Lab 3: MapReduce Programming

zc1134, xj573

# Objectives

* Understand the MapReduce concept.
* Get familiar with the Hadoop framework.
* Experience working with small Hadoop cluster using VMs.

# Equipment Needs

* Computers
* Internet

# Experiments

## Basics

1. Go through the Apache Hadoop introduction to get the general idea about Hadoop:

<http://hadoop.apache.org/>

1. Go through the Apache Hadoop release notes to understand the evolution of Hadoop:

<http://hadoop.apache.org/releases.html>

## Hadoop Single Node Mode.

1. Follow the instructions on   
   <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html>  
   to set up Hadoop environment on your own Linux machine.
2. Follow the instructions/tutorials from the above link to run simple practice with single node mode.

## Hadoop “cluster”

Create two VMs (one master and one slave) on your own computer (ex: Virtualbox) and construct a small hadoop cluster for running the word count program. You can also use Docker containers to perform this assignment.

1. Create two VMs.
2. Configure the VM network so that the VMs can ping and communicate with each other.
3. Download and install hadoop and all required tools.
4. Configure hadoop configure files for your two-VM cluster
5. Run the WordCount Hadoop job on the file (wordCountText.txt) provided by the TA.

# Reports

1. **What are the differences between Hadoop 0.X, 1.X, 2.X?**

|  |  |  |
| --- | --- | --- |
| Version | Components | Operations |
| Hadoop 0.X | 1. HDFS. 2. Pipes: A C++ API for MapReduce |  |
| Hadoop 1.X | 1. HDFS(V1) 2. MapReduce(V1) | It’s a single purpose system, and it supports only MapReduce-based Batch/Data Processing Application |
| Hadoop 2.X | 1. HDFS(V2) 2. YARN(V2) 3. MapReduce(v1) | 1. Hadoop 1.x Job Tracker component is divided into two components: resource manager and application master 2. Decoupling MapReduce component responsibilities into different components. 3. Introducing new YARN component for Resource management. 4. Ddecoupling component’s responsibilities, it supports multiple namespace, Multi-tenancy, Higher Availability and Higher Scalability. |

1. **What is YARN? Why do we need YARN?**

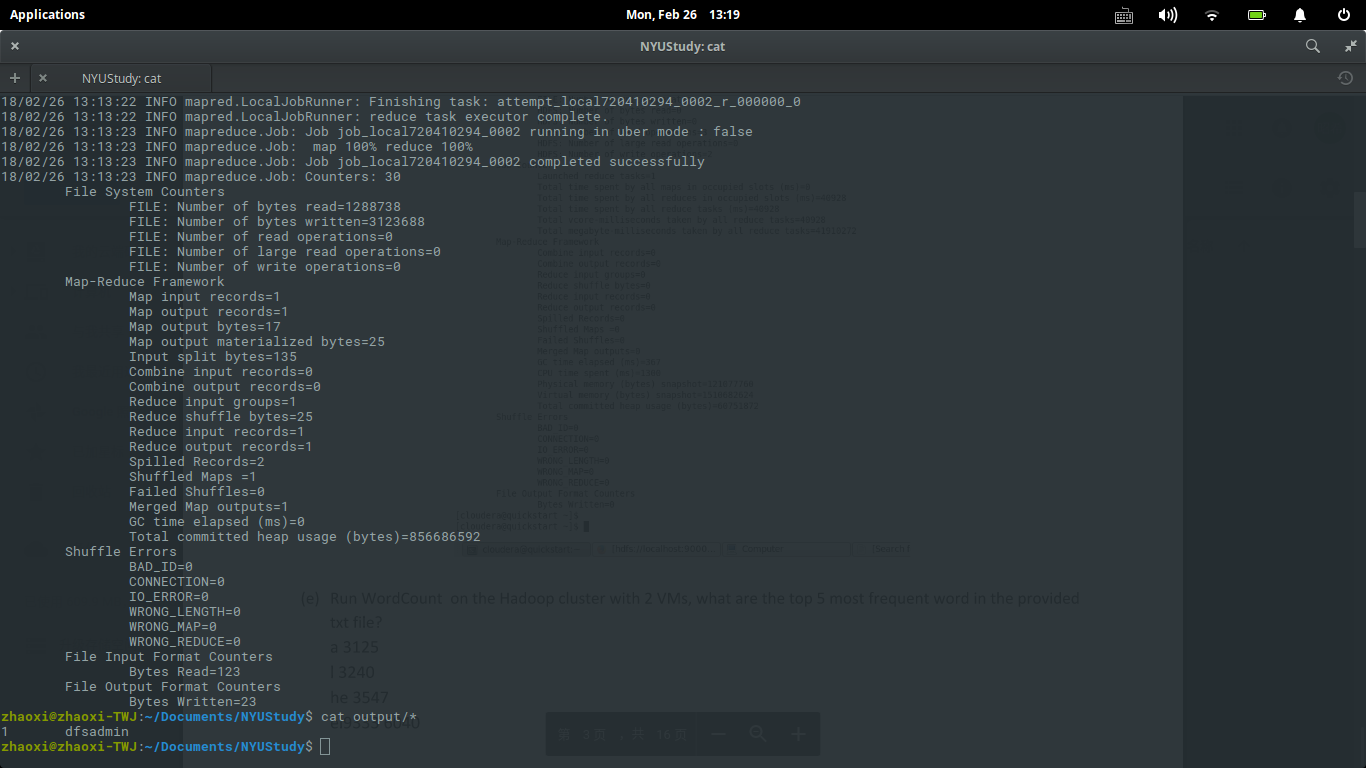
YARN is short for Yet Another Resource Negotiator, a cluster management technology. YARN is a software rewrite that decouples MapReduce's resource management and scheduling capabilities from the data processing component, enabling [Hadoop](http://searchcloudcomputing.techtarget.com/definition/Hadoop) to support more varied processing approaches and a broader array of applications. For example, [Hadoop clusters](http://searchbusinessanalytics.techtarget.com/definition/Hadoop-cluster) can now run interactive querying and streaming data applications simultaneously with MapReduce batch jobs. The original incarnation of Hadoop closely paired the Hadoop Distributed File System ([HDFS](http://searchbusinessanalytics.techtarget.com/definition/Hadoop-Distributed-File-System-HDFS)) with the batch-oriented [MapReduce](http://searchcloudcomputing.techtarget.com/definition/MapReduce) programming framework, which handles resource management and job scheduling on Hadoop systems and supports the parsing and condensing of data sets in parallel. YARN combines a central resource manager that reconciles the way applications use Hadoop system resources with node manager [agents](http://whatis.techtarget.com/definition/software-agent) that monitor the processing operations of individual cluster nodes. Running on [commodity hardware](http://whatis.techtarget.com/definition/commodity-hardware) clusters, Hadoop has attracted particular interest as a staging area and data store for large volumes of structured and unstructured data intended for use in analytics applications. Separating HDFS from MapReduce with YARN makes the Hadoop environment more suitable for operational applications that can't wait for batch jobs to finish.

Hadoop 2.X YARN has the following benefits:

1. Highly Scalability
2. Highly Availability
3. Supports Multiple Programming Models
4. Supports Multi-Tenancy
5. Supports Multiple Namespaces
6. Improved Cluster Utilization
7. Supports Horizontal Scalability
8. **What is Hadoop streaming?**

Hadoop streaming is a utility that comes with the Hadoop distribution. The utility allows you to create and run Map/Reduce jobs with any executable or script as the mapper and/or the reducer.

1. **Screenshots of the practice of single node mode Hadoop on your own computer. The screenshot should show the output and result of Hadoop execution as well as the files in HDFS.**   
   (ref: <http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html>)



1. **Run WordCount on the Hadoop cluster with 2 VMs, what are the top 5 most frequent word in the provided txt file?**

The

Of

I

And

In

1. **Use jps commands on both VMs to show running Hadoop daemons and provide and screenshots.**
2. **Screenshots of configuration files and IP addresses for Master node and Slave node of your small cluster as well as the MapReduce execution result. For each configuration file, please also briefly explain what it does.**

**Changes:**

1. Update core-site.xml  
     
   *Update this file by changing hostname from localhost to HadoopMaster*

## To edit file, fire the below given command

hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$ sudo gedit core-site.xml

## Paste these lines into <configuration> tag OR Just update it by replacing localhost with master

<property>

<name>fs.default.name</name>

<value>hdfs://HadoopMaster:9000</value>

</property>

1. Update hdfs-site.xml  
     
   *Update this file by updating repliction factor from 1 to 3.*

## To edit file, fire the below given command

hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$ sudo gedit hdfs-site.xml

## Paste/Update these lines into <configuration> tag

<property>

<name>dfs.replication</name>

<value>3</value>

</property>

Update yarn-site.xml

*Update this file by updating the following three properties by updating hostname from localhost to HadoopMaster,*

## To edit file, fire the below given command

hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$ sudo gedit yarn-site.xml

## Paste/Update these lines into <configuration> tag

<property>

<name>yarn.resourcemanager.resource-tracker.address</name>

<value>HadoopMaster:8025</value>

</property>

<property>

<name>yarn.resourcemanager.scheduler.address</name>

<value>HadoopMaster:8035</value>

</property>

<property>

<name>yarn.resourcemanager.address</name>

<value>HadoopMaster:8050</value>

</property>

1. Update Mapred-site.xml  
   *Update this file by updating and adding following properties,*

## To edit file, fire the below given command

hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$ sudo gedit mapred-site.xml

## Paste/Update these lines into <configuration> tag

<property>

<name>mapreduce.job.tracker</name>

<value>HadoopMaster:5431</value>

</property>

<property>

<name>mapred.framework.name</name>

<value>yarn</value>

</property>

1. Update masters  
     
   *Update the directory of master nodes of Hadoop cluster*

## To edit file, fire the below given command

hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$ sudo gedit masters

## Add name of master nodes

HadoopMaster

1. Update slaves  
     
   *Update the directory of slave nodes of Hadoop cluster*

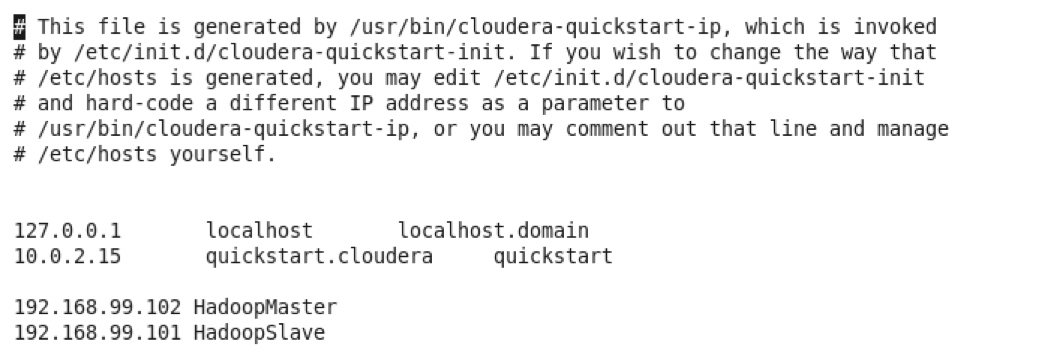
## To edit file, fire the below given command

hduser@HadoopMaster:/usr/local/hadoop/etc/hadoop$ sudo gedit slaves

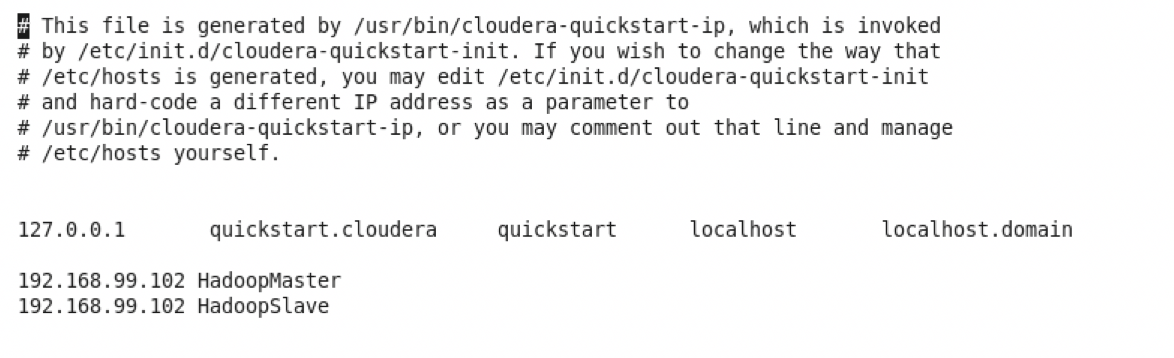
## Add name of slave nodes

HadoopSlave

Master

****

Slave



1. **What are the differences between Hadoop master and slave nodes? Also name what functionalities are performed on each node.**

The Master nodes oversee the two key functional pieces that make up Hadoop: storing lots of data (HDFS), and running parallel computations on all that data (Map Reduce). Slave Nodes make up the majority of machines and performs the function of storing the data and running the computations.

1. **Write a pseudo code to multiply large matrices using Hadoop. Also explain the function of your Mappers and Reducers.**

Let A and B be two large matrices of order m x n and n x p respectively

Then the AB matrix will have order of m x p

Input Format for Hadoop:

The input file in Hadoop is a single line which has the format

<M><i><j><m\_ij>

Where

M is the matrix which needs to be mapped

i is the number of rows of the matrix with increasing order and

j is the number of columns of the matrix in the increasing order

The input file that represents A and B has the following lines:

The formatted input with M = A, i = 0 to m and j = 0 to n

 A,0,1,1.0

 A,0,2,2.0

 A,0,3,3.0

 A,0,4,4.0

. . . .

. . . .

 A,1,0,5.0

 A,1,1,6.0

 A,1,2,7.0

 A,1,3,8.0

 A,1,4,9.0

 So On

 B,0,1,1.0

 B,0,2,2.0

 So On

 B,1,0,3.0

 B,1,1,4.0

 B,1,2,5.0

 So On

 B,2,0,6.0

 B,2,1,7.0

 B,2,2,8.0

So On

 B,3,0,9.0

 B,3,1,10.0

 B,3,2,11.0

 So On

 B,4,0,12.0

 B,4,1,13.0

 B,4,2,14.0

 So On

The following is the pseudo code for Hadoop:

map(key, value):

// Here the value is ("A", i, j, a\_ij) or ("B", j, k, b\_jk) which are the

details of two matrices which mapper can choose

if value[0] == "A":

i = value[1]

j = value[2]

a\_ij = value[3]

// The matrix Input function stored in the hadoop has the Value [0] as

A or B, Value[1] as i , Value[2] is j and Value[3] as a\_jk or b\_jk

depending on the Value[0]

for k = 1 to p:

emit((i, k), (A, j, a\_ij))

// emit is like the print function in the Java which will emit if the

Value[0] is A, say Apple, so it will emit all the Apple mapped data to

the mapper function

else:

j = value[1]

k = value[2]

b\_jk = value[3]

for i = 1 to m:

emit((i, k), (B, j, b\_jk))

//It will map A matrix function or B matrix function depends on the

mapper result.

reduce(key, values):

// The values is a list of ("A", j, a\_ij) and ("B", j, b\_jk) matrix values

which is in the form of Input format discussed above

hash\_A = {j: a\_ij for (x, j, a\_ij) in values if x == A}

hash\_B = {j: b\_jk for (x, j, b\_jk) in values if x == B}

//defining hash\_A and B so that we can multiply and add the values

with ease and can conclude as a result.

result = 0

for j = 1 to n:

result += hash\_A[j] \* hash\_B[j]

//reduce function gives the result of adding the values from the matrix

multiplication, and gives the final result as a number

emit(key, result)

//finally the result is emitted.

//This is how two large matrices are multiplied using Hadoop

 Mapper stage:

It is the first stage in the process which splits out each word into a separate string (i.e. tokenizing the string) and for each word seen, it will output the word and a 1 (which is the count value) to indicate that it has seen the word one time.

 Reducer phase:

The reduce phase will then sum up the number of times each word was seen and write that sum count together with the word as output.

1. **What is a combiner? Add a combiner to the last question and explain its function.**

A Combiner is also known as a semi-reducer. It is an optional class which operates by accepting

the inputs from the Mapping class and by passing the output value pairs to the Reducer class. The main

function of a Combiner is to summarize the map output which records with the same key. The Combiner

class is used in between the Map class and the Reduce class which is useful to reduce the volume of data

transfer between Map and Reduce.

**We have zero tolerance to forged or fabricated data!!** A single piece of forged/fabricated data would bring the total score down to zero.