Hadoop Big data

I. Introduction

Hadoop: is open source distributed system produced by apache. It's a framework was built in Java to distribute the big dataset among a set of machines. Big data (data set) could be structured data (same as data from database), un-structured data (same as video , documents and HTML document contents) or semi-s structured (same as json or xml documents)

Hadoop has two main components:

- Hadoop Distributed File System (HDFS). The main concepts of HDFS is to distribute the data set to different machines (nodes). The data will sit in different files on the one machine. The size of each file could be 65 MB or 128 MB (depending on configuration of Hadoop)
- MapReduce is framework to process the dataset. MapReduce is consist of two steps. The first step is Map. Map is the way of parsing the data (row for example) into words. The Reduce is the process to group by these words (for example count each word)
-

II. HDFS Components

A Hadoop instance consists of a cluster of HDFS machines often referred to as the Hadoop cluster or the HDFS cluster. There are three main components of an HDFS cluster:

• NameNode: The "master" node of HDFS that manages the data (without actually storing it) by determining and maintaining how the chunks of data are distributed across the DataNodes

• DataNode: The "salve" node of HDFS that stores chunks of data. Also it is responsible for replicating the chunks across other DataNodes. The NameNode and DataNode are daemon processes running in the cluster.

• Secondary NameNode: is responsible for managing the logs in the NameNode (fsimage, edits) logs

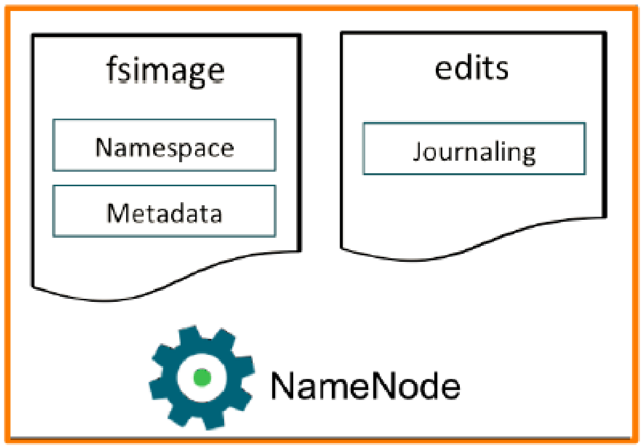
HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode, which is a master server that manages the filesystem namespace and regulates access to files by clients.

NameNode (Master node): has the following characteristics:

• Acts as the master of the DataNodes
• Executes filesystem namespace operations, like opening, closing, and renaming files and directories
• Determines the mapping of blocks to DataNodes
• Maintains the filesystem namespace

The NameNode performs these tasks by maintaining two files:

• fsimage_N: Contains the entire filesystem namespace, including the mapping of blocks to files and filesystem properties
• edits_N: A transaction log that persistently records every change that occurs to filesystem metadata



The NameNode

When the NameNode starts up, it enters safemode (a read-only mode). It loads the `fsimage_N` and `edits_N` from disk, applies all the transactions from the `edits_N` to the in-memory representation of the `fsimage_N`, and flushes out this new version into a new `fsimage_N+1` on disk.

DataNodes (Slave Node)

HDFS exposes a filesystem namespace and allows the data to be stored in files. Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes.

The NameNode determines the mapping of blocks to DataNodes.

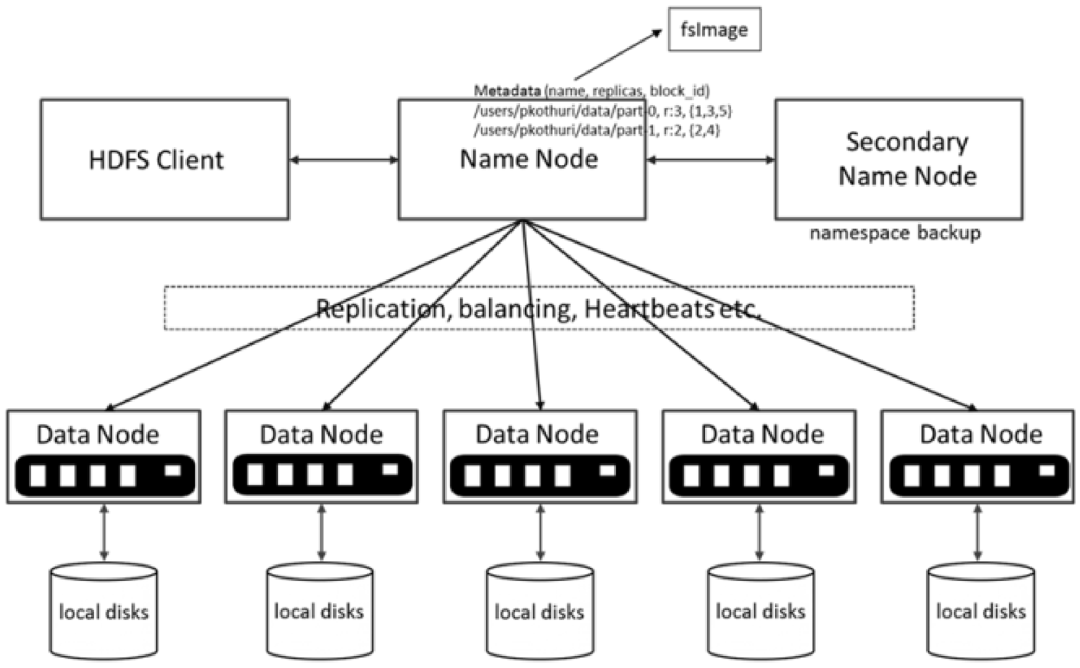
The DataNodes are responsible for:

- Handling read and write requests from application clients
- Performing block creation, deletion, and replication upon instruction from the NameNode (The NameNode makes all decisions regarding replication of blocks)
- Sending heartbeats to the NameNode
- Sending a Blockreport to the NameNode

The NameNode periodically receives a Heartbeat and a Blockreport from each of the DataNodes in the cluster. Receipt of a Heartbeat implies that the DataNode is functioning properly. A Blockreport contains a list of all blocks on a DataNode.

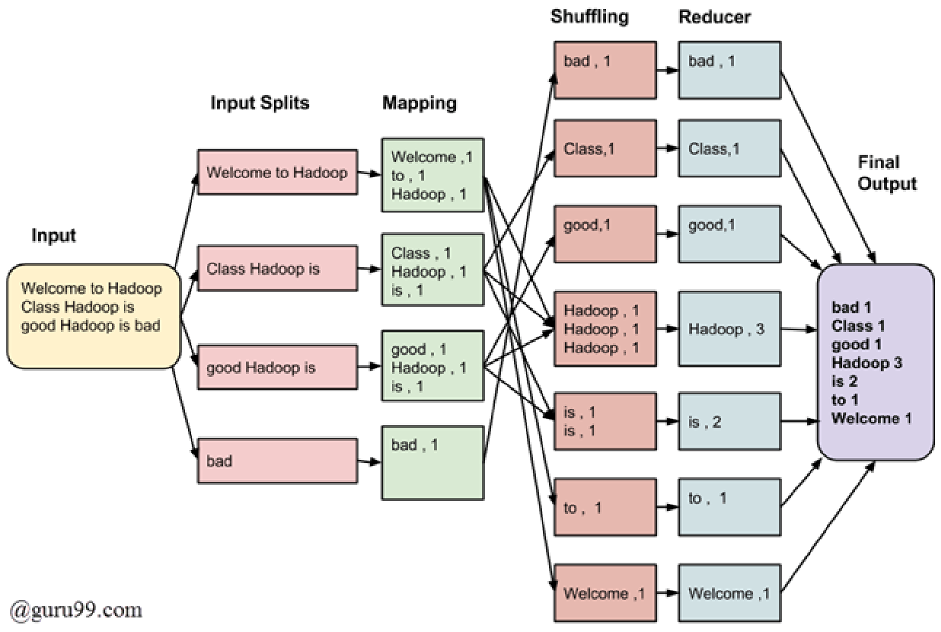
Secondary NameNode:

The Secondary NameNode merges the fsimage and the edits log files periodically and keeps edits log size within a limit. The secondary NameNode service is not a standby secondary NameNode, despite its name. Specifically, it does not offer High Availability (HA) for the NameNode.


fsImage
Metadata (name, replicas, block_id)
/users/pkothuri/data/part 0, r:3, {1,3,5}
/users/pkothuri/data/part 1, r:2, {2,4}
HDFS Client
Name Node
Secondary Name Node
namespace backup
Replication, balancing, Heartbeats etc.
Data Node
local disks

III. MapReduce framework:

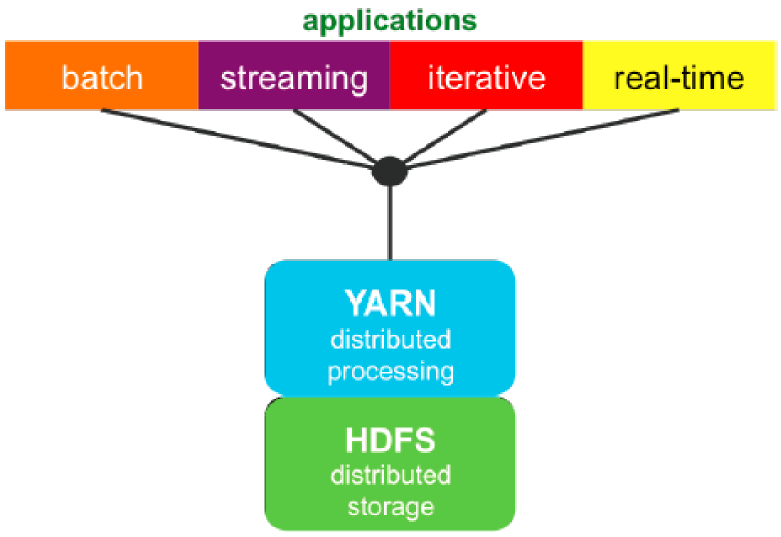
MapReduce is a processing technique built on divide and conquer algorithm. It is made of two different tasks - Map and Reduce. While Map breaks different elements into tuples to perform a job, Reduce collects and combines the output from Map task and fetches it.



We should write MapReduce process in Java. In our session, we will use spark for data processing. We will use python language to write Spark applications. Spark applications run faster than traditional MapReduce. The MapReduce is required in save the results in HDFS many times during the data processing. While Spark keeps the result in the memory through the processing.

IV. Yarn

YARN (Yet Another Resource Manager) is the resource manger which was introduces in Hadoop 2.x. The major process of YARN is take the job which is submitted to Hadoop and then distributed the job among multiple slave nodes.

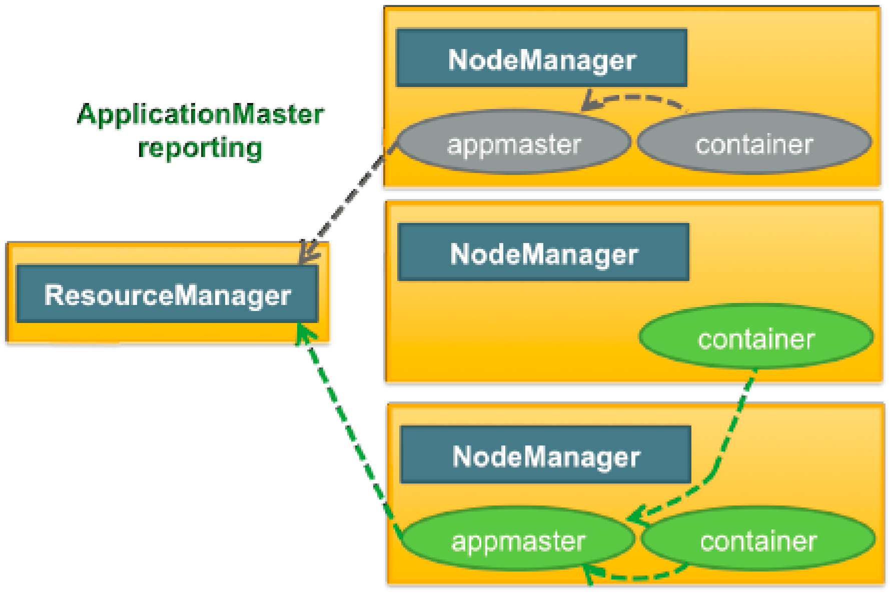


Hadoop processing after Yarn

YARN relies on these three main components for all of its functionality.
The first component is the **ResourceManager (RM)**, which is the arbitrator of all cluster resources. It has two parts: a pluggable scheduler and an ApplicationManager that manages user jobs on the cluster.

The second component is the per-node **NodeManager (NM)**, which manages users' jobs and workflow on a given node. The central ResourceManager and the collection of NodeManagers create the unified computational infrastructure of the cluster.

The third component is the **ApplicationMaster** has the responsibility of negotiating appropriate resource containers from the ResourceManager. It also tracks container status, monitors application progress and restarts failed application tasks. The ApplicationMaster can manage a range of application types, which includes the original
MapReduce application. Each ApplicationMaster runs in a container allocated by the ResourceManager on a slave node.

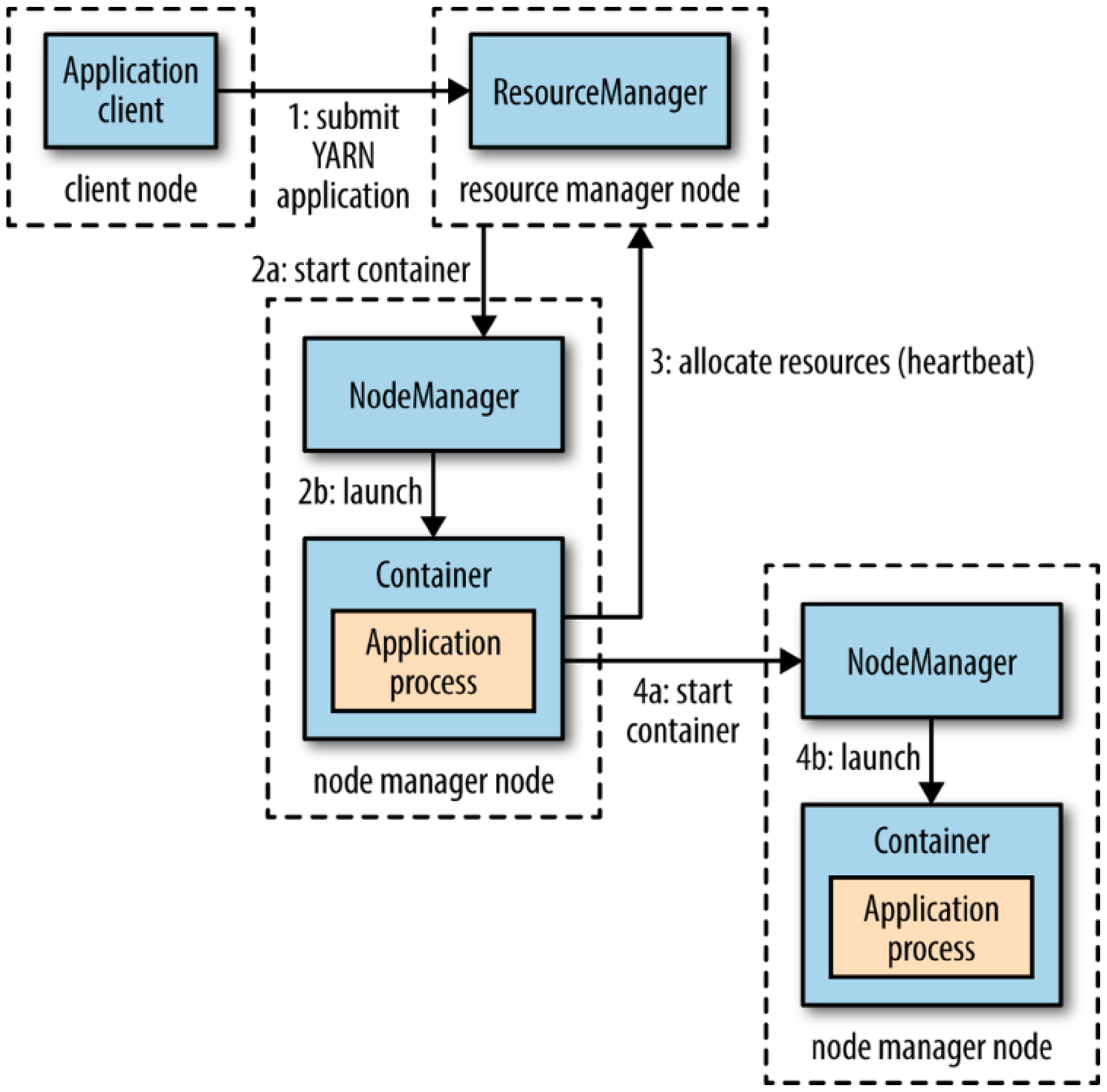


The ApplicationMaster

V. Yarn submit job

- MapReduce job
- Spark submit job

The job should submit with Queue. The queue in Yarn is to allow the applications to share the cluster



VI. The real picture of the cluster

Hadoop keeps all configurations in \etc\hadoop

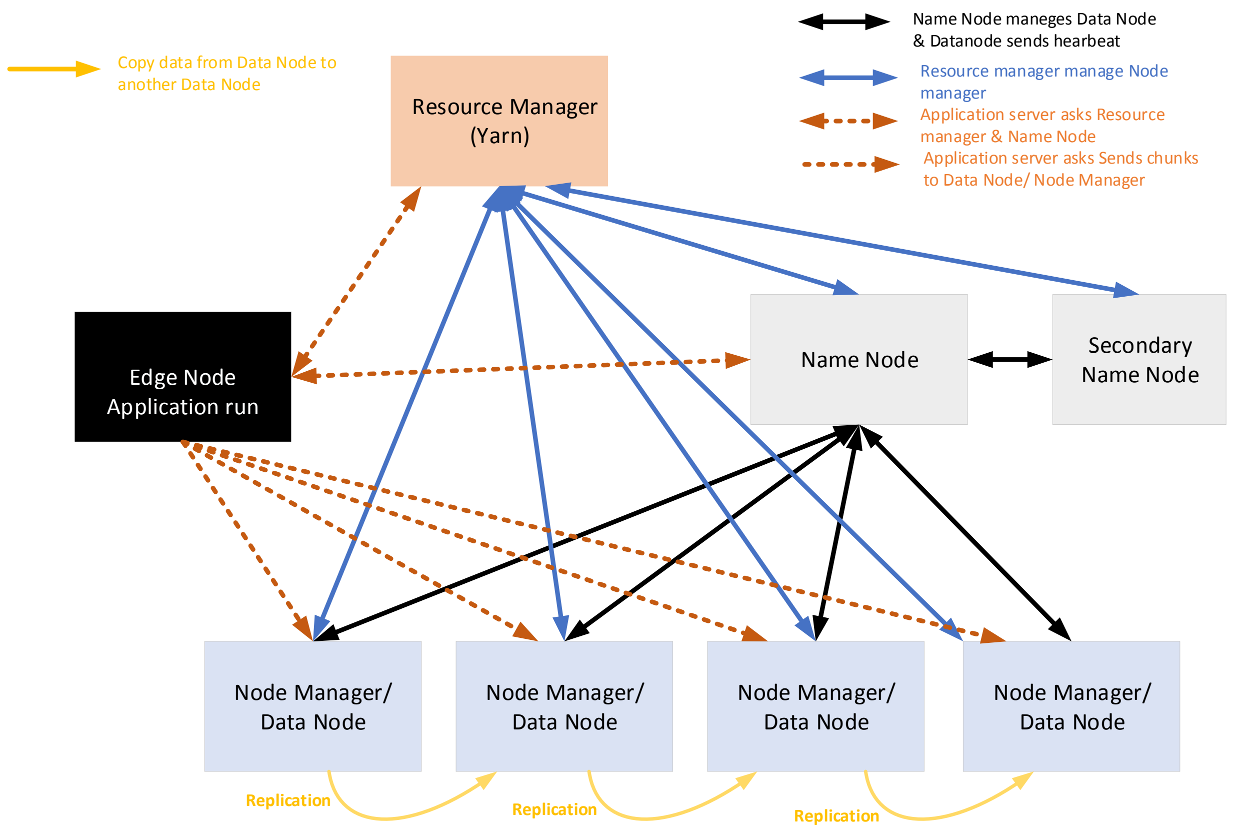
The main configuration files are:

core-site.xml (the file is responsible for NameNode that runs in the cluster)

hdfs-site.xml (The hdfs-site. xml file contains the configuration settings for HDFS daemons; the NameNode, the Secondary NameNode, and the DataNodes)

mapred-site.xml (contains configuration information that overrides the default values for MapReduce)

yarn-site.xml contains all the configuration for (Resource Manager, Container (RmContainer|Resource Container), Schedulers,Hadoop-Configuration, Spark-Yarn, Yarn-Fair Scheduler, Yarn-Log (Container, Application), Yarn-Security)


Name Node maneges Data Node
& Datanode sends hearbeat
Copy data from Data Node to another Data Node
Resource Manager (Yarn)
Resource manager manage Node manager
Application server asks Resource manager & Name Node
Application server asks Sends chunks to Data Node/ Node Manager
Edge Node Application run
Name Node
Secondary Name Node
Node Manager/ Data Node
Node Manager/ Data Node
Node Manager/ Data Node
Node Manager/ Data Node
Replication
Replication
Replication

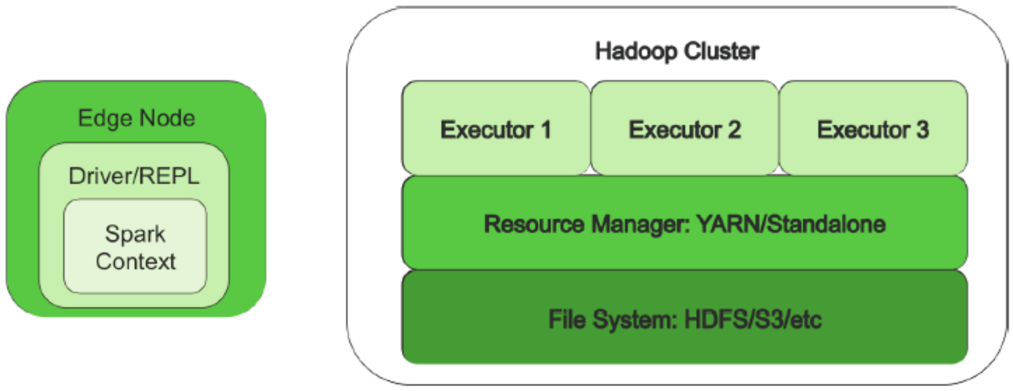
VII. Spark

Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

Apache Spark Application Components

There are 5 core components of a spark application.

- SparkContext
- Driver
- Executor
- Resource Manager
- Storage System (HDFS)



Spark application aunning on a Hadoop cluster

Spark Executor: The Spark executor is the component that does almost all of the work and can be thought of as a worker. An executor is comparable to a mapper and a reducer, but not exactly the same. The executor does both map and reduce tasks, which means they do most of the work of the application

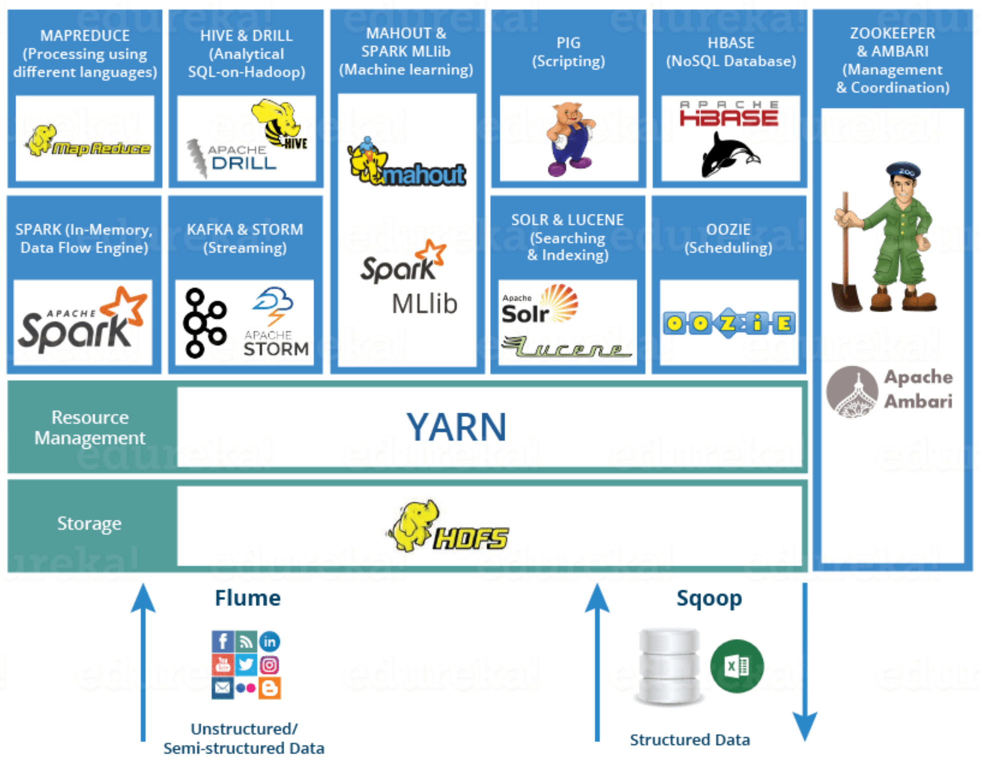
SparkContext: The SparkContext is comparable to a boss. The SparkContext contains all the code and objects required to process the data in the cluster, and works with the resource manager to get the requested resources for the application. It is also responsible for scheduling tasks following the DAG schedule. The SparkContext checks in with the executors to report work being done and provide log updates to the developer.

Spark Driver: The Spark driver is the owner. Similar to the ApplicationMaster in the MapReduce framework, the Spark driver is a JVM that contains the SparkContext. The driver contains the main() function and defines RDDs. It is allocated a predetermined amount of resources for some processing and holding the SparkContext.

Resource Manager : Spark can use two resource managers when running on a cluster. First, and most common, is YARN; second, is the Spark Standalone Resource Manager.

VIII. Hadoop ecosystem

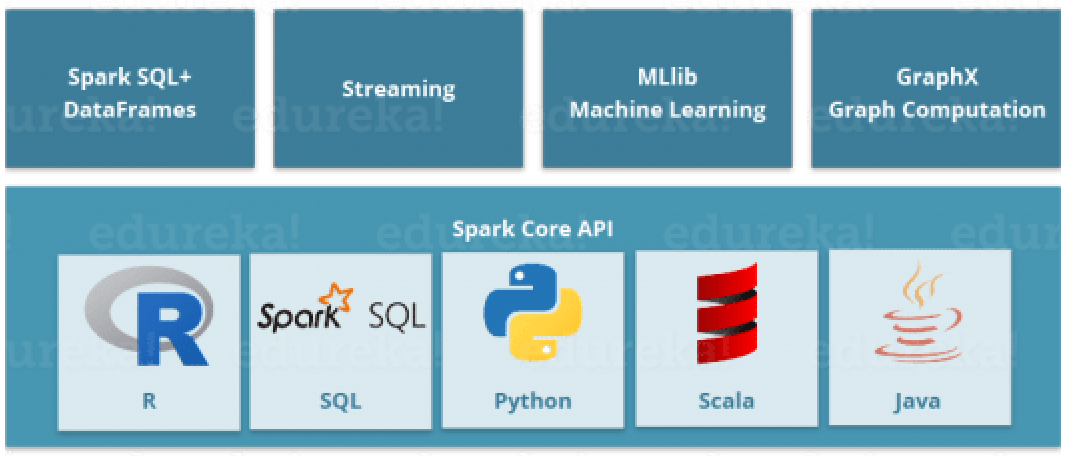
Hadoop ecosystem is a platform or framework which solves big data problems. It’s consist of different open source applications that produce by apache to solve a certain problem in big data. You can consider it as a suite which encompasses a number of services (ingesting, storing, analyzing and maintaining) inside it. Hadoop HDFS/MapReduce/Yarn are the core of Hadoop ecosystem.


MAPREDUCE (Processing using different languages)
HIVE & DRILL (Analytical SQL-on-Hadoop)
MAHOUT & SPARK MLlib (Machine learning)
PIG (Scripting)
HBASE (NoSQL Database)
ZOOKEEPER & AMBARI (Management & Coordination)
SPARK (In-Memory, Data Flow Engine)
KAFKA & STORM (Streaming)
SOLR & LUCENE (Searching & Indexing)
OOZIE (Scheduling)
Apache Ambari
Resource Management
YARN
Storage
HDFS
Flume
Unstructured/ Semi-structured Data
Sqoop
Structured Data

HDFS -> Hadoop Distributed File System
YARN -> Yet Another Resource Negotiator
MapReduce -> Data processing using programming
Spark -> In-memory Data Processing
PIG, HIVE-> Data Processing Services using Query (SQL-like)
HBase -> NoSQL Database
Mahout, Spark MLlib -> Machine Learning
Apache Drill -> SQL on Hadoop
Zookeeper -> Managing Cluster
Oozie -> Job Scheduling
Flume, Sqoop -> Data Ingesting Services
Solr & Lucene -> Searching & Indexing
Ambari -> Provision, Monitor and Maintain cluster

IX. Our journey

in our study, we will concentrate on HDFS. We will show how to write/read data in HDFS. HDFS is a data storage for distributed system. For Data processing or MapReduce process, we will use Spark. Spark is better in performance than traditional MapReduce. In Spark, we will concentrate on Dataframe at data analytics and SparkMLib for machine learning. We will use Python when we write Spark code.



X. Hadoop Commands

Hadoop commands (Lab):

1- Start/Stop hadoop
```
> start-all
> stop-all
```

2- Make directory
```
> hadoop fs -mkdir /lab
```

3- List the folders
```
> Hadoop fs -ls /
```

4- Put file

```
> notepad hadoop-file.txt
> A,A,A,A
  B,B,B,B
  C,C,C,C
> hadoop fs -put hadoop-file.txt /lab/
> hadoop fs -mkdir /tmp
> hdfs dfs -copyFromLocal hadoop-file.txt /temp
```

5- Get file

```
> hadoop fs -get /lab/hadoop-file.txt
> hdfs dfs -copyToLocal  /temp/hadoop-file.txt hadoop-file-99.txt
```

6- Cat file

```
> hadoop fs -cat /lab/hadoop-file.txt
```

7- Copy file

```
> hadoop fs -mkdir /temp2
> hadoop fs -cp /temp/ hadoop-file.txt /temp2
```

8- Move file

```
>hadoop fs -mkdir
```

9- Delete directory / Delete file

10- File size in folder

```
>  hdfs dfs -du /temp
```

10- File information

```
> hdfs fsck /temp/hadoop-file.txt -files -blocks -locations
```

11- Change mode permissions for the folders or files

```
> hadoop fs -chmod 777  /temp/hadoop-file.txt
```

Yarn commands:

Display available queues

```
> hadoop queue -list
```

Note: system has default queue, when we want to add new queue, we have to go to capacity-scheduler.xml

```
> yarn rmadmin -refreshQueues
```

Note: Refresh the queues after update capacity-scheduler.xml