Running MapReduce Programs On Single Node Hadoop Cluster - Word Count/Word Frequency

Expt No: 3 March 06, 2019
Author: Subalakshmi Shanthosi S (186001008)

Aim

Implementation of MapReduce in Hadoop single node cluster.

Description

- Apache Hadoop
  - Large Scale,Open Source Software Framework.
  - Supports Three Projects:
    - Hadoop Common.
    - HDFS : Hadoop Distributed File System.
    - MapReduce.
- Hadoop MapReduce
  - Hadoop Programming Model and Software Framework.
  - Computational Processing:
    - Unstructured Data : File system
    - Structured Data : Database
  - MapReduce Layer has job and task tracker nodes.
  - Cluster nodes:
    - Single JobTracker per master.
    - Single TaskTracker per slave.
  - Fundamental Steps:
    - Map Step:
      - Master node slices problem input into several subproblems input.
      - Distributes data slices to worker nodes.
      - Worker nodes processes and hands over the control to master.
    - Reduce Step:
      - Master node takes the answers to the sub problems and combines them in a predefined way to get the output/answer to original problem.

Software's Used

- Ubuntu 16.04 LTS
- Hadoop 1.0.3

Hadoop MapReduce Architecture

- MapReduce Architecture:

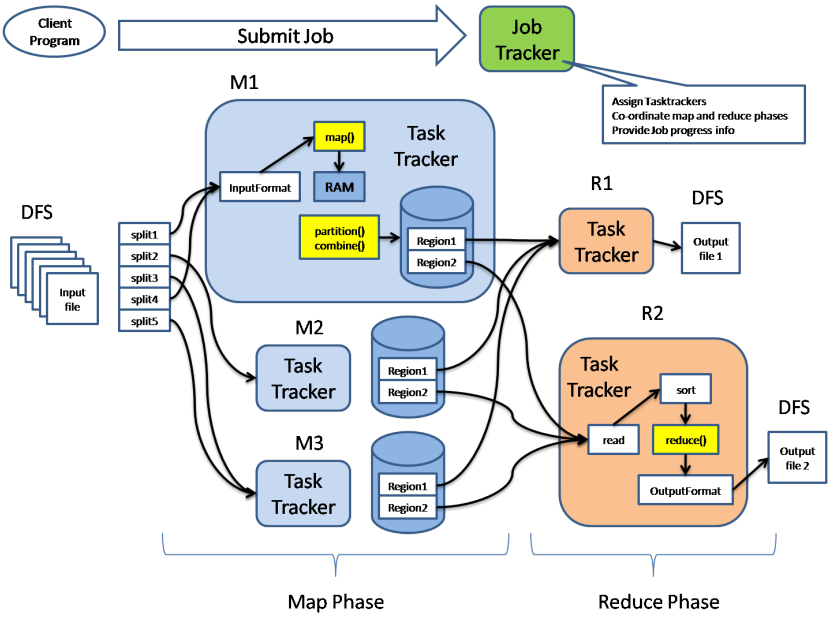

Figure 1: Hadoop MapReduce Architecture Diagram.

Procedure

1. Launch Ubuntu 16.04 LTS in virtual environment.
2. Login to the OS with sudo permission and install the following packages using apt-get command
   - openssh-server
   - openssh-client
   - java jdk 8
   - javac compiler
   - hadoop 1.0.3

3. Install and configure appropriate environment variables for Hadoop 1.0.3.
4. Start Hadoop DFS by invoking script start-all.sh in bin directory.
5. Examine the running Hadoop Job Process using jps command.
6. Download and place the input files in appropriate directory.
7. Copy the input Files from Local File System to HDFS using command attribute CopyFromLocal of dfs command.
8. Run the MapReduce Job.
9. Retrieve the Job Result from HDFS.
10. View stats from Web Interface for the following information listed below.
    - JobTracker Web Interface - `http://localhost:50030/`
    - TaskTracker Web Interface - `http://localhost:50060/`
    - NameNode Web Interface - `http://localhost:50070/`

Code

```
// WordCount.java
package org.myorg;
import java.io.IOException;
import java.util.*;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;
import org.apache.hadoop.util.*;
public class WordCount {

public static void main(String[] args) throws Exception {
   JobConf conf = new JobConf(WordCount.class);
   conf.setJobName("wordcount");
   conf.setOutputKeyClass(Text.class);
   conf.setOutputValueClass(IntWritable.class);
   conf.setMapperClass(Map.class);
   conf.setReducerClass(Reduce.class);
   conf.setInputFormat(TextInputFormat.class);
   conf.setOutputFormat(TextOutputFormat.class);
   FileInputFormat.setInputPaths(conf, new Path(args[0]));
   FileOutputFormat.setOutputPath(conf, new Path(args[1]));
   JobClient.runJob(conf);
}

}
```

```
\\Map.java
public static class Map extends MapReduceBase ... {
   private final static IntWritable one = new IntWritable(1);
   private Text word = new Text();
   public void map(LongWritable key, Text value,
   OutputCollector<Text, IntWritable> output, ...) ... {
   String line = value.toString();
   StringTokenizer tokenizer = new StringTokenizer(line);
   while (tokenizer.hasMoreTokens()) {
      word.set(tokenizer.nextToken());
      output.collect(word, one);
      } } }
```

```
\\Reduce.java
public static class Reduce extends MapReduceBase ... {
   public void reduce(Text key, Iterator<IntWritable> values,
   OutputCollector<Text, IntWritable> output, ...) ... {
   int sum = 0;
   while (values.hasNext()) {
      sum += values.next().get();
      }
   output.collect(key, new IntWritable(sum));
      }
   }
}
```

Output

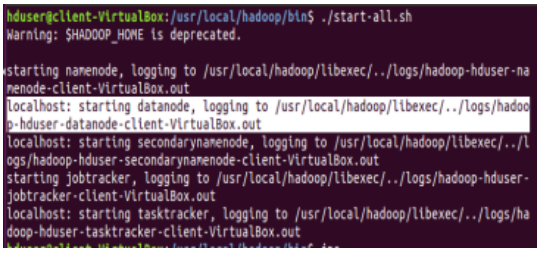

Figure 2: Starting Hadoop DFS.

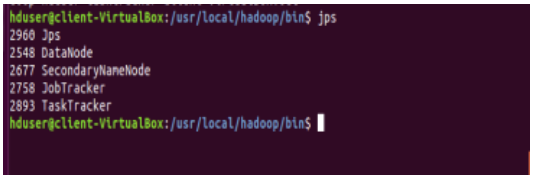

Figure 3: Examining Running Hadoop Process.

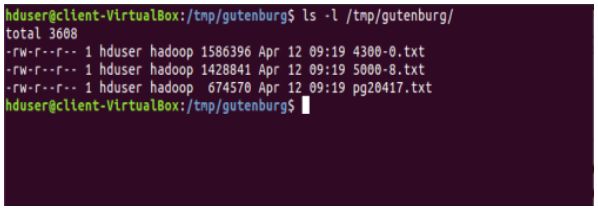

Figure 4: Placing the inputFiles in appropriate location.

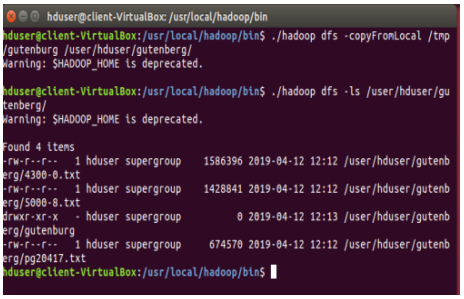

Figure 5: Copy Files from Local to HDFS.

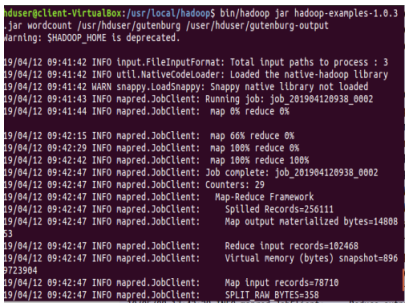

Figure 6: Run MapReduce Task for Input Files.

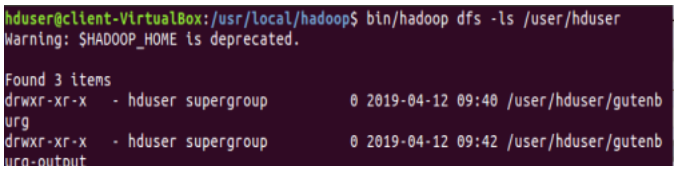

Figure 7: Retrieve Hadoop MapReduce Output from appropriate folder.

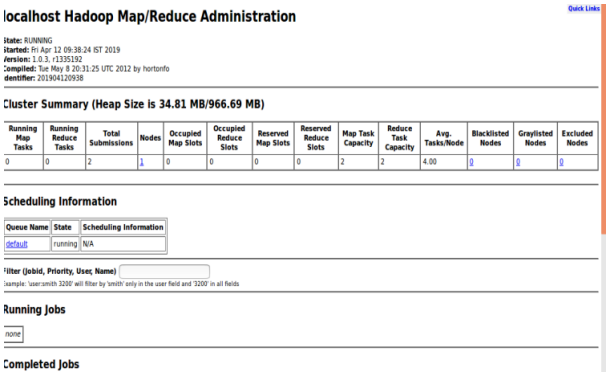

Figure 8: JobTracker of HDFS Web Interface.

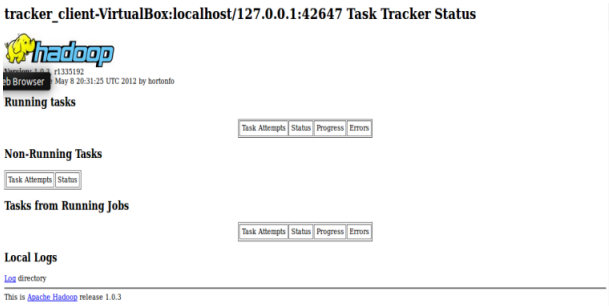

Figure 9: Task Tracker of HDFS Web Interface.

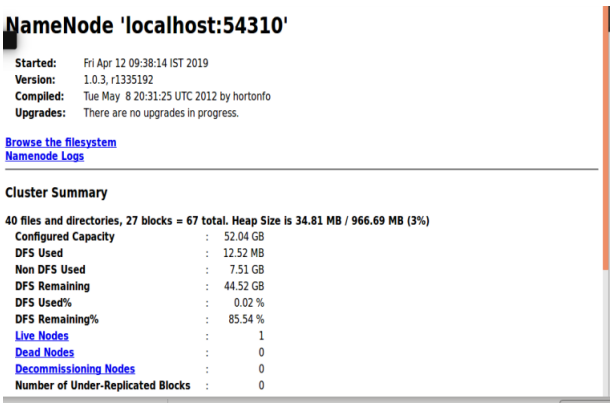

Figure 10: Web Interface of HDFS Namenode.

Result

Thus the hadoop MapReduce program for finding word cound and frequency was successly executed and its results in Web Interface for a better understanding.