# Lab 4 Report: AWS and Hadoop/MapReduce

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

Name: _____ ID: _____ Date: _____

## 1. 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.

## 2. Experiments Tasks

### 2.1 Basics

a) Go through the Apache Hadoop introduction to get the general idea about Hadoop: http://hadoop.apache.org/
b) Go through the Apache Hadoop release notes to understand the evolution of Hadoop: http://hadoop.apache.org/releases.html
c) Play with Amazon Web Services (AWS) EC2 and learn how to create instances on AWS https://aws.amazon.com/

### 2.2 Install Hadoop

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

### 2.3 Build a Hadoop Cluster (Bonus)

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

## 3. Reports

(a) 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?

Root user is for management(creation/deletion/etc) of other IAM users. We should use IAM for daily tasks and root for managing users.
Ref:
https://docs.aws.amazon.com/IAM/latest/UserGuide/id_root-user.html

(b) What are the major differences between Hadoop version 1, 2, and 3?

- 1.x
  - MapReduce
    - batch processing
    - Cluster management
  - HdFS
  - only single Namenode
  - doesnot support Microsoft windows
- 2.x
  - MapReduce
    - batch processing
  - HdFS
  - YARN
    - cluster management
  - multi NameNode
  - support Microsoft windows
  - data balancing uses HDFS
  - We can scale up to 10,000 Nodes per cluster.
- 3.x
  - data balancing uses Intra-data node balancer, which is invoked via the HDFS disk balancer CLI.
  - It supports as well as Microsoft Azure Data Lake filesystem.
  - We can scale up to >10,000 Nodes per cluster.

Ref:
https://thebigan.wordpress.com/2017/09/11/differences-between-hadoop-1-x-2-x-and-3-x/

(c) What is YARN? What are advantages of YARN over MapReduce?

YARN (Yet Another Resource Negotiator)

In 1.x MapReduce do both batch processing and Cluster management.
In 2.x YARN takes over the Cluster management part.

- Yarn does efficient utilization of the resource and can run multiple applications in

Hadoop
  - In Hadoop 1.0, MapReduce predefines number of map slots and reduce slots for each TaskTrackers. Resource Utilization issues occur because maps slots might be 'full' while reduce slots is empty (and vice-versa).
- Yarn can even run application that do not follow MapReduce model.
  - Since resource management and data processing are departed in 2.x with YARN, new 2.x can support varied processing approaches.

Ref:
https://bigishere.wordpress.com/2016/06/22/how-yarn-overcomes-mapreduce-limitations-in-hadoop-2-0/
https://www.techopedia.com/2/31276/trends/big-data/what-are-the-advantages-of-the-hadoop-20-yarn-framework

(d) What is Spark? What is the difference between Spark and Hadoop?

Ref:
https://logz.io/blog/hadoop-vs-spark/

Spark is a Apache project, focusing on processing data in parallel across a cluster, using in-memory tech.

Hadoop contains HDFS as file system and other components for data processing(MapReducspae) and recourse management (MapReduce or YARN, depends on hadoop version)

Spark do the data processing part, while we choose other components of Hadoop to do recourse management and file system management.

Hadoop reads and writes files to HDFS, which is composed by hard disks. Spark processes data in RAM using a concept known as an RDD, Resilient Distributed Dataset.

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

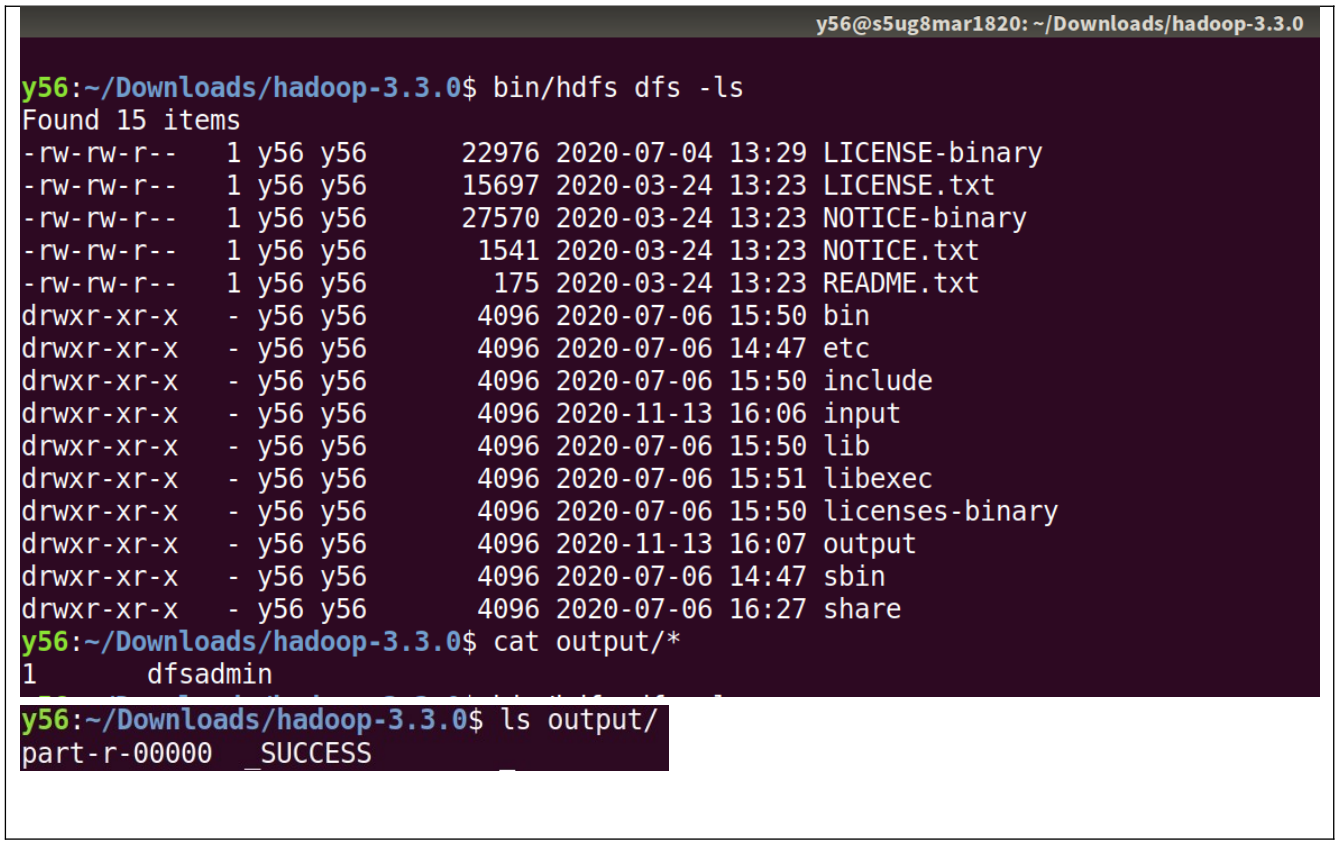
Ref:
https://www.tutorialspoint.com/hadoop/hadoop_streaming.htm

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

Ref:
https://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

We will use the Hadoop Streaming API . We will simply use Python's sys.stdin to read input data and print our own output to sys.stdout.
We also need to have a Reducer following the standard input/output.
We also need to put out data into HDFS.

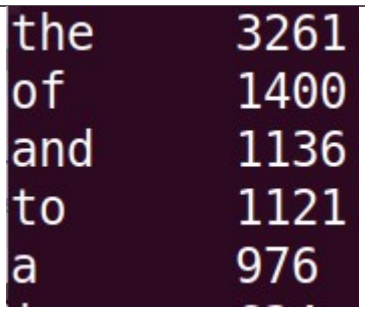
(f) Screenshots of the practice of Hadoop (in 2.2) on your own computer. The screenshots should show the result of Hadoop execution ($ cat output/*) as well as the files in HDFS ($ hdfs dfs -ls).




(g) 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?

```
y56:~/Downloads/hadoop-3.3.0$ bin/hadoop jar wc.jar WordCount input output
2020-11-24 20:01:16,576 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2020-11-24 20:01:16,633 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2020-11-24 20:01:16,633 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2020-11-24 20:01:16,681 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Imp
terface and execute your application with ToolRunner to remedy this.
2020-11-24 20:01:16,711 INFO input.FileInputFormat: Total input files to process : 11
2020-11-24 20:01:16,726 INFO mapreduce.JobSubmitter: number of splits:11
2020-11-24 20:01:16,824 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local692958862_0001
2020-11-24 20:01:16,824 INFO mapreduce.JobSubmitter: Executing with tokens: []
2020-11-24 20:01:16,924 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2020-11-24 20:01:16,925 INFO mapreduce.Job: Running job: job_local692958862_0001
2020-11-24 20:01:16,926 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2020-11-24 20:01:16,931 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
```

```
                Map output bytes=469844
                Map output materialized bytes=138626
                Input split bytes=1365
                Combine input records=47980
                Combine output records=9535
                Reduce input groups=8669
                Reduce shuffle bytes=138626
                Reduce input records=9535
                Reduce output records=8669
                Spilled Records=19070
                Shuffled Maps =11
                Failed Shuffles=0
                Merged Map outputs=11
                GC time elapsed (ms)=76
                Total committed heap usage (bytes)=5840568320
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=293653
        File Output Format Counters
                Bytes Written=93888
y56:~/Downloads/hadoop-3.3.0$ cat output/*
"*"     22
"AS     10
"Defects,"      1
"Information    1
"License");     10
"Plain  2
"Project        5
"Right  1
```




cat part-r-00000 | sort -t $'\t' -k 2rn | less

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

Ref:
https://findanyanswer.com/what-is-master-node-and-slave-node-in-hadoop

Master(NameNode) manages jobs and slaves(DataNode) run the jobs.

Master node in a hadoop cluster is responsible for storing data in HDFS and executing parallel computation the stored data using MapReduce. Master Node has 3 nodes – NameNode, Secondary NameNode and JobTracker.

Slave nodes are where Hadoop data is stored and where data processing takes place. The following services enable slave nodes to store and process data: NodeManager: Coordinates the resources for an individual slave node and reports back to the Resource Manager.

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

We want NM, where M and N are matrices
M is in dimension of row_M by col_M.
N is in dimension of row_N by col_N.

**Mapper:**

- for each m_ij in M:
  - produce (key,val) pair as ((i,k),(M,j,m_ij)) for k = 1 ~~ col_N
- for each n_jk in N:
  - produce (key,val) pair as ((i,k),(N,j,n_ij)) for k = 1 ~~ row_M
- return set of (key.val) pairs that each key, ie,(i,k), has a list containing (M,j,m_ij) and (N,j,n_ij) for all possible j

```
// key: (i,k)
// val: [ (M,j,m_ij) , (N,j,n_ij) ]
```

```
// so we are listing all numbers we need when calculate elements of the
resulted matrix, i.e.,
```

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \begin{pmatrix} 6 & 3 \\ 5 & 2 \\ 4 & 1 \end{pmatrix} = \begin{pmatrix} 1*6+2*5+3*4 & 1*3+2*2+3*1 \\ 4*6+5*5+6*4 & 4*3+5*2+6*1 \end{pmatrix}$$

**Reducer:**

- for each key (i,k):
  - sort values begin w/ M by j in list_M
  - sort valuse begin w/ N by j in list_N
    // sort both by j to ensure m_ij and n_jk will be multiplied
    // since the summation formula for element of the product is

$$P_{(i,k)} = \sum_{j=1} m_{ij} * n_{jk}$$

  - multiply m_ij and n_jk for j-th val of each list
  - sum up all m_ij * n_jk
- return (i,k), \sum_{j=1} m_ij * n_jk

http://www.mathcs.emory.edu/~cheung/Courses/554/Syllabus/9-parallel/matrix-mult.html
https://www.geeksforgeeks.org/matrix-multiplication-with-1-mapreduce-step/
https://lendap.wordpress.com/2015/02/16/matrix-multiplication-with-mapreduce/

(j) What is a “combiner” in MapReduce?

The Combiner class is used in between the Map class and the Reduce class to reduce the volume of data transfer between Map and Reduce. Usually, the output of the map task is large and the data transferred to the reduce task is high.

https://www.tutorialspoint.com/map_reduce/map_reduce_combiners.htm

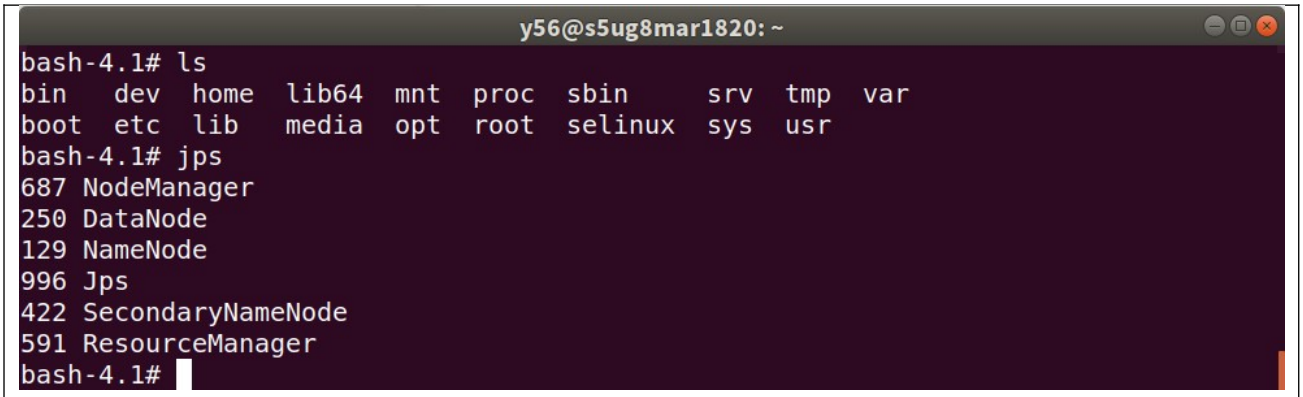
(k) Try to add a combiner to the question (i) and **explain its function**. Justify why adding the combiner can make your code faster.

In the pseudo code we let `m_ij * n_jk` to be computed by each reducer. If the matrix are large, the output of mapper will use large space. We can add combiners between mappers and reducers. For example, we can divide M and N into many 2by2 blocks and we can let the combiners do many 2by2*2by2 first; and then let reducer do the remaining multiplication. Once a reducer is processing a smaller amount of data, it is more likely that it can do the computation with more benefit from CPU cache. So it can be faster

https://www.geeksforgeeks.org/strassens-matrix-multiplication/
http://www.joefkelley.com/853/

**(Bonus) Hadoop Cluster Mode**

(a) Use jps commands on both VMs to show running Hadoop daemons and provide and screenshots.




(b) 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.

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