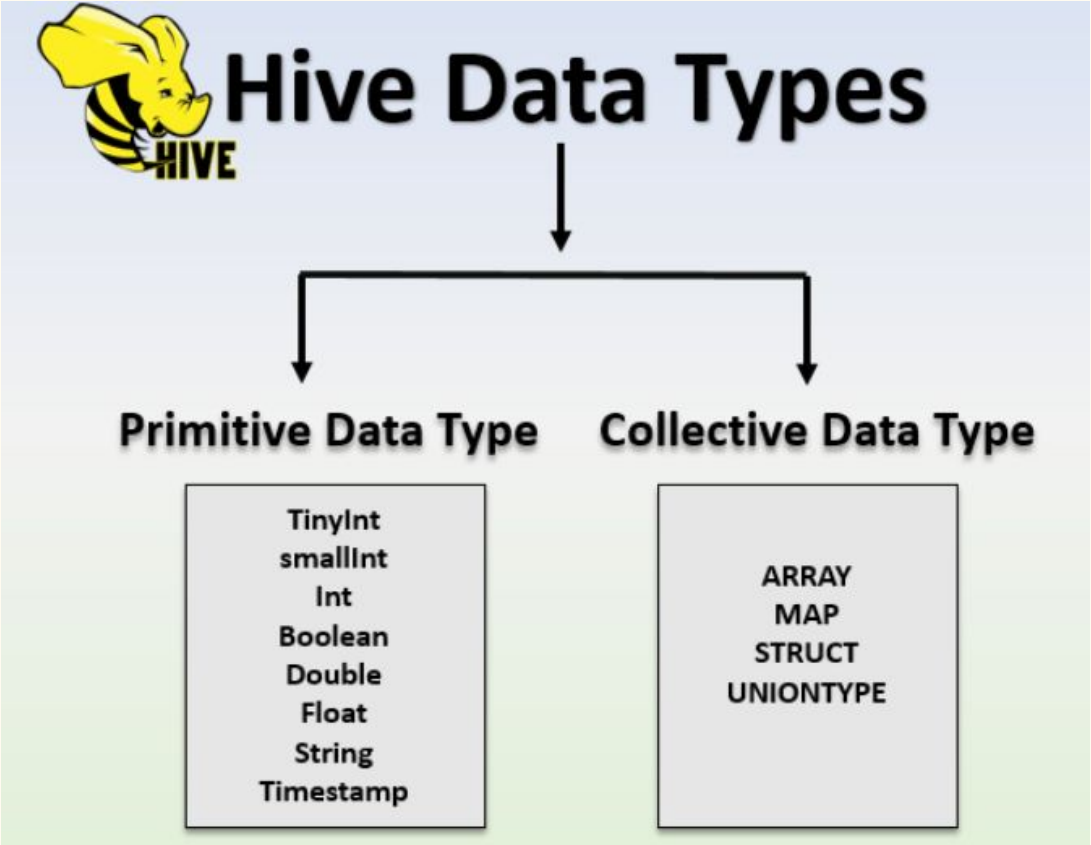
Comparison : Hive vs Traditional Databases

Hive	Traditional Databases
Schema on READ – it does not verify the schema while it load the data	Schema on WRITE – table schema is enforced at data load time i.e if the data being loaded does not conformed on schema in that case it will rejected
It’s very easily scalable at low cost	Not much Scalable, costly scale up.
It’s based on Hadoop notation that is Write once and read many times	In traditional database we can read and write many time
Record level updates is not possible in Hive	Record level updates, insertions and deletes, transactions and indexes are possible
OLTP (On-line Transaction Processing) is not yet supported in Hive but it’s supported OLAP (On-line Analytical Processing)	Both OLTP (On-line Transaction Processing) and OLAP (On-line Analytical Processing) are supported in RDBMS

Hive - Operating Modes

- Hive can operate in two modes depending on the size of data nodes in Hadoop.
  - ***Local Mode (when to use)***
    - If the Hadoop installed under pseudo mode with having one data node we use Hive in this mode
    - If the data size is smaller in term of limited to single local machine, we can use this mode
    - Processing will be very fast on smaller data sets present in the local machine
  - ***Map Reduce (when to use)***
    - If Hadoop is having multiple data nodes and data is distributed across different node we use Hive in this mode
    - It will perform on large amount of data sets and query going to execute in parallel way
    - Processing of large data sets with better performance can be achieved through this mode

Hive - HiveQL Data Types



Hive - Data Types - Primitive

- Numeric

Data type	Size
TINYINT – 1 byte signed integer	-128 to 127
SMALLINT – ‘2 byte signed integer	-32, 768 to 32, 767
INT – 4 byte signed integer	–2,147,483,648 to 2,147,483,647
BIGINT – 8 byte signed integer’	9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
FLOAT – ‘Single precision floating point	Single Precision
DOUBLE – Double precision floating point	Double Precision
DECIMAL – Precise decimal type based on Java BigDecimal Object	Big Decimal

Hive - Data Types - Primitive

- Date/Time Data Type -

```
TIMESTAMP - 'YYYY-MM-DD HH:MM:SS.fffffffff   = 9 decimal place precision
```

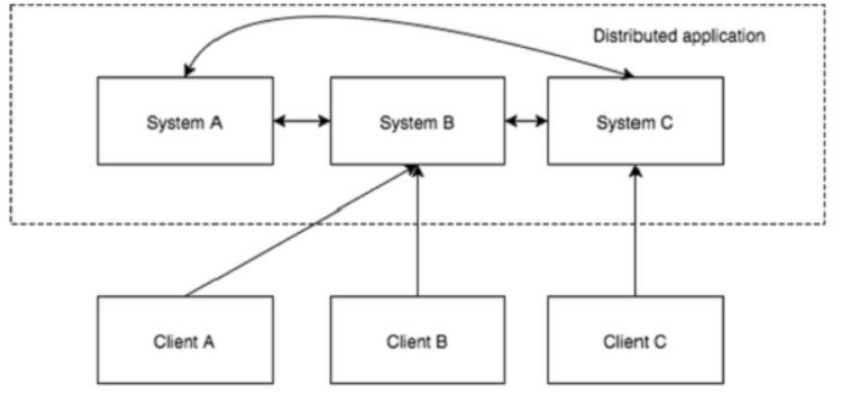
- String Data Type -
  - String - sequence of characters enclosed in ‘ ’ or “ ”
  - Varchar - max length 65355
  - Char - 1
- Boolean - True False

Hive - Data Types - Collective

- Array - An array is a collection of fields all of the same data type indexed by an integer.
  - `ARRAY<TINYINT>`
- Map - The map is a Collection of Key, Value Pairs where the Key is a Primitive Type and the Value can be anything. The chosen data types for the keys and values must remain the same per map.
  - `MAP<STRING,INT>`
- Struct - It is a nested complex data structure.
  - `STRUCT<first : SMALLINT, second : FLOAT, third : STRING>`
- Union - It is a Complex Data Type that can hold One of its Possible Data Types at Once.
  - `UNIONTYPE<INT,FLOAT,STRING>`

Distributed Applications

- A distributed application is an application which can run on multiple systems in a network.
- It runs simultaneously by coordinating themselves to complete a certain task.
- A distributed application has two parts, *Server and Client application.*
- Server applications are actually distributed and have a common interface so that clients can connect to any server in the cluster and get the same result.
- Client applications are the tools to interact with a distributed application.



Distributed Applications - Benefits

- Reliability − Failure of a single or a few systems does not make the whole system to fail.
- Scalability − Performance can be increased as and when needed by adding more machines with minor change in the configuration of the application with no downtime.
- Transparency − Hides the complexity of the system and shows itself as a single entity / application.

Distributed Applications - Challenges

- ***Race condition***
  - Two or more machines trying to perform a particular task, which actually needs to be done only by a single machine at any given time.
  - For example, shared resources should only be modified by a single machine at any given time. (resolved with ZooKeeper's serialization feature.)
- ***Deadlock***
  - Two or more operations waiting for each other to complete indefinitely. (Synchronization in Zookeeper helps resolve deadlocks.)
- ***Inconsistency***
  - Partial failure of data.

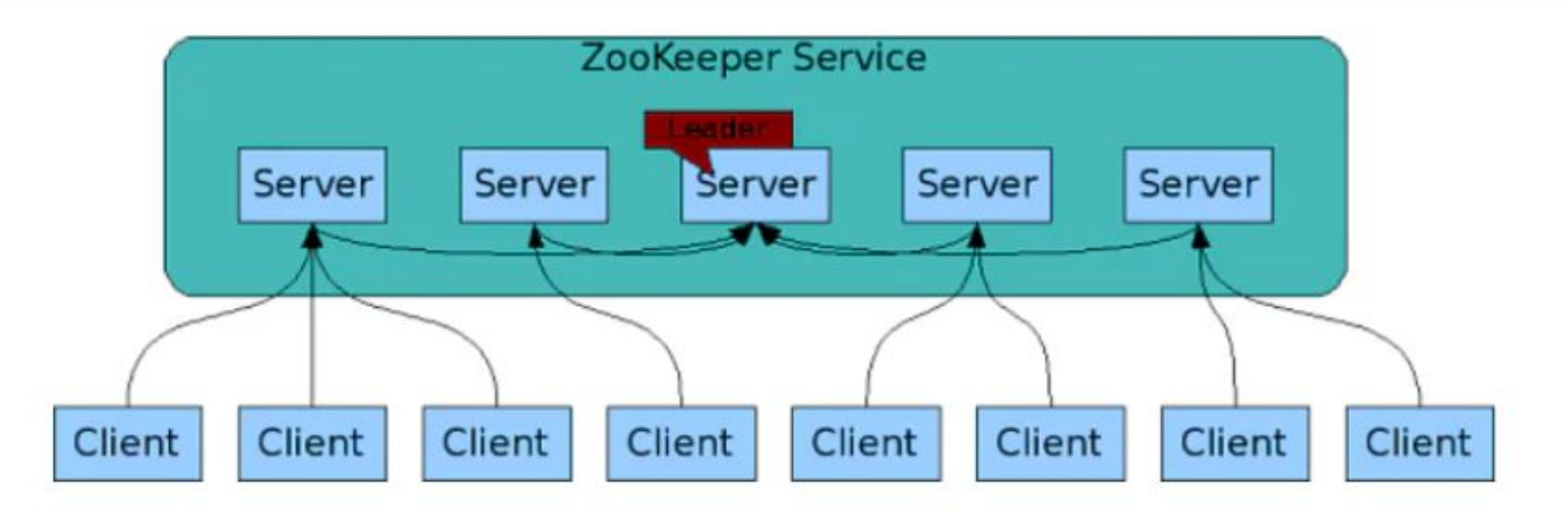
Zookeeper , how it helps in monitoring a cluster

- Zookeeper is a distributed, open-source coordination service for distributed applications.
- It exposes a simple set of primitives to implement higher-level services for synchronization, configuration maintenance, and group and naming.
- ZooKeeper is widely used in distributed systems such as Hadoop, Kafka, and HBase, and it has become an essential component of many distributed applications.

Zookeeper - Services

- ***Naming service***
  - Identifying the nodes in a cluster by name. It is similar to DNS, but for nodes.
- ***Configuration management***
  - Latest and up-to-date configuration information of the system for a joining node.
- ***Cluster management***
  - Joining / leaving of a node in a cluster and node status at real time.
- ***Leader election***
  - Electing a node as leader for coordination purpose.
- ***Locking and synchronization service***
  - Locking the data while modifying it. This mechanism helps you in automatic fail recovery while connecting other distributed applications like Apache HBase.
- ***Highly reliable data registry***
  - Availability of data even when one or a few nodes are down.

Zookeeper - Architecture



Zookeeper - Architecture - Working

- Hadoop Zookeeper Architecture is a distributed application that follows a simple client-server model,
- where clients are the nodes that consume the service and servers are the nodes that provide the service.
- Multiple server nodes are collectively called a ZooKeeper file.
- A zookeeper client uses at least one server at a given time.
- The master node is dynamically selected based on consensus within the ensemble, so the Zookeeper file is usually an odd number, so that's where the majority of votes are.
- If the master node fails, another master node is instantly selected and takes over from the previous master node.
- In addition to masters and slaves, there are also observers in Zookeeper.
- Observers were invited to address the scaling issue.
- The addition of slaves affected writing performance because the voting process was expensive.
- Observers are, therefore, slaves who do not participate in voting but have similar duties to other slaves.

END (Part 1)