Module 2

Syllabus:

Introduction to Hadoop (T1): *Introduction, Hadoop and its Ecosystem, Hadoop Distributed File System, MapReduce Framework and Programming Model, Hadoop Yarn, Hadoop Ecosystem Tools.*

Hadoop Distributed File System Basics (T2): *HDFS Design Features, Components, HDFS User Commands.*

Essential Hadoop Tools (T2): *Using Apache Pig, Hive, Sqoop, Flume, Oozie, HBase..*

Introduction to Hadoop:

Introduction:

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

A programming model is centralized computing of data in which the data is transferred from multiple distributed data sources to a central server. Analyzing, reporting, visualizing, business-intelligence tasks compute centrally. Data are inputs to the central server.

An enterprise collects and analyzes data at the enterprise level.

Big Data Store Model:

Model for Big Data store is as follows: Data store in file system consisting of data blocks (physical division of data). The data blocks are distributed across multiple nodes. Data nodes are at the racks of a cluster. Racks are scalable. A Rack has multiple data nodes (data servers), and each cluster is arranged in a number of racks.

Data Store model of files in data nodes in racks in the clusters Hadoop system uses the data store model in which storage is at clusters, racks, data nodes and data blocks. Data blocks replicate at the DataNodes such that a failure of link leads to access of the data block from the other nodes replicated at the same or other racks

Big Data Programming Model:

Big Data programming model is that application in which application jobs and tasks (or sub-tasks) is scheduled on the same servers which store the data for processing.

Job means running an assignment of a set of instructions for processing. For example, processing the queries in an application and sending the result back to the application is a job. Other example is instructions for sorting the examination performance data is a job.

Hadoop and its Ecosystem:

Apache initiated the project for developing storage and processing framework for Big Data storage and processing. Doug Cutting and Machael J. Cafarelle the creators named that framework as Hadoop. Cutting's son was fascinated by a stuffed toy elephant, named Hadoop, and this is how the name Hadoop was derived.

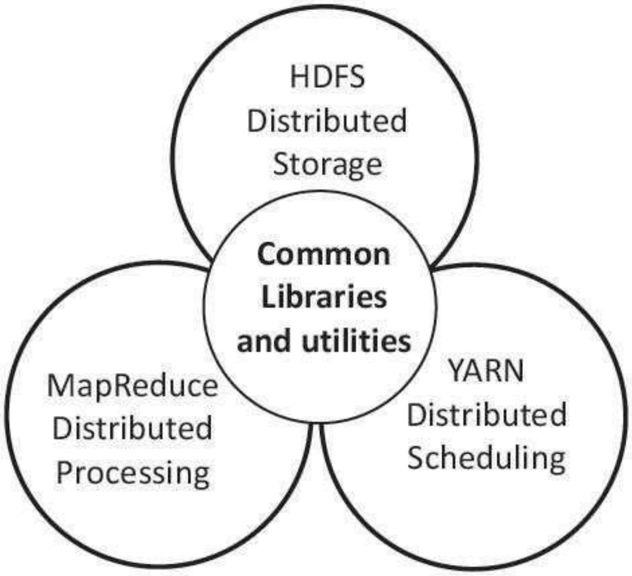
The project consisted of two components, one of them is for data store in blocks in the clusters and the other is computations at each individual cluster in parallel with another.

Hadoop components are written in Java with part of native code in C. The command line utilities are written in shell scripts.

Infrastructure consists of cloud for clusters. A cluster consists of sets of computers or PCs. The Hadoop platform provides a low cost Big Data platform, which is open source and uses cloud services. Tera Bytes of data processing takes just few minutes. Hadoop enables distributed processing of large datasets (above 10 million bytes) across clusters of computers using a programming model called MapReduce. The system characteristics are scalable, self-manageable, self-healing and distributed file system.

Hadoop core components:

The following diagram shows the core components of the Apache Software Foundation's Hadoop framework.



The Hadoop core components of the framework are:

1. Hadoop Common - The common module contains the libraries and utilities that are required by the other modules of Hadoop. For example, Hadoop common provides various components and interfaces for distributed file system and general input/output. This includes serialization. Java RPC (Remote Procedure Call) and file based data structures.
2. Hadoop Distributed File System (HDFS) - A Java-based distributed file system which can store all kinds of data on the disks at the clusters.
3. MapReduce v1 - Software programming model in Hadoop I using Mapper and Reducer. The v1 processes large sets of data in parallel and in batches.
4. YARN-Software for managing resources for computing. The user application tasks or sub-tasks run in parallel at the Hadoop, uses scheduling and handles the requests for the resources in distributed running of the tasks.
5. MapReduce v2 – Hadoop 2 YARN based system for parallel processing of large datasets and distributed processing of the application tasks.

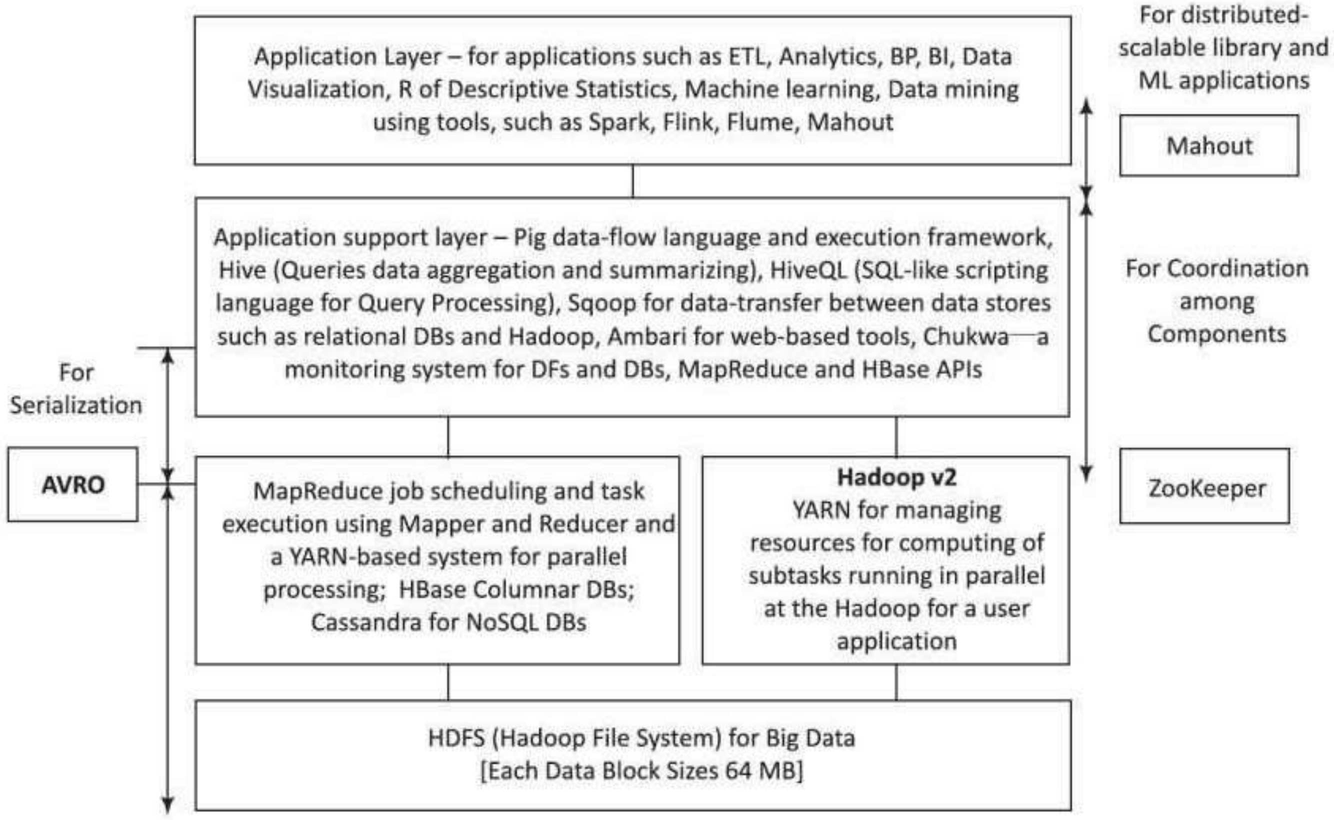
Features of Hadoop:

Hadoop features are as follows:

1. Fault-efficient scalable, flexible and modular design which uses simple and modular programming model. The system provides servers at high scalability. The system is scalable by adding new nodes to handle larger data. Hadoop proves very helpful in storing, managing, processing and analysing Big Data. Modular functions make the system flexible. One can add or replace components at ease. Modularity allows replacing its components for a different software tool.
2. Robust design of HDFS: Execution of Big Data applications continue even when an individual server or cluster fails. This is because of Hadoop provisions for backup (due to replications at least three times for each data block) and a data recovery mechanism. HDFS thus has high reliability.
3. Store and process Big Data: Processes Big Data of 3V characteristics.
4. Distributed clusters computing model with data locality: Processes Big Data at high speed as the application tasks and sub-tasks submit to the DataNodes. One can achieve more computing power by increasing the number of computing nodes. The processing splits across multiple DataNodes (servers), and thus fast processing and aggregated results.
5. Hardware fault-tolerant: A fault does not affect data and application processing. If a node goes down, the other nodes take care of the residue. This is due to multiple copies of all data blocks which replicate automatically. Default is three copies of data blocks.
6. Open-source framework: Open source access and cloud services enable large data store. Hadoop uses a cluster of multiple inexpensive servers or the cloud.

7. Java and Linux based: Hadoop uses Java interfaces. Hadoop base is Linux but has its own set of shell commands support.

Hadoop Ecosystem Components:


Application Layer – for applications such as ETL, Analytics, BP, BI, Data Visualization, R of Descriptive Statistics, Machine learning, Data mining using tools, such as Spark, Flink, Flume, Mahout
For distributed-scalable library and ML applications
Mahout
Application support layer – Pig data-flow language and execution framework, Hive (Queries data aggregation and summarizing), HiveQL (SQL-like scripting language for Query Processing), Sqoop for data-transfer between data stores such as relational DBs and Hadoop, Ambari for web-based tools, Chukwa—a monitoring system for DFs and DBs, MapReduce and HBase APIs
For Coordination among Components
For Serialization
AVRO
MapReduce job scheduling and task execution using Mapper and Reducer and a YARN-based system for parallel processing; HBase Columnar DBs; Cassandra for NoSQL DBs
Hadoop v2
YARN for managing resources for computing of subtasks running in parallel at the Hadoop for a user application
ZooKeeper
HDFS (Hadoop File System) for Big Data
[Each Data Block Sizes 64 MB]

Hadoop main components and ecosystem components

The four layers are:

1. Distributed storage layer
2. Resource manager layer for job or application sub tasks scheduling and execution.
3. Processing framework layer, consisting of Mapper and Reducer for the Map reduce process flow
4. APIs at the application support layer.

Hadoop Distributed File System:

HDFS Data storage:

- Hadoop data store concept implies storing the data at a number of clusters. Each cluster has a number of data stores, called racks.
- Each rack stores a number of DataNodes. Each DataNode has a large number of data blocks. The racks distribute across a cluster. The nodes have processing and storage capabilities.
- The nodes have the data in data blocks to run the application tasks. The data blocks replicate by

default at least on three DataNodes in same or remote nodes.

- Data at the stores enable running the distributed applications including analytics, data mining, OLAP using the clusters. A file, containing the data divides into data blocks.
- A data block default size is 64 MBs (HDFS division of files concept is similar to Linux or virtual memory page in Intel x86 and Pentium processors where the block size is fixed and is of 4 KB).

Hadoop HDFS features are as follows:

(i) Create, append, delete, rename and attribute modification functions

(ii) Content of individual file cannot be modified or replaced but appended with new data at the end of the file.

(iii) Write once but read many times during usages and processing

(iv) Average file size can be more than 500 MB.

Example:

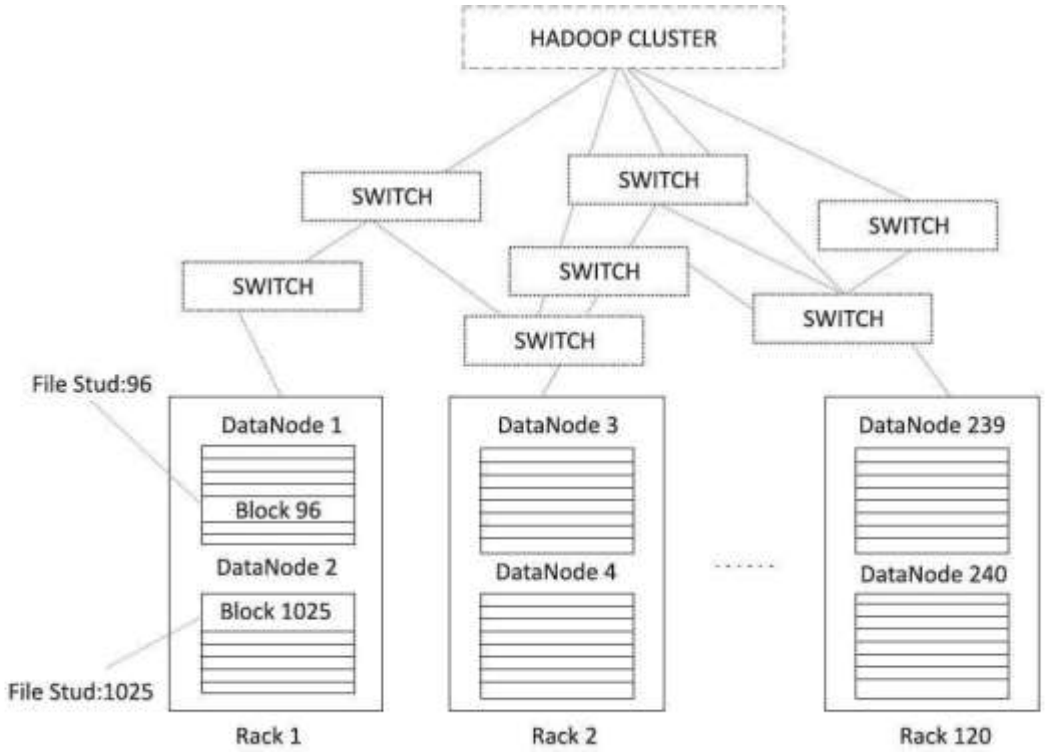
Consider a data storage for University students. Each student data, stuData which is in a file of size less than 64 MB (1MB=2^{20}B) A data block stores the full file data for a student of stuData_idN, where N=1 to 500.

(i) How the files of each student will be distributed at a Hadoop cluster? How many student data can be stored at one cluster? Assume that each rack has two DataNodes for processing each of 64 GB (1 GB=2^{30}B) memory. Assume that cluster consists of 120 racks, and thus 240 DataNodes.

(ii) What is the total memory capacity of the cluster in TB ((1 TB=2B) and DataNodes in each rack?

(iii) Show the distributed blocks for students with ID=9 and 1025. Assume default replication in the DataNodes=3

(iv) What shall be the changes when a stuData file size ≤ 128 MB?



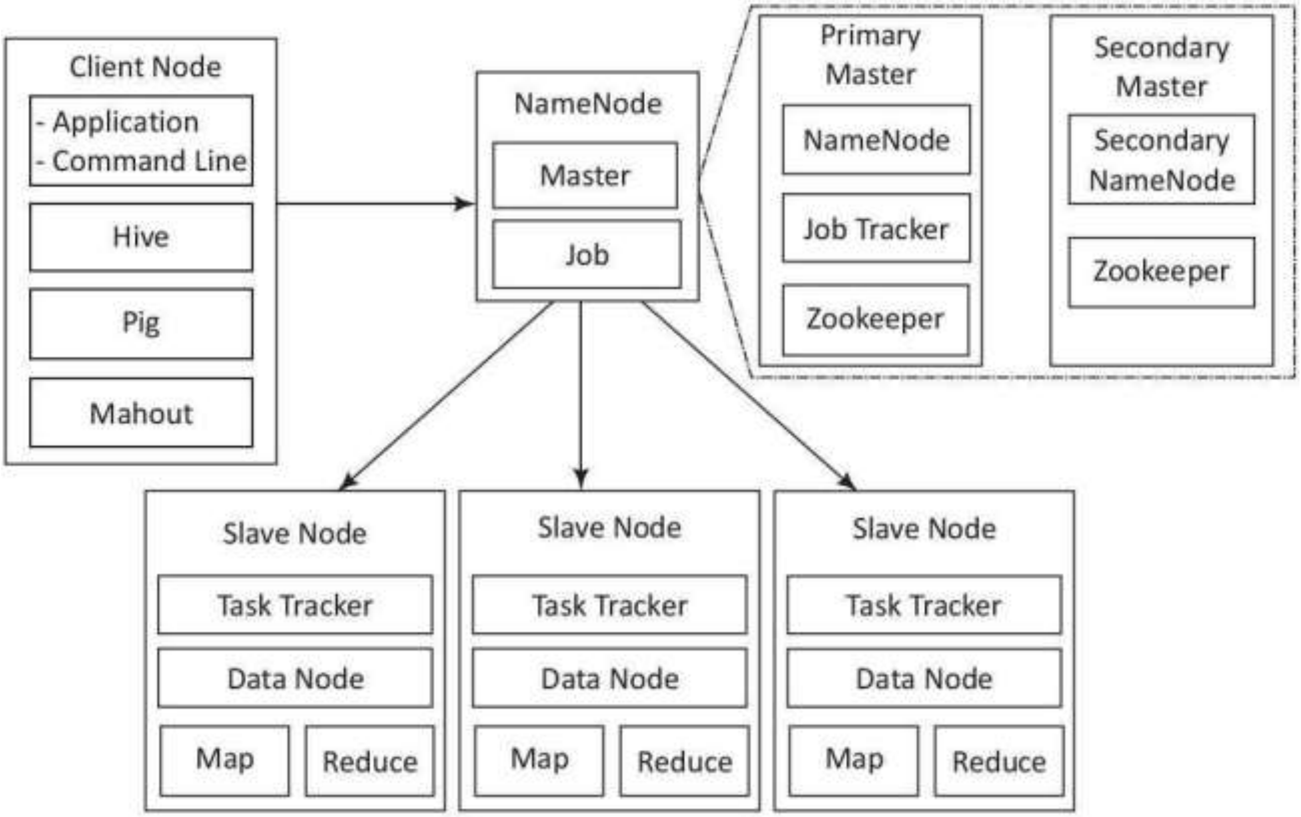
A Hadoop cluster example and the replication of data blocks in racks for two students of IDs 96 and 1025

Hadoop Physical Organization:

Few nodes in a Hadoop cluster act as NameNodes. These nodes are termed as MasterNodes or simply masters. The masters have a different configuration supporting high DRAM and processing power. The masters have much less local storage. Majority of the nodes in Hadoop cluster act as DataNodes and Task Trackers. These nodes are referred to as slave nodes or slaves. The slaves have lots of disk storage and moderate amounts of processing capabilities and DRAM. Slaves are responsible to store the dat and process the computation tasks submitted by the clients.

The following Figure shows the client, master NameNode, primary and secondary MasterNodes and slave nodes in the Hadoop physical architecture.

Clients as the users run the application with the help of Hadoop ecosystem projects. For example, Hive, Mahout and Pig are the ecosystem's projects. They are not required to be present at the Hadoop cluster. A single MasterNode provides HDFS, MapReduce and Hbase using threads in small to medium sized clusters. When the cluster size is large, multiple servers are used, such as to balance the load. The secondary NameNode provides NameNode management services and Zookeeper is used by HBase for metadata storage.



The client, master NameNode, master nodes and slave nodes

The MasterNode fundamentally plays the role of a coordinator. The MasterNode receives client connections, maintains the description of the global file system namespace, and the allocation of file blocks. It also monitors the state of the system in order to detect any failure. The Masters consists of three components NameNode, Secondary NameNode and JobTracker. The NameNode stores all the fil system related information such as:

- The file section is stored in which part of the cluster
- Last access time for the files
- User permissions like which user has access to the file.

Secondary NameNode is an alternate for NameNode. Secondary node keeps a copy of NameNode meta data.

MapReduce Framework and Programming Model:

Mapper means software for doing the assigned task after organizing the data blocks imported using the keys. Reducer means software for reducing the mapped data by using the aggregation, query or user specified function.

Hadoop Mapreduce framework:

MapReduce provides two important functions. The distribution of job based on client application task or users query to various nodes within a cluster is one function. The second function is organizing and reducing the results from each node into a cohesive response to the application or answer to the query.

The processing tasks are submitted to the Hadoop. The Hadoop framework in turns manages the task of issuing jobs, job completion, and copying data around the cluster between the DataNodes with the help of JobTracker.

A client node submits a request of an application to the JobTracker. A JobTracker is a Hadoop daemon (background program).

The following are the steps on the request to MapReduce:

(i) estimate the need of resources for processing that request

(ii) analyze the states of the slave nodes

(iii) place the mapping tasks in queue

(iv) monitor the progress of task, and on the failure, restart the task on slots of time available.

The job execution is controlled by two types of processes in MapReduce:

1. The Mapper deploys map tasks on the slots. Map tasks assign to those nodes where the data for the application is stored. The Reducer output transfers to the client node after the data serialization using AVRO.

2. The Hadoop system sends the Map and Reduce jobs to the appropriate servers in the cluster. The Hadoop framework in turns manages the task of issuing jobs, job completion and copying data around the cluster between the slave nodes. Finally, the cluster collects and reduces the data to obtain the result and sends it back to the Hadoop server after completion of the given tasks.

MapReduce Programming model:

MapReduce program can be written in any language including JAVA, C++ PIPES or Python. Map function of MapReduce program do mapping to compute the data and convert the data into other data sets (distributed in HDFS). After the Mapper computations finish, the Reducer function collects the result of map and generates the final output result. MapReduce program can be applied to any type of data, i.e., structured or unstructured stored in HDFS.

- The input data is in the form of file or directory and is stored in the HDFS.
- The MapReduce program performs two jobs on this input data, the Map job and the Reduce job. They are also termed as two phases Map phase and Reduce phase.

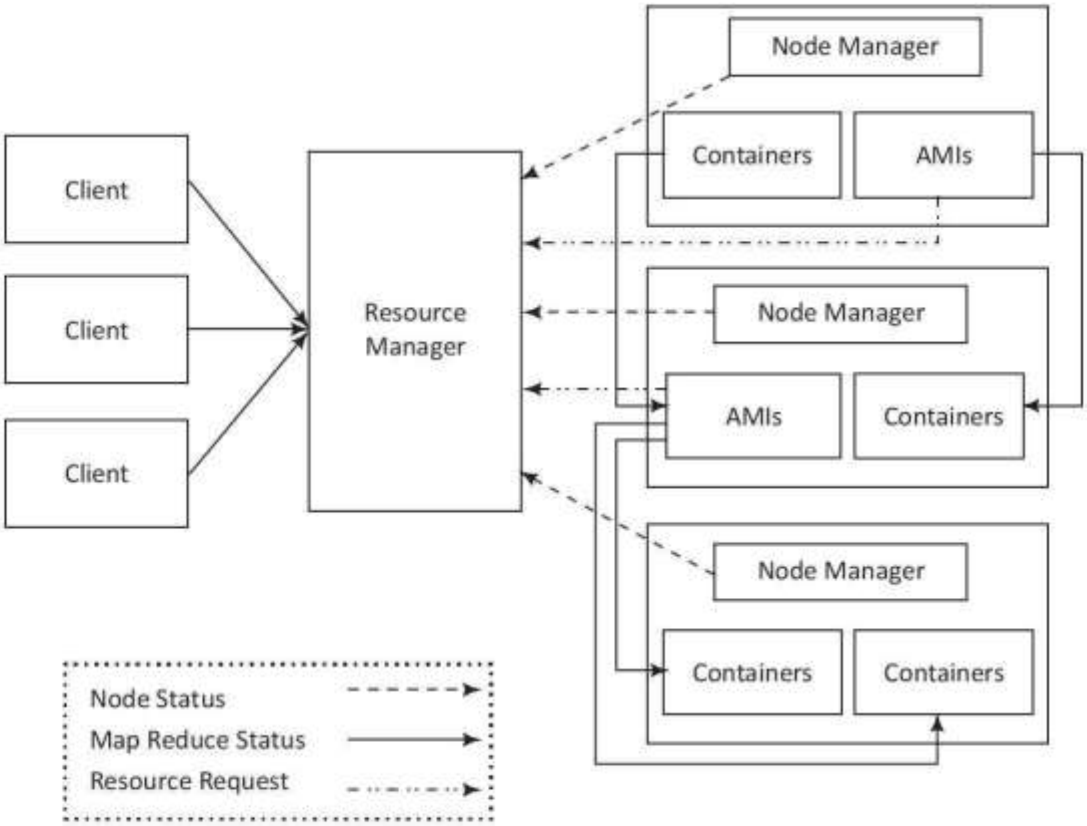
- The map job takes a set of data and converts it into another set of data. The individual elements are broken down into tuples (key/value pairs) in the resultant set of data.
- The reduce job takes the output from a map as input and combines the data tuples into a smaller set of tuples.
- Map and reduce jobs run in isolation from one another. As the sequence of the name MapReduce implies, the reduce job is always performed after the map job.

Hadoop YARN:

YARN is a resource management platform. It manages computer resource.YARN manages the schedules for running of the sub-tasks.

Hadoop 2 Execution model:

Following shows the YARN-based execution model. The figure shows the YARN components Client, Resource Manager (RM), Node Manager (NM), Application Master (AM) and Containers. And also illustrates YARN components namely, Client, Resource Manager (RM), Node Manager (RM), Application Master (AM) and Containers.



YARN based execution model

List of actions of YARN resource allocation and scheduling functions is as follows:

- A MasterNode has two components: (i) Job History Server and (ii) Resource Manager(RM).
- A Client Node submits the request of an application to the RM. The RM is the master. One RM exists per cluster. The RM keeps information of all the slave NMs.
- Multiple NMs are at a cluster. An NM creates an AM instance (AMI) and starts up. The AMI initializes itself and registers with the RM. Multiple AMIs can be created in an AM.
- The AMI performs role of an Application Manager (AppIM), that estimates the resources requirement for running an application program or sub-task. The ApplMs send their requests for the necessary resources to the RM.
- NM is a slave of the infrastructure. It signals whenever it initializes. All active NMs send the controlling signal periodically to the RM signaling their presence.
- Each NM assigns a container(s) for each AMI. The container(s) assigned at an instance may be at same NM or another NM. ApplM uses just a fraction of the resources available. The AppIM at an instance uses the assigned container(s) for running the application sub-task.
- RM allots the resources to AM, and thus to ApplMs for using assigned containers on the sam or other NM for running the application subtasks in parallel

Hadoop Ecosystem Tools:

Zookeeper:

Zookeeper in Hadoop behaves as a centralized repository where distributed applications can write data at a node called JournalNode and read the data out of it. Zookeeper uses synchronization, serialization and coordination activities. It enables functioning of a distributed system as a single function. ZooKeeper's main coordination services are:

1 Name service - A name service maps a name to the information associated with that name. For example, DNS service is a name service that maps a domain name to an IP address. Similarly, name keeps a track of servers or services those are up and running, and looks up their status by name in name service.

2 Concurrency control - Concurrent access to a shared resource may cause inconsistency of the resource. A concurrency control algorithm accesses shared resource in the distributed system and controls concurrency.

3 Configuration management - A requirement of a distributed system is a central configuration manager. A new joining node can pick up the up-to-date centralized configuration from the ZooKeeper coordination service as soon as the node joins the system. 4 Failure Distributed systems are susceptible to the problem of node failures. This requires implementing an automatic recovering strategy by selecting some alternate node for processing

Oozie:

Apache Oozie is an open-source project of Apache that schedules Hadoop jobs. An efficient process for job handling is required. Analysis of Big Data requires creation of multiple jobs and sub-tasks in a process. Oozie design provisions the scalable processing of multiple jobs. Thus, Oozie provides a way to package and bundle multiple coordinator and workflow jobs, and manage the lifecycle of those jobs. Oozie workflow jobs are represented as Directed Acrylic Graphs (DAGs), specifying a sequence of Oozie coordinator jobs are recurrent Oozie workflow jobs that are triggered by time and data.

Oozie provisions for the following:

1. Integrates multiple jobs in a sequential manner
2. Stores and supports Hadoop jobs for MapReduce, Hive, Pig, and Sqoop
3. Runs workflow jobs based on time and data triggers
4. Manages batch coordinator for the applications
5. Manages the timely execution of tens of elementary jobs lying in thousands of workflows in a Hadoop cluster.

Sqoop:

The loading of data into Hadoop clusters becomes an important task during data analytics. Apache Sqoop is a tool that is built for loading efficiently the voluminous amount of data between Hadoop and external data. Sqoop initially parses the arguments passed in the command line and prepares the map task. The map task initializes multiple Mappers depending on the number supplied by the user in the command line. Each map task will be assigned with part of data to be imported based on key defined in the command line. Sqoop distributes the input data equally among the Mappers. Then each Mapper creates a connection with the database using JDBC and fetches the part of data assigned by Sqoop and writes it into HDFS/Hive/HBase as per the choice provided in the command line.

Sqoop provides the mechanism to import data from external Data Stores into HDFS. Sqoop relates to Hadoop eco-system components, such as Hive and HBase. Sqoop can extract data from Hadoop or other ecosystem components.

Sqoop provides command line interface to its users. Sqoop can also be accessed using Java APIs. The tool allows defining the schema of the data for import. Sqoop exploits MapReduce framework to import and export the data, and transfers for parallel processing of sub-tasks. Sqoop provisions for fault tolerance. Parallel transfer of data results in parallel results and fast data transfer.

Flume:

Apache Flume provides a distributed, reliable and available service. Flume efficiently collects, aggregates and transfers a large amount of streaming data into HDFS. Flume enables upload of large files into Hadoop clusters.

The features of flume include robustness and fault tolerance. Flume provides data transfer which is reliable and provides for recovery in case of failure. Flume is useful for transferring a large amount of data in applications related to logs of network traffic, sensor data, geo-location data, e-mails and social-media messages.

Apache Flume has the following four important components:

1. Sources which accept data from a server or an application.
2. Sinks which receive data and store it in HDFS repository or transmit the data to another source. Data units that are transferred over a channel from source to sink are called events.
3. Channels connect between sources and sink by queuing event data for transactions. The size of events data is usually 4 KB. The data source is considered to be a source of various set of events.

Sources listen for events and write events to a channel. Sinks basically write event data to a target and remove the event from the queue,

4. Agents run the sinks and sources in Flume. The interceptors drop the data or transfer data as it flows into the system.

Hadoop Distributed File System Basics 2

Hadoop Distributed File system design features

The Hadoop Distributed file system(HDFS) was designed **for Big Data processing**. Although capable of supporting many users simultaneously, HDFS is not designed as a true parallel file system. Rather, the design assumes a large file **write-once/read-many model.** HDFS rigorously restricts data writing to one user at a time. Bytes are always appended to the end of a stream, and **byte streams** are guaranteed to be **stored in the order written.** The design of HDFS is based on the design of **the**

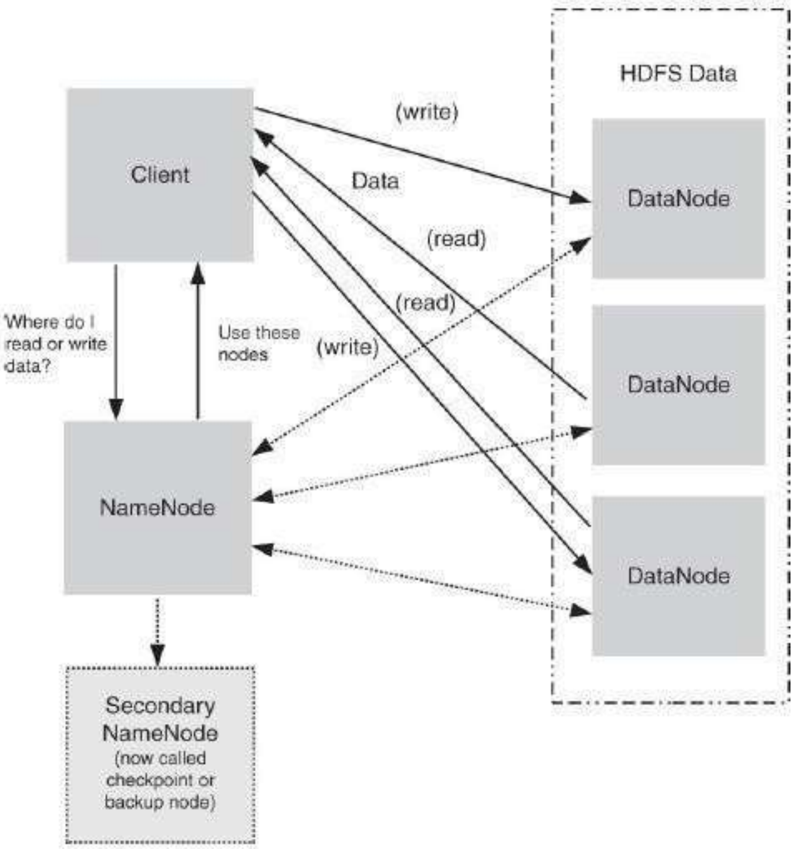
Google File System(GFS). HDFS is designed for data streaming where **large amounts** of data are **read from disk in bulk**. **The HDFS block size is typically 64MB or 128MB**. Thus, this approach is unsuitable for standard POSIX file system use.

Due to sequential nature of data, there is **no local caching mechanism**. The large block and file sizes makes it more efficient to reread data from HDFS than to try to cache the data. A principal design aspect of Hadoop MapReduce is the emphasis on moving the computation to the data rather than moving the data to the computation. In other high performance systems, a parallel file system will exist on hardware separate from computer hardware. Data is then moved to and from the computer components via high-speed interfaces to the parallel file system array. Finally, Hadoop clusters assume node failure will occur at some point. To deal with this situation, it has a redundant design that can tolerate system failure and still provide the data needed by the compute part of the program.

The following points are important aspects of HDFS:

- The **write-once/read-many design** is intended to facilitate streaming reads.
- Files may be appended, but **random seeks are not permitted**. There is **no caching** of data.
- Converged **data storage and processing** happen on the **same server nodes.**
- ―Moving computation is cheaper than moving data.‖
- A reliable file **system maintains multiple copies of data** across the cluster.
- Consequently, **failure of a single will not bring down the file system**.
- A specialized file system is used, which **is not designed for general use**.

HDFS components:



various system roles in an HDFS deployment

Reference: Douglas Eadline,**"Hadoop 2 Quick-Start Guide: Learn the Essentials of Big Data Computing in the Apache Hadoop 2 Ecosystem",** 1st Edition, Pearson Education, 2016

The design of HDFS is based on two types of nodes**: NameNode and multiple DataNodes.** In a basic design, **NameNode manages all the metadata** needed to store and **retrieve the actual data from the DataNode**s. No data is actually stored on the NameNode. The design is a **Master/Slave architecture** in which **master(NameNode)** manages the **file system namespace and regulates access to files by clients**. File system namespace operations such as opening, closing and renaming files and directories are all managed by the NameNode. The **NameNod**e also determines the **mapping of blocks to DataNodes** and **handles Data Node failures.**

The **slave(DataNodes) are responsible for serving read and write requests** from the file system to the clients. The **NameNode manages block creation, deletion and replication**. When a client writes data, it first communicates with the NameNode and requests to create a file. The NameNode determines how many blocks are needed and provides the client with the DataNodes that will store the data. As part of the storage process, the data blocks are replicated after they are written to the assigned node.

Depending on how many nodes are in the cluster, the NameNode will attempt to write replicas of the data blocks on nodes that are in other separate racks. If there is only one rack, then the replicated blocks are written to other servers in the same rack. After the Data Node acknowledges that the file block replication is complete, the client closes the file and informs the NameNode that the operation is complete. Note that the **NameNode does not write any data directly to the DataNodes**. It does, however, give the client a limited amount of time to complete the operation. **If it does not complete in the time period, the operation is cancelled.**

The client requests a file from the NameNode, which returns the best DataNodes from which to read the data. The client then access the data directly from the DataNodes. Thus, once the metadata has been delivered to the client, the NameNode steps back and lets the conversation between the client and the DataNodes proceed. While data transfer is progressing, the NameNode also monitors the DataNodes by listening for heartbeats sent from DataNodes. The **lack of a heartbeat signal indicates a node failure**. Hence the NameNode will route around the failed Data Node and begin re-replicating the now-missing blocks. The mappings b/w data blocks and physical DataNodes are not kept in persistent storage on the NameNode. The NameNode stores all metadata in memory.In almost all Hadoop deployments, there is a SecondaryNameNode(Checkpoint Node). It is not an active failover node and cannot replace the primary NameNode in case of it failure.

- Thus the various important roles in HDFS are:
    - ❖ HDFS uses a **master/slave model** designed for large file reading or streaming.
    - ❖ The **NameNode** is a **metadata server or “Data traffic cop”.**
    - ❖ HDFS provides a **single namespace** that is **managed by the NameNode**.
    - ❖ **Data is redundant**ly stored **on DataNodes** ; there is **no data on NameNode.**
    - ❖ SecondaryNameNode performs checkpoints of NameNode file system’s state but is not a failover node.

HDFS user commands:

- **List Files in HDFS**
    - ❖ To list the files in the root HDFS directory, enter the following command:

```
Syntax: $ hdfs dfs -ls /
Output:
            Found 2  items
```

```
drwxrwxrwx - yarn hadoop 0 2015-04-29 16:52 /app-logs
drwxr-xr-x - hdfs hdfs 0 2015-04-21 14:28 /apps
```

- ❖ To list files in your home directory, enter the following command:

  Syntax: $ hdfs dfs -ls

  Output:

```
Found 2  items
drwxr-xr-x - hdfs hdfs 0 2015-05-24 20:06 bin
drwxr-xr-x - hdfs hdfs 0 2015-04-29 16:52 examples
```

- **Make a Directory in HDFS**
  - ❖ To make a directory in HDFS, use the following command. As with the -ls command, when no path is supplied, the user's home directory is used

    Syntax: $ hdfs dfs -mkdir stuff
- **Copy Files to HDFS**
  - ❖ To copy a file from your current local directory into HDFS, use the following command. If a full path is not supplied, your home directory is assumed. In this case, the file test is placed in the directory stuff that was created previously.

    Syntax: $ hdfs dfs -put test stuff
  - ❖ The file transfer can be confirmed by using the -ls command:Syntax: $ hdfs dfs -ls stuff

    Output:

```
Found 1 items
-rw-r--r-- 2 hdfs hdfs 12857 2015-05-29 13:12 stuff/test
```

- **Copy Files from HDFS**
  - ❖ Files can be copied back to your local file system using the following command.
  - ❖ In this case, the file we copied into HDFS, test, will be copied back to the current local directory with the name test-local.

    Syntax: $ hdfs dfs -get stuff/test test-local
- **Copy Files within HDFS**
  - ❖ The following command will copy a file in HDFS:

    Syntax: $ hdfs dfs -cp stuff/test test.hdfs
- **Delete a File within HDFS**
  - ❖ The following command will delete the HDFS file test.

    Syntax: $ hdfs dfs -rm test.hd

Essential Hadoop Tools:

Using Apache pig:

Apache Pig is a high-level language that enables programmers to write complex Map Reduce transformations using a simple scripting language.

Pig’s simple SQL-like scripting language is called **Pig Latin**, and appeals to developers already familiar with scripting languages and SQL.

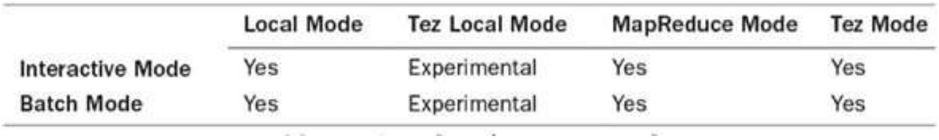
Pig Latin (the actual language) defines a set of transformations on a data set such as aggregate, join, and sort.

Pig is often used to extract, transform, and load (ETL) data pipelines, quick research on raw data.

Apache Pig has several usage modes. The first is a **local mode** in which all processing is done on the local machine. The **non-local (cluster) modes** are Map Reduce and Tez.

These modes execute the job on the cluster using either the Map Reduce engine or the optimized Tez engine.

There are also interactive and batch modes available; they enable Pig applications to be developed locally in interactive modes, using small amounts of data, and then run at scale on the cluster in a production mode. The modes are in below fig.



	Local Mode	Tez Local Mode	MapReduce Mode	Tez Mode
Interactive Mode	Yes	Experimental	Yes	Yes
Batch Mode	Yes	Experimental	Yes	Yes

Apache Hive:

Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, ad hoc queries, and the analysis of large data sets using a SQL-like language called HiveQL.

Hive is considered the de facto standard for interactive SQL queries over petabytes of data using Hadoop.

Some essential features:

Tools to enable easy data extraction, transformation, and loading (ETL) A mechanism to impose structure on a variety of data formats

Access to files stored either directly in HDFS or in other data storage systems such as HBase Query execution via MapReduce and Tez (optimized MapReduce) Hive is also installed as part of the Hortonworks HDP Sandbox. To work in Hive with Hadoop, user with access to HDFS can run the Hive queries.

Simply enter the hive command. If Hive start correctly,it get a hive> prompt.

```
$ hive
(some messages may show up here)
hive>
```

Hive command to create and drop the table. That Hive commands must end with a semicolon (;).

```
hive> CREATE TABLE pokes (foo INT, bar STRING);
OK
Time taken: 1.705 seconds
```

To see the table is created,

```
hive> SHOW TABLES;
OK
pokes
Time taken: 0.174 seconds, Fetched: 1 row(s)
```

To drop the table,

```
hive> DROP TABLE pokes;
OK
Time taken: 4.038 seconds
```

Apache Sqoop:

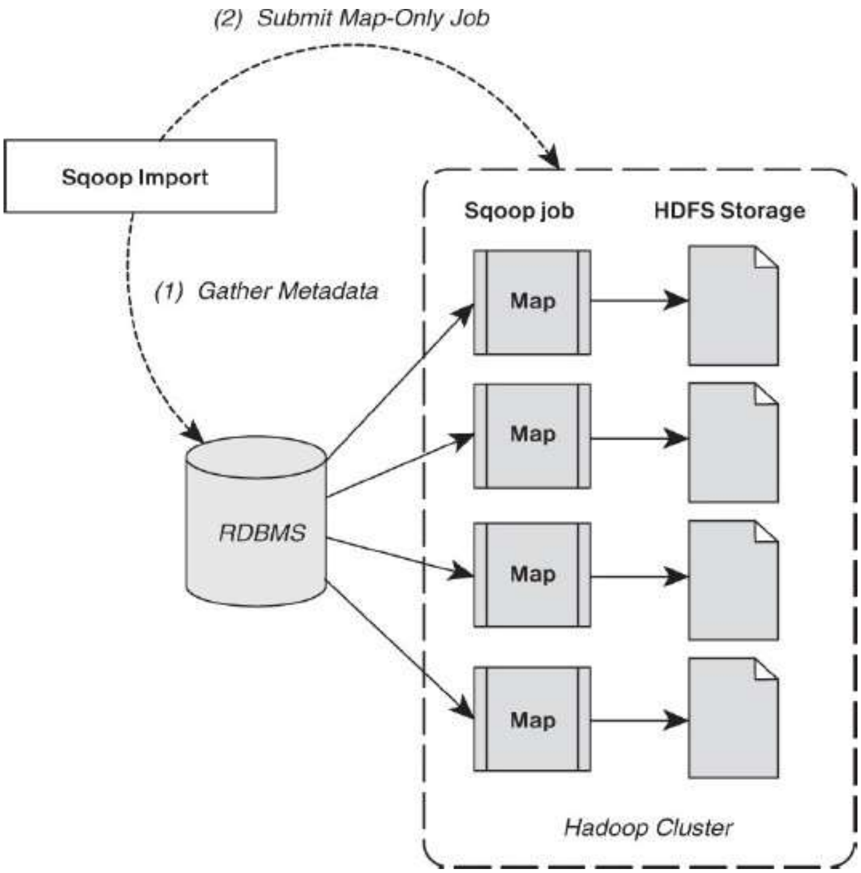
Sqoop is a tool designed to transfer data between Hadoop and relational databases.

Sqoop is used to

-import data from a relational database management system (RDBMS) into the Hadoop Distributed File System(HDFS),

- transform the data in Hadoop and
- export the data back into an RDBMS.

Sqoop import method:



Sqoop import

The data import is done in two steps :

1) Sqoop examines the database to gather the necessary metadata for the data to be imported.

2) Map-only Hadoop job : Transfers the actual data using the metadata.

- The imported data are saved in an HDFS directory.

 Sqoop will use the database name for the directory, or the user can specify any alternative directory where the files should be populated. By default, these files contain comma delimited fields, with new lines separating different records.

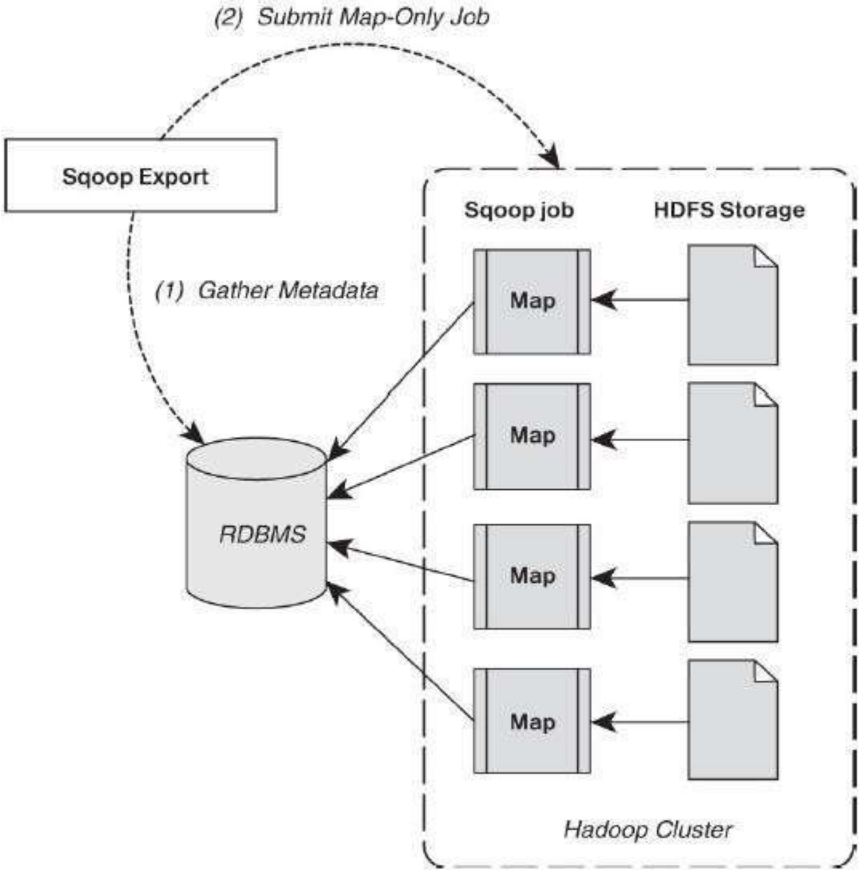
Sqoop Export method :

Data export from the cluster works in a similar fashion. The export is done in two steps :

1) examine the database for metadata.

2) Map-only Hadoop job to write the data to the database.

Sqoop divides the input data set into splits, then uses individual map tasks to push the splits to the database.

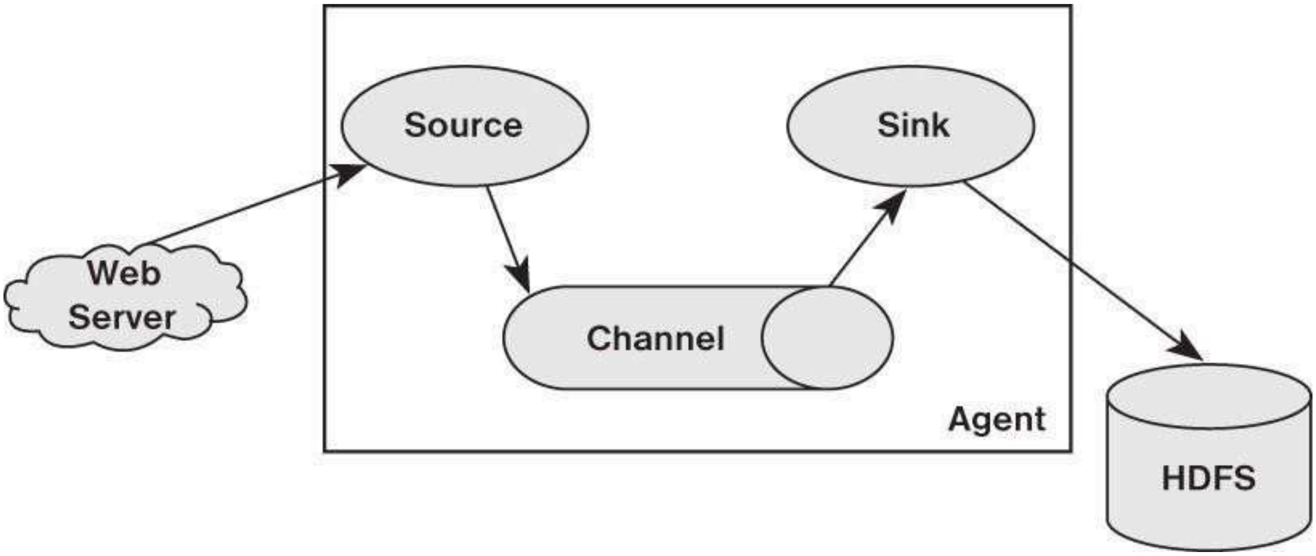


Sqoop export

Apache Flume:

ApacheFlume is an independent agent designed to collect, transport, and store data into HDFS.

Data transport involves a number of Flume agents that may traverse a series of machines and locations. Flume is often used for log files, social media-generated data, email messages, and just about any continuous data source.



Flume agent with source, channel, and sink

Flame agent is composed of three components.
1.**Source**: The source component receives data and sends it to a channel. It can send the data tomore than one channel.

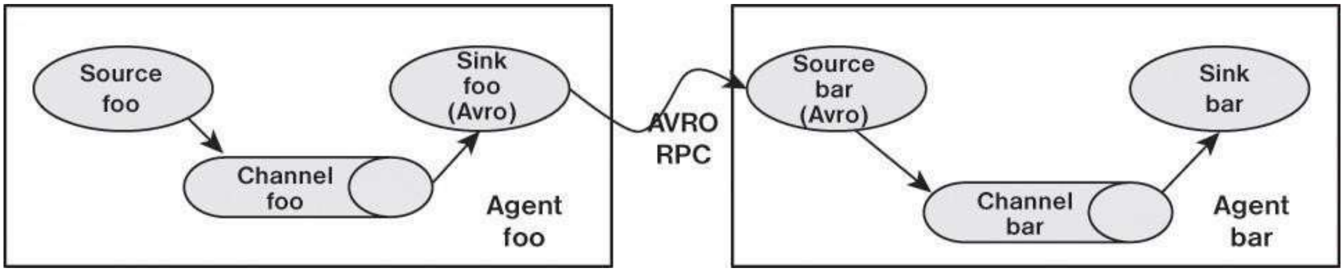
2.**Channel**: A channel is a data queue that forwards the source data to the sink destination.

3.**Sink**: The sink delivers data to destination such as HDFS, a local file, or another Flume agent.
A Flume agent must have all three of these components defined. Flume agent can have several source, channels, and sinks.

Source can write to multiple channels, but a sink can take data from only a single channel.

Data written to a channel remain in the channel until a sink removes the data.

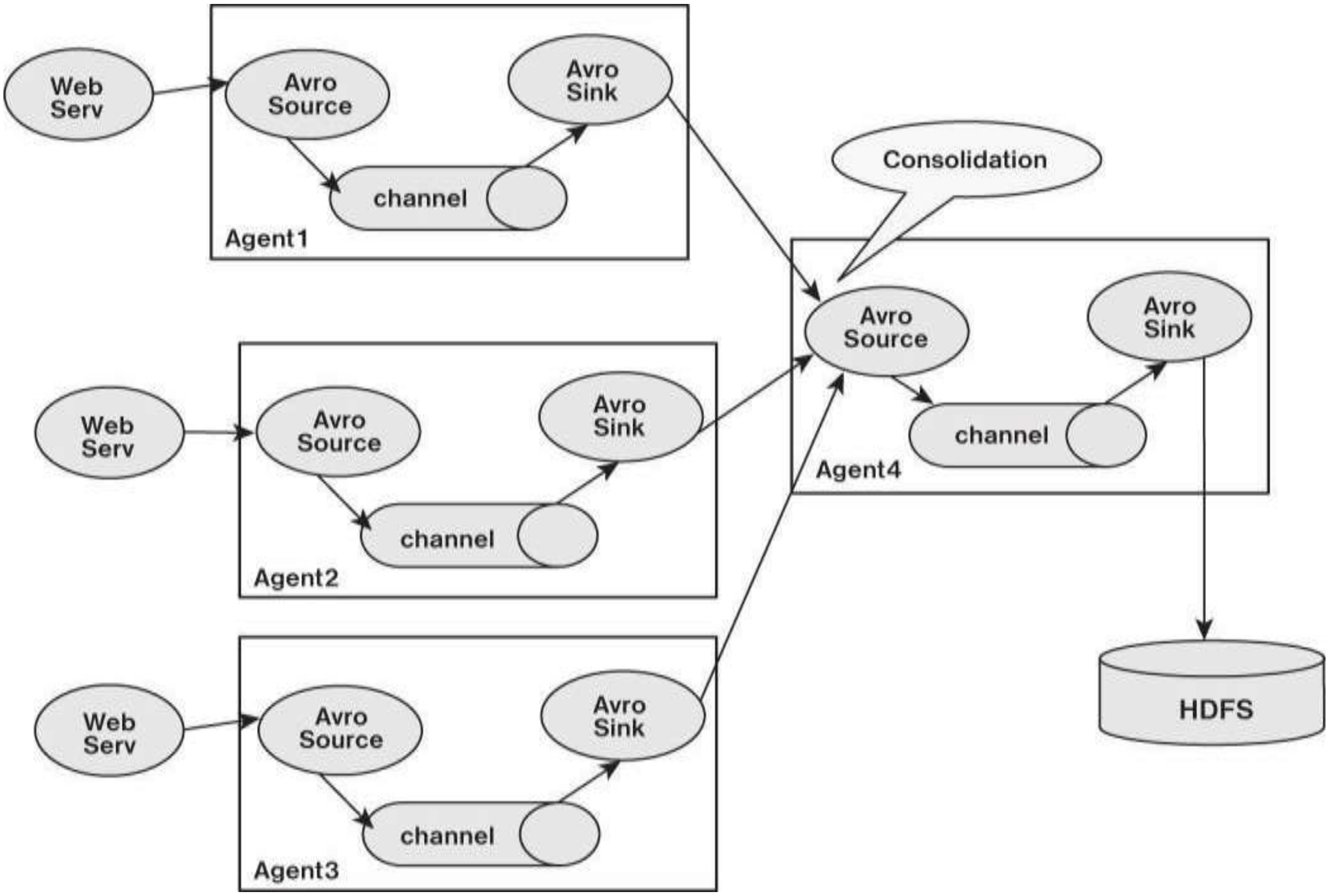
By default, the data in a channel are kept in memory but may be optionally stored on disk to prevent data loss in the event of a network failure.



Pipeline created by connecting Flume agents

As shown in the above figure, Sqoop agents may be placed in a pipeline, possibly to traverse several machines or domains.

- In this Flume pipeline, the sink from one agent is connected to the source of another.
- The data transfer normally used by Flume, which is called Apache Avro.
- Avro is a data serialization/deserialization system that uses a compact binary format.
- The scheme is sent as part of the data exchange and is defined using JSON.
- Avro also uses remote procedure calls (RPCs) to send data.



A Flume consolidation network

Apache Oozie:

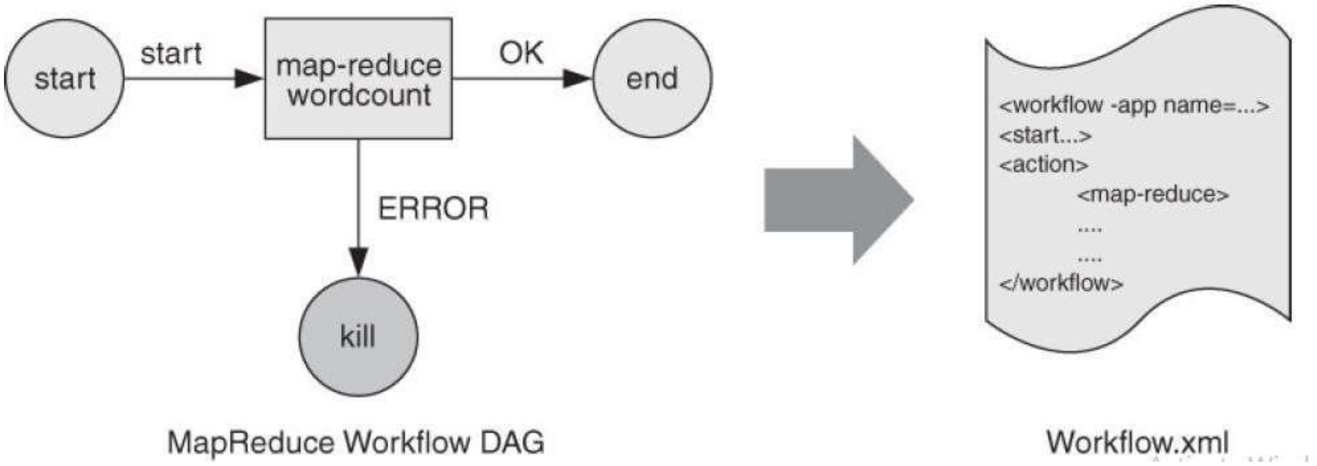
Oozie is a workflow director system designed to run and manage multiple related Apache Hadoop jobs. For instance, complete data input and analysis may require several discrete Hadoop jobs to be run as a workflow in which the output of one job serves as the input for a successive job. Oozie is designed to construct and manage these workflows. Oozie is not a substitute for the YARN scheduler. That is, YARN manages resources for individual Hadoop jobs, and Oozie provides a way to connect and control Hadoop jobs on the cluster.

Oozie workflow jobs are represented as directed acyclic graphs (DAGs) of actions. (DAGs are basically graphs that cannot have directed loops.) Three types of Oozie jobs are permitted:

Workflow—a specified sequence of Hadoop jobs with outcome-based decision points and control dependency. Progress from one action to another cannot happen until the first action is complete. Coordinator—a scheduled workflow job that can run at various time

intervals or when data become available. Bundle—a higher-level Oozie abstraction that will batch a set of coordinator jobs. Oozie is integrated with the rest of the Hadoop stack, supporting several types of

Hadoop jobs out of the box (e.g., Java MapReduce, Streaming MapReduce, Pig, Hive, and Sqoop) as well as system-specific jobs (e.g., Java programs and
shell scripts). Oozie also provides a CLI and a web UI for monitoring jobs.
Following figure depicts a simple Oozie workflow. In this case, Oozie runs a basic MapReduce
operation. If the application was successful, the job ends; if an error occurred, the job is killed.



```
<workflow -app name=...>
<start...>
<action>
        <map-reduce>
        ....
        ....
</workflow>
```

MapReduce Workflow DAG

Workflow.xml

simple Oozie DAG workflow

Apache HBase:

Like Bigtable, HBaseleverages the distributed data storage provided by the underlying distributed file systems spread across commodity servers. Apache HBase provides Bigtable-like capabilities on top of Hadoop and HDFS. Some of the more important features include the following capabilities:
Linear and modular scalability Strictly consistent reads and writes
Automatic and configurable sharding of tables Automatic failover support between RegionServers Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables Easy-to-use Java API for client access.

HBase Data Model Overview

A table in HBase is similar to other databases, having rows and columns. Columns in HBase are grouped into column families, all with the same prefix.

- Specific HBase cell values are identified by a row key, column (column family and column), and version (timestamp).
- It is possible to have many versions of data within an HBase cell.
- A version is specified as a timestamp and is created each time data are written to a cell.
- Rows are lexicographically sorted with the lowest order appearing first in a table.
- The empty byte array denotes both the start and the end of a table’s namespace.
- All table accesses are via the table row key, which is considered its primary key.