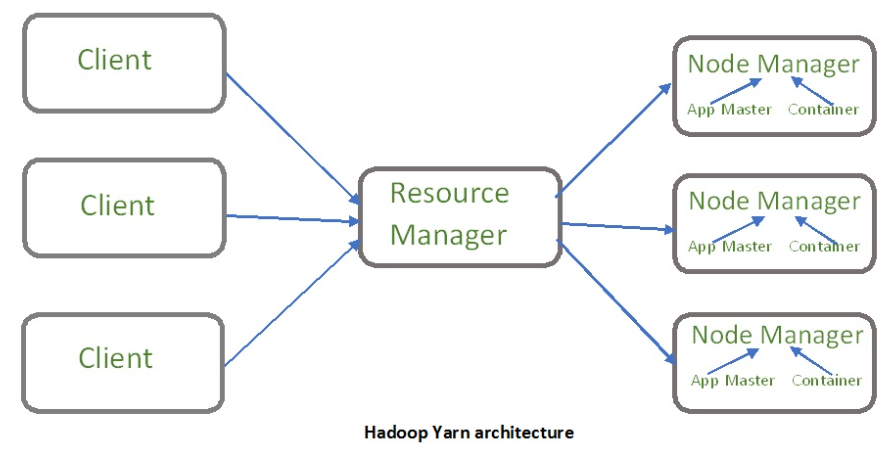
Big Data Assignment 1

1. **Briefly describe the YARN architecture.**

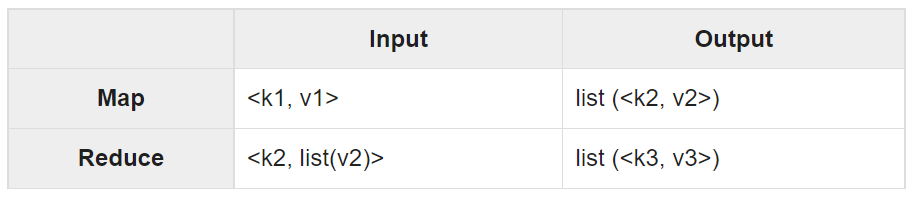
YARN stands for “Yet Another Resource Negotiator“. It was introduced in Hadoop 2.0 to remove the bottleneck on Job Tracker which was present in Hadoop 1.0. YARN architecture basically separates resource management layer from the processing layer. The architecture of YARN is shown in the below figure. It consists of the following components:

* **Client:** It submits map-reduce jobs.
* **Resource Manager:**
  + It is the master daemon of YARN and is responsible for resource assignment and management among all the applications. Whenever it receives a processing request, it forwards it to the corresponding node manager and allocates resources for the completion of the request accordingly. It has two major components:
  + **Scheduler:**
    - It performs scheduling based on the allocated application and available resources. It is a pure scheduler, means it does not perform other tasks such as monitoring or tracking and does not guarantee a restart if a task fails. The YARN scheduler supports plugins such as Capacity Scheduler and Fair Scheduler to partition the cluster resources.
  + **Application manager:**
    - It is responsible for accepting the application and negotiating the first container from the resource manager. It also restarts the Application Master container if a task fails.
* **Node Manager:**
  + It takes care of individual node on Hadoop cluster and manages application and workflow in that particular node. Its primary job is to keep-up with the Resource Manager. It registers with the Resource Manager and sends heartbeats with the health status of the node. It monitors resource usage, performs log management and also kills a container based on directions from the resource manager. It is also responsible for creating the container process and start it on the request of Application master.
* **Application Master:**
  + An application is a single job submitted to a framework. The application master is responsible for negotiating resources with the resource manager, tracking the status and monitoring progress of a single application. The application master requests the container from the node manager by sending a Container Launch Context (CLC) which includes everything an application needs to run. Once the application is started, it sends the health report to the resource manager from time-to-time.
* **Container:**
  + It is a collection of physical resources such as RAM, CPU cores and disk on a single node. The containers are invoked by Container Launch Context (CLC) which is a record that contains information such as environment variables, security tokens, dependencies etc.

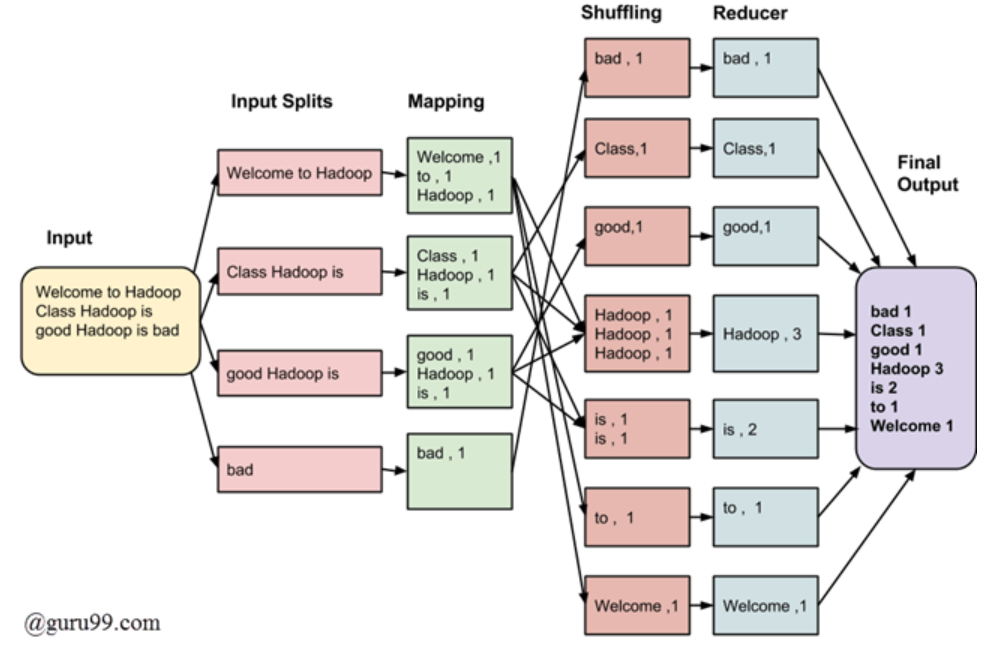


1. **In Hadoop, what is map and reduce? Create real-world examples that explain how to use maps and reduce.**

MapReduce is a framework using which we can write applications to process huge amounts of data, in parallel, on large clusters of commodity hardware in a reliable manner. It is based on Java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job. MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.

* **Map stage −** The map or mapper’s job is to process the input data. Generally, the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
* **Reduce stage −** This stage is the combination of the Shuffle stage and the Reduce stage. The Reducer’s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
* **Broad Steps:**
  + 
  + Map phase takes input in Key-Value pair
  + It produces output in the form of Key-Value pair.
  + Output from various Map tasks are grouped together on the basis of Key.
  + Key and its associated set of values are sent to the Reduce phase.
  + Reduce method operates on key and associated list of values.
  + Output of Reduce is written to HDFS.

**Example – Word Count**

* ****
* **Split:** An input to a MapReduce in Big Data job is divided into fixed-size pieces called input splits Input split is a chunk of the input that is consumed by a single map
* **Map:** In our example, a job of mapping phase is to count a number of occurrences of each word from input splits (more details about input-split is given below) and prepare a list in the form of <word, frequency>
* **Shuffle:** In our example, the same words are clubbed together along with their respective frequency.
* **Reduce:** In our example, this phase aggregates the values from Shuffling phase i.e., calculates total occurrences of each word.
* Final output is obtained by combining input from reducer.

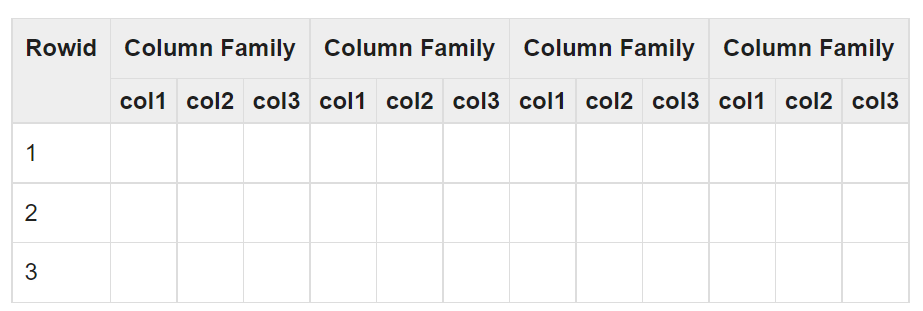
1. **What are the differences between HBASE and NoSQL? What are the most important features of HBase?**

HBase is a part of the Hadoop ecosystem that provides random real-time read/write access to data in the Hadoop File System. HBASE is a column-oriented database that runs on top of HDFS. It is a NoSQL database which does not understand the structured query. It is usually suited for sparse datasets. Unlike Hive, HBase operations run in real-time on its database rather than MapReduce jobs. HBase is partitioned to tables, and tables are further split into column families. Column families, which must be declared in the schema, group together a certain set of columns (columns don’t require schema definition). For example, the "message" column family may include the columns: "to", "from", "date", "subject", and "body".

In short, in an HBase:

* Table is a collection of rows.
* Row is a collection of column families.
* Column family is a collection of columns.
* Column is a collection of key value pairs.

Given below is an example schema of table in HBase.



**Applications of HBase:**

* It is used whenever there is a need to write heavy applications.
* HBase is used whenever we need to provide fast random access to available data.
* Companies such as Facebook, Twitter, Yahoo, and Adobe use HBase internally.

**Features of HBASE:**

* HBase is linearly scalable.
* It has automatic failure support.
* It provides consistent read and writes.
* It integrates with Hadoop, both as a source and a destination.
* It has easy java API for client.
* It provides data replication across clusters.

1. **In HBase, what data manipulation commands are available?**

HBase works by storing data as key/value. It supports four primary operations: put to add or update rows, scan to retrieve a range of cells, get to return cells for a specified row, and delete to remove rows, columns or column versions from the table. Versioning is available so that previous values of the data can be fetched (the history can be deleted every now and then to clear space via HBase compactions). Although HBase includes tables, a schema is only required for tables and column families, but not for columns. The Data Manipulation commands available in HBase are:

* **put** - Puts a cell value at a specified column in a specified row in a particular table.
* **get** - Fetches the contents of row or a cell.
* **delete** - Deletes a cell value in a table.
* **deleteall** - Deletes all the cells in a given row.
* **scan** - Scans and returns the table data.
* **count** - Counts and returns the number of rows in a table.
* **truncate** - Disables, drops, and recreates a specified table.
* **Java client API** - Prior to all the above commands, Java provides a client API to achieve DML functionalities, CRUD (Create Retrieve Update Delete) operations and more through programming, under org.apache.hadoop.hbase.client package. HTable Put and Get are the important classes in this package.

1. **In HBase, explain tombstone markers.**

When a Delete command is issued through the HBase client, no data is actually deleted. Instead, a tombstone marker is set, making the deleted cells effectively invisible. User Scans and Gets automatically filter deleted cells until they get removed. HBase periodically removes deleted cells during compactions. The tombstone markers are only deleted during major compactions (which compacts all store files to a single one), because in order to prove that a tombstone marker has no effect HBase needs to look at all cells. There are three types of tombstone markers:

* **Version delete marker**
  + Marks a single version of a column for deletion
* **Column delete marker**
  + Marks all versions of a column for deletion
* **Family delete marker**
  + Marks all versions of all columns for a column family for deletion

**Note on Versions:**

* In the Apache HBase you can have many cells where row and columns are same but differs only in version values. A version is a timestamp values is written alongside each value.
* Why HBase Maintain Versions?
  + Since HBase also uses hdfs, it’s not easy to update data. So, to enable that feature HBase creates a version on the cells being updated. By default, it maintains 3 versions.
  + For example, let us assume you have row with value 123, and updated this value with 456. HBase does not overwrite the 123 with 456, instead, it will add another row with updated value and latest timestamp as a version number.