Name: Jessica Saini
Registration Number: 15BCE0164

Implementation of Hadoop and Mapreduce in K-Nearest neighbor Algorithm

Hadoop

Apache Hadoop is an open-source software framework used for distributed storage and processing of dataset of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware. All the modules in Hadoop are designed with a fundamental assumption that hardware failures are common occurrences and should be automatically handled by the framework.

The core of Apache Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part which is a MapReduce programming model. Hadoop splits files into large blocks and distributes them across nodes in a cluster.

It then transfers packaged codeinto nodes to process the data in parallel. This approach takes advantage of data locality, where nodes manipulate the data they have access to. This allows the dataset to be processed faster and more efficiently than it would be in a more conventional supercomputer architecture that relies on a parallel file system where computation and data are distributed via high-speed networking.

The base Apache Hadoop framework is composed of the following modules:

- ***Hadoop Common*** – contains libraries and utilities needed by other Hadoop modules;
- ***Hadoop Distributed File System (HDFS)*** – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
- ***Hadoop YARN*** – a platform responsible for managing computing resources in clusters and using them for scheduling users' applications; and
- ***Hadoop MapReduce*** – an implementation of the MapReduce programming model for large-scale data processing.

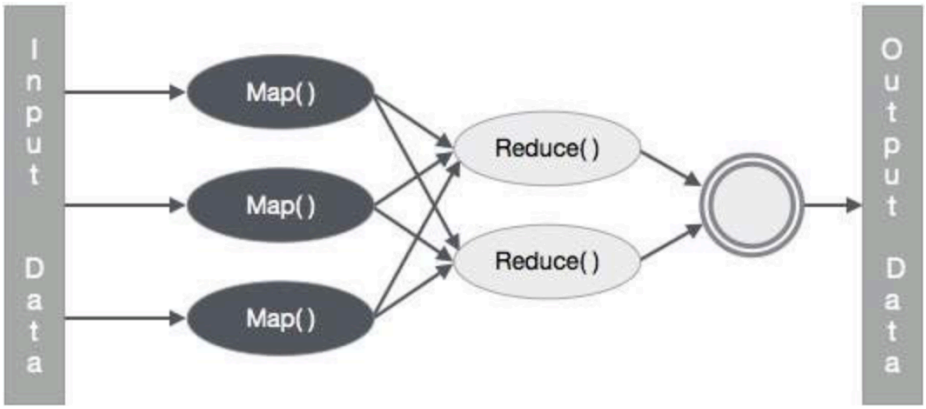
The Hadoop framework itself is mostly written in the Java programming language, with some native code in C and command line utilities written as shell scripts. Though MapReduce Java code is common, any programming language can be used with "Hadoop Streaming" to implement the "map" and "reduce" parts of the user's program.

MapReduce

MapReduce is a programming model and an associated implementation for processing and generating big data sets with a parallel, distributed algorithm on a cluster.

A MapReduce program is composed of a **Map()** procedure (method) that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a **Reduce()** method that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce

System" (also called "infrastructure" or "framework") orchestrates the processing by marshalling the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for redundancy and fault tolerance.



Map function

The *Map* function takes a series of key/value pairs, processes each, and generates zero or more output key/value pairs. The input and output types of the map can be (and often are) different from each other.

If the application is doing a word count, the map function would break the line into words and output a key/value pair for each word. Each output pair would contain the word as the key and the number of instances of that word in the line as the value.

Reduce function

The framework calls the application's *Reduce* function once for each unique key in the sorted order. The *Reduce* can iterate through the values that are associated with that key and produce zero or more outputs.

In the word count example, the *Reduce* function takes the input values, sums them and generates a single output of the word and the final sum.

	Input	**Output**
Map	<k1, v1>	list (<k2, v2>)
Reduce	<k2, list(v2)>	list (<k3, v3>)

K-Nearest Algorithm

Generalising the k-Nearest Neighbor Testing Phase

1. Determine the value of ‘k’ (input)

2. Prepare the training data set by storing the coordinates and class labels of the data points.

3. Load the data point from the testing data set.

4. Conduct a majority vote amongst the ‘k’ closest neighbors of the testing data point from the training
data set based on a distance metric.

5. Assign the class label of the majority vote winner to the new data point from the testing data set.

6. Repeat this until all the Data points in the testing phase are classified.

Implementation:

4 nodes were employed overa a non-public computer network. One node was used as a Namenode and 3 other nodes were used as Datanodes and Task Trackers. . All the four nodes had Intel i3 processors with a pair of 2.40Ghz and 2 GB memory. 40Ghz and a couple of GB memory. The artificial language accustomed code the kNN rule in each sequent and MapReduce Implementation was JAVA. Apache Hadoop was installed on all the nodes.