Lab 4 Report: AWS and Hadoop/MapReduce

Please **fill in the report** and submit the **pdf** to NYUClasses

# Objectives

* A peek of the Amazon AWS and managing VMs on cloud
* Understand the MapReduce concept.
* Get familiar with the Hadoop framework.
* Experience working a small Hadoop cluster with VMs.

# Experiments Tasks

## 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>

1. Play with Amazon Web Services (AWS) EC2 and learn how to create instances on AWS

<https://aws.amazon.com/>

## Install Hadoop

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. Compile the given WordCount java program and run it with MapReduce

## Build a Hadoop Cluster (Bonus)

1. Use two or more VMs to build a Hadoop Cluster with a master node and slave nodes.
2. Compile the given WordCount java program and run it with MapReduce

# Reports

1. When creating the AWS account, you can choose IAM user or Root user. What privilege does Root user have over IAM user? Which one will you choose and why?

|  |
| --- |
| For the Root users, it can allow account owner have right to access all resources in the account. Also we can only use the AWS Organization service control policy to limit the permission of the Root users. However, for the IAM user. WE may not have right to access all resources in the account. We can securely control the AWS services and also control the resources which is belong to the users in your AWS account.  I prefer to choose IAM user. Because for the root user it can only be limited by the AWS Organizations SCP. So we can create IAM user account with administrator permission for AWS tasks, and we can lock away the access keys of the root users. According to this way, it can be more security. Also for the IAM users, we can set credential for everyone who want to access our resources. For example, the read only. So other users will not have right to edit your resources. |

1. What are the major differences between Hadoop version 1, 2, and 3?

|  |
| --- |
| In Hadoop 1we have two major components which are MapReduce and HDFS. However, in Hadoop 2, we have three major components which are MapReduce, HDFS and YARN. As we know in Hadoop 1 MapReduce does both bath processing and Cluster management. But in Hadoop 2, Yarn does the resource management.  For the Hadoop 3, the major difference is on the utilization. For Hadoop 2 it has 3 times of replication scheme which is used. However, for Hadoop 3, it uses the eraser encoding which is in the HDFS. For the storage overhead, Hadoop 2 can consume 200 percent of HDFS. However, Hadoop 3 only consume 50 percent of HDFS. And for the compatible, Hadoop2 can be used really limited file system. However, Hadoop 3 can be used in all files systems which includes the Microsoft Azure Data Lake filesystem. Also Hadoop 2 uses java 7, and Hadoop 3 uses java 8. |

1. What is YARN? What are advantages of YARN over MapReduce?

|  |
| --- |
| YARN is Yet Another Resource Negotiator. MapReduce is a programming model and YARN is a resource management architecture. YARN is used to separate the processing component and resource management process. So we can say YARN is used for a large scale, distributed operating system. And the target of it is to deal with the big data application. It is the major difference compare Hadoop 2.0 to Hadoop 1.0  For the MapReduce framework. We know MapReduce includes two parts, one is Map and the other is Reduce. So in the MapReduce job we need to divide work into mappers and reducers. So we will have some map slots and reduce slots. They are predefined, so it might be one of the slots is full and the other slots are empty, but because the slots has been predefined so we cannot use it whatever we want. This may cause some problem of utilization. Also, because MapReduce is not an architecture, so it will have problem when running on some applications with not MapReduce, it can only support its own batch processing applications. The scalability is also limited because it needs the JobTracker to perform lots of tasks on single machine. However for the other machine this will not be used.  However, for the YARN, it is an architecture for the resource management. So it takes the cluster management out of the MapReduce. It has a central resource manager which is used to manage resources and then allocate resources to the applications. So it will not have some predefined slots, we will not care about the utilization problem. YARN provides a central resource manager as I mentioned before and this manager will allow you to share lots of applications with a common resource. According to this point, YARN has better utilization and efficiency of the resources than MapReduce does. Also we can run the application that do not have MapReduce model. In YARN we do not need the JobTracker. We know the major tasks for the JobTracker is resource management and job scheduling. And in YARN we have NodeManager and ResourceManager to deal with this two major tasks, so the scalability will not be limited. |

1. What is Spark? What is the difference between Spark and Hadoop?

|  |
| --- |
| Spark is also used for big data processing. Spark is an open-source unified analytics engine for a big data processing. It is focusing on processing data in parallel across a cluster.  For the Hadoop, it reads and writes files to HDFS. However, Spark will process data in RAM. Also we know that when spark process data it will use a concept named RDD (Resilient Distributed Dataset). |

1. What is Hadoop streaming? If I have a Mapper written in python, how can I use Hadoop Streaming to run the code with Hadoop?

|  |
| --- |
| Hadoop streaming is a utility which comes with the Hadoop distribution. Hadoop Streaming allows to create the MapReduce jobs with any executable or script as mapper and / or the reducer.  $HADOOP\_HOME/bin/Hadoop jar $HADOOP\_HOME/Hadoop-streaming.jar \  -input myInputDirs \  -outpit myOutputDir \  -mapper Mapper.py  -reducer /bin/wc  -file Mapper.py |

1. Screenshots of the practice of Hadoop (in 2.2) on your own computer. The screenshots should show the result of Hadoop execution ($ cat output/\*) as well as the files in HDFS ($ hdfs dfs -ls).

|  |
| --- |
|  |

1. Run WordCount on the Hadoop. Please attach the screenshots of your result, and what are the top 5 most frequent word in the provided txt file?

|  |
| --- |
| 5 top frequency in file: |

1. In Hadoop Cluster Mode, what are the differences between Hadoop master and slave nodes? Also name what functionalities are performed on each node.

|  |
| --- |
| **Master Node** is used to store data in the HDFS. And we know the data which is stored in HDFS will be parallel computation via MapReduce. And this process will be executed in master node. Also Mster Node has three node named NameNode, Secondary NameNode and JobTracker. For the Secondary NameNode is used for backup. So we can say Master node is responsible for storing data which means that Master node can control the metadata. And we can use metadata to access the data.  For the **slave node**, it is the actual location of the data which means that Slave node is where the data in Hadoop can be stored. It is also where the data processing can take place. So we can say that Master Node controls the Slave Node.  Also we can say that Master Node can also be called the Name Node. In a non-high available cluster, there can only be one NameNode. So the Master Node is the single NameNode for managing the metadata.  And we can also call the Slave Nodes as Multiple DataNodes. They are used for storing data. |

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

|  |
| --- |
| first we have two matrix. One is Matrix M and the other is matrix N. If we do the multiplication we will get Matrix MxN which is Matrix MN. For the multiplication of matrix we will do the multiply of each element in the row of M with each element in the column of N. So we can define M is axb. And N is bxk. From this point we can get MN with axk. However, the number of the elements is determined by n in M and b in N.  **Mapper:**  **For element in M:**  **(key, value) -> ((a,k),(M,b,Mab))**  **K ->1,2,3,4,5…..bN**(which means that k should up to the number of columns of N, because we need to use row number of M multiple the column number of N)  For element in N:  **(key, value) -> ((a,k),(N,b,Nbk))**  **K -> 1,2,3,4,5…bM**(which means that k should up to the number of row of M, because we need to. use column number of N multiple the row number of M)  **Return (key,value)set**. (In the(key,value) set we will have (a,k) as key, and we will have (M,b,Mab) and. (N,b,Nbk) for value of b.)  **Reducer:**  **For key in (a,k):**  **Sort each values in listM begin with M by b**  **Sort each values in listN begin with N by b**  **Do the multiplication Mab x Nbk for each b value**  **Sum each Mab x Nbk as a new result in MN**  **Return ((i,k),(sum(Mab,Nbk)))** |
|  |

1. What is a “combiner” in MapReduce?

|  |
| --- |
| As we know that for MapReduce we have two stage. One is Mao stage and the other one is reduce stage. In the Map Stage we have the input data which should be stored in the HDFS. And this data is generated line by line according to the mapper function. And then we will do the reduce stage. In the reduce stage we will have the combination which is the combiner in the MapReduce. And then after the combination we will do the reducer to get the output.  So we can also call the combiner as the Semi-Reducer. It is used to summarize the output outs from the Mapper with the same Key. And then we pass this “output” to the Reducer. |

1. Try to add a combiner to the question (i) and **explain its function**. Justify why adding the combiner can make your code faster.

|  |
| --- |
| Job.setCombinerClass(reducer.class) can add a combiner to the question(i).  According to the official document we have setCombinerClass(Class<? extends Reducer> cls) so we just create a Job configuration and then call this set combiner to set combiner to the reducer class.  We know the combiner can be optionally used in the code. And from the frame, the combiner works after the mapper and before the reducer. So we know that during the reducer it has a process called the combine. So combiner her is used to help local aggregation of the intermediate outputs from mapper. It can help to cut down the transfer of data after the mapper and before the reduce. Because the data transfer has been cut down. So we can make our code faster. |

**(Bonus) Hadoop Cluster Mode**

1. Use jps commands on both VMs to show running Hadoop daemons and provide and screenshots.

|  |
| --- |
|  |
|  |

1. 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.

|  |
| --- |
| Master: |
|  |
| Slave1: |
|  |
|  |
|  |
| Slave2:        In hadoop we need to edit fotr xml files which is core-site.xml, hdfs-site.xml, mapred-site.xml and yearn-site.xml  For the core-site.xml we use this to defice the parameter of the system such as HDFS URl, temperory directory of Hadoop and so on. For the fs.defaultFS, we use this to define the address of hdfs protocol of the namenode. So we can add one host with port number. Also we can also assign one namenode service here. For thr io.file.buffer.szie executes the size of the IO buffer area.  For hdfs-site.xml file which we set the HDFS parameter. Like the address of data. Number of copy of files and the permission to read the file and so on. For the dfs.namenode .name.dir we set the secondary namenode http address. For the dfs.blocksize, we know when we upload a file, it will be cut in blocks and here it set the blocksize. For dfs.namenode.handler.count we set the handler count to the namenode so we can deal with the larger number cluster and larger number of clients. And for dfs.replication, this is the setting of the copy of HDFS. It is the number of the redundancy copy of the blocks.  For mapred-site.xml we set the MapReduce parameter.so We have two parts which should be application and the JobHistory Server. Like the number of elemtns in reducer, and the limitation of the default memory. We only have mapreduce.framework.name here to assign the frame of the mapreduce is yarn.  For yarn-site.xml. this fule is used for managing the yarn system parameter. Which we used to set the resourceManager, NodeManager. Yarn.acl.enable we set true here to enable the YARN ACL. And for Yarn.admin.acl we set the YARN ACL type which is admin. So we will have YARN Admin ACL access. For Yarn.nodemanager.aux-services we can set some services here like the shuffle of the Map-Reduce. So we can extend our own service on NodeManager. And for yarn.application.classpath we set the classpath to the yarn. |

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