

# BIG DATA AND BUSINESS ANALYTICS

## LAB MANUAL

Academic Year	:	2018 - 2019
Course Code	:	ACS111
Regulations	:	IARE - R16
Semester	:	VII
Branch	:	CSE

Prepared by

**Ms. S SWARAJYA LAXMI, ASSISTANT PROFESSOR**

**Ms. E UMA SHANKARI, ASSISTANT PROFESSOR**

**Ms. G SULAKSHANA, ASSISTANT PROFESSOR**

**Ms. G SREE LEKHA, ASSISTANT PROFESSOR**



**INSTITUTE OF AERONAUTICAL ENGINEERING**

(Autonomous)

Dundigal, Hyderabad - 500 043



# INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

## 1. PROGRAM OUTCOMES:

PROGRAM OUTCOMES (POs)	
<b>PO-1:</b>	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems ( <b>Engineering Knowledge</b> ).
<b>PO-2:</b>	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences ( <b>Problem Analysis</b> ).
<b>PO-3:</b>	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations ( <b>Design/Development of Solutions</b> ).
<b>PO-4:</b>	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions ( <b>Conduct Investigations of Complex Problems</b> ).
<b>PO-5:</b>	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations ( <b>Modern Tool Usage</b> ).
<b>PO-6:</b>	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice ( <b>The Engineer and Society</b> ).
<b>PO-7:</b>	Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development ( <b>Environment and Sustainability</b> ).
<b>PO-8:</b>	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice ( <b>Ethics</b> ).
<b>PO-9:</b>	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings ( <b>Individual and Team Work</b> ).
<b>PO-10:</b>	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions ( <b>Communication</b> ).
<b>PO-11:</b>	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
<b>PO-12:</b>	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change ( <b>Life-long learning</b> ).

## 2. PROGRAM SPECIFIC OUTCOMES (PSOs):

**PROGRAM SPECIFIC OUTCOMES (PSO'S)**

<b>PSO – I</b>	<b>Professional Skills:</b> The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient design of computer-based systems of varying complexity.
<b>PSO – II</b>	<b>Problem-Solving Skills:</b> The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.
<b>PSO – III</b>	<b>Successful Career and Entrepreneurship:</b> The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies.

### **3. ATTAINMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:**

<b>WEEK.N O</b>	<b>Experiment</b>	<b>Program Outcomes Attained</b>	<b>Program Specific Outcomes Attained</b>
1.	INSTALL VMWARE	PO1, PO2, PO5	PSO2
2.	HADOOP MODES	PO1, PO2, PO5	PSO2
3.	USING LINUX OPERATING SYSTEM	PO1, PO2, PO5	PSO2
4.	FILE MANAGEMENT IN HADOOP	PO1, PO2, PO5	PSO2
5.	MAPREDUCE PROGRAM 1	PO1, PO2, PO5	PSO2
6.	MAPREDUCE PROGRAM 2	PO1, PO2, PO3, PO4, PO12	PSO1, PSO2
7.	MAPREDUCE PROGRAM 3	PO1, PO2, PO3, PO4, PO5, PO12	PSO1, PSO2
8.	PIG LATIN LANGUAGE - PIG	PO1, PO2, PO3, PO4, PO5, PO12	PSO1, PSO2
9.	PIG COMMANDS	PO1, PO2, PO3, PO4, PO5, PO12	PSO1, PSO2
10.	PIG LATIN MODES, PROGRAMS	PO1, PO2, PO3, PO4, PO5, PO12	PSO1, PSO2
11.	HIVE	PO1, PO2, PO3, PO4, PO5, PO12	PSO1, PSO2
12.	HIVE OPERATIONS	PO1, PO2, PO3, PO4, PO5, PO12	PSO1, PSO2

**4. MAPPING COURSE OBJECTIVES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES:**

Course Objectives (COs)	Program Outcomes (POs)												Program Specific Outcomes (PSOs)		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
I	√	√	√										√		
II	√	√		√	√								√		
III			√		√							√	√	√	
IV	√	√	√									√		√	

## 5. SYLLABUS:

## BIG DATA AND BUSINESS ANALYTICS LABORATORY

### VII Semester: CSE/IT

Course Code	Category	Hours / Week			Credits	Maximum Marks		
		L	T	P		C	CIA	SEE
ACS111	Core	-	-	3	2	30	70	100
<b>Contact Classes: Nil</b>	<b>Tutorial Classes: Nil</b>	<b>Practical Classes:36</b>			<b>Total Classes: 36</b>			

#### COURSE OBJECTIVES:

##### The course should enable the students to:

- I. Optimize business decisions and create competitive advantage with Big data analytics
- II. Practice java concepts required for developing map reduce programs.
- III. Impart the architectural concepts of Hadoop and introducing map reduce paradigm.
- IV. Practice programming tools PIG and HIVE in Hadoop eco system.
- V. Implement best practices for Hadoop development.

#### COURSE LEARNING OUTCOMES (CLOs):

1. Understand the installation of VMWare
2. Understand and apply the Perform setting up and Installing Hadoop in its three operating modes.
3. Implementing the basic commands of LINUX Operating System
4. Implement the file management tasks in Hadoop.
5. Understand Map Reduce Paradigm.
6. Apply Map Reduce program that mines weather data.
7. Implement matrix multiplication with Hadoop MapReduce
8. Apply Map Reduce program that makes the dataset to be compressed.
9. Understand the installation of PIG.
10. Understand Pig Latin scripts sort, group, join, project, and filter your data.
11. Implement the Pig Latin scripts in two different modes
12. Understand the installation of HIVE
13. Apply Hive to create, alter, and drop databases, tables, views, functions, and indexes.

#### Week-1

#### INSTALL VMWARE

Installation of VMWare to setup the Hadoop environment and its ecosystems.

#### Week-2

#### HADOOP MODES

- a. Perform setting up and Installing Hadoop in its three operating modes.
  - i. Standalone.
  - ii. Pseudo distributed.
  - iii. Fully distributed.
- b. Use web based tools to monitor your Hadoop setup.

#### Week-3

#### USING LINUX OPERATING SYSTEM

Implementing the basic commands of LINUX Operating System – File/Directory creation, deletion, update operations.

#### Week-4

#### FILE MANAGEMENT IN HADOOP

Implement the following file management tasks in Hadoop:

- i. Adding files and directories
- ii. Retrieving files

iii. Deleting files

Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.

**Week-5**

**MAPREDUCE PROGRAM 1**

Run a basic word count Map Reduce program to understand Map Reduce Paradigm.

**Week-6**

**MAPREDUCE PROGRAM 2**

Write a Map Reduce program that mines weather data. Hint: Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with Map Reduce, since it is semi structured and record-oriented

**Week-7**

**MAPREDUCE PROGRAM 3**

Implement matrix multiplication with Hadoop Map Reduce.

**Week-8**

**PIG LATIN LANGUAGE – PIG**

Installation of PIG.

**Week-9**

**PIG COMMANDS**

Write Pig Latin scripts sort, group, join, project, and filter your data.

**Week-10**

**PIG LATIN MODES, PROGRAMS**

- a. Run the Pig Latin Scripts to find Word Count
- b. Run the Pig Latin Scripts to find a max temp for each and every year.

**Week-11**

**HIVE**

Installation of HIVE.

**Week-12**

**HIVE OPERATIONS**

Use Hive to create, alter, and drop databases, tables, views, functions, and indexes.

**Reference Books:**

Jay Liebowitz, —Big Data And Business Analytics Laboratory, CRC Press.

**6. INDEX:**

<b>S.NO</b>	<b>LIST OF EXPERIMENTS</b>	<b>PAGE NO</b>
<b>WEEK-1</b>	<b>INSTALL VMWARE</b>	

	Installation of VMWare to setup the Hadoop environment and its ecosystems.	10	
<b>WEEK-2</b>	<b>HADOOP MODES</b>		
	a	Perform setting up and Installing Hadoop in its three operating modes. i. Standalone. ii. Pseudo distributed. iii. Fully distributed.	17
	b	Use web based tools to monitor your Hadoop setup.	
<b>WEEK-3</b>	<b>USING LINUX OPERATING SYSTEM</b>		
	a	Implementing the basic commands of LINUX Operating System – File/Directory creation, deletion, update operations.	24
<b>WEEK-4</b>	<b>FILE MANAGEMENT IN HADOOP</b>		
	a	Implement the following file management tasks in Hadoop: i. Adding files and directories ii. Retrieving files iii. Deleting files Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.	26
<b>WEEK-5</b>	<b>MAPREDUCE PROGRAM 1</b>		
		Run a basic word count Map Reduce program to understand Map Reduce Paradigm.	28
<b>WEEK-6</b>	<b>MAPREDUCE PROGRAM 2</b>		
		Write a Map Reduce program that mines weather data. Hint: Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with Map Reduce, since it is semi structured and record-oriented.	31
<b>WEEK-7</b>	<b>MAPREDUCE PROGRAM 3</b>		
		Implement matrix multiplication with Hadoop Map Reduce	34
<b>WEEK-8</b>	<b>PIG LATIN LANGUAGE - PIG</b>		
	a	Installation of PIG.	37
<b>WEEK-9</b>	<b>PIG COMMANDS</b>		
	a	Write Pig Latin scripts sort, group, join, project, and filter your data.	39
<b>WEEK-10</b>	<b>PIG LATIN MODES, PROGRAMS</b>		
	a	Run the Pig Latin Scripts to find Word Count.	41
	b	Run the Pig Latin Scripts to find a max temp for each and every year.	42
<b>WEEK-11</b>	<b>HIVE</b>		



	a	Installation of HIVE.	43
<b>WEEK-12</b>	<b>HIVE OPERATIONS</b>		
	a	Use Hive to create, alter, and drop databases, tables, views, functions, and indexes.	45

## WEEK-1

### INSTALL VMWARE

#### 1.1 OBJECTIVE:

To Install VMWare.

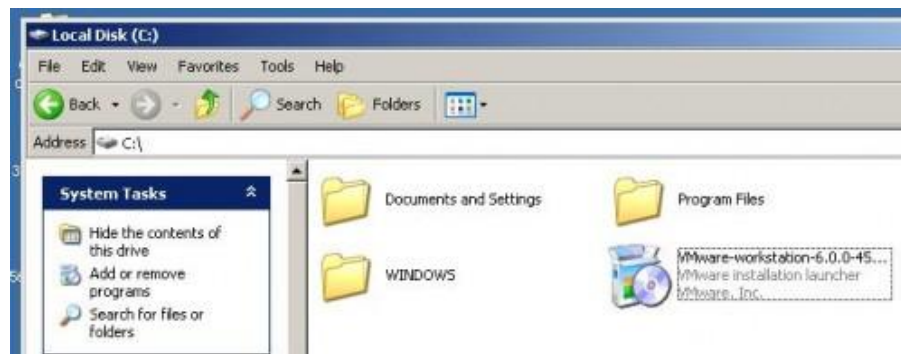
#### 1.2 RESOURCES:

VMWare stack, 4 GB RAM, Web browser, Hard Disk 80 GB.

#### 1.3 PROGRAM LOGIC:

STEP 1. First of all, enter to the official site of VMWare and download VMWare Workstation  
<https://www.vmware.com/tryvmware/?p=workstation-w>

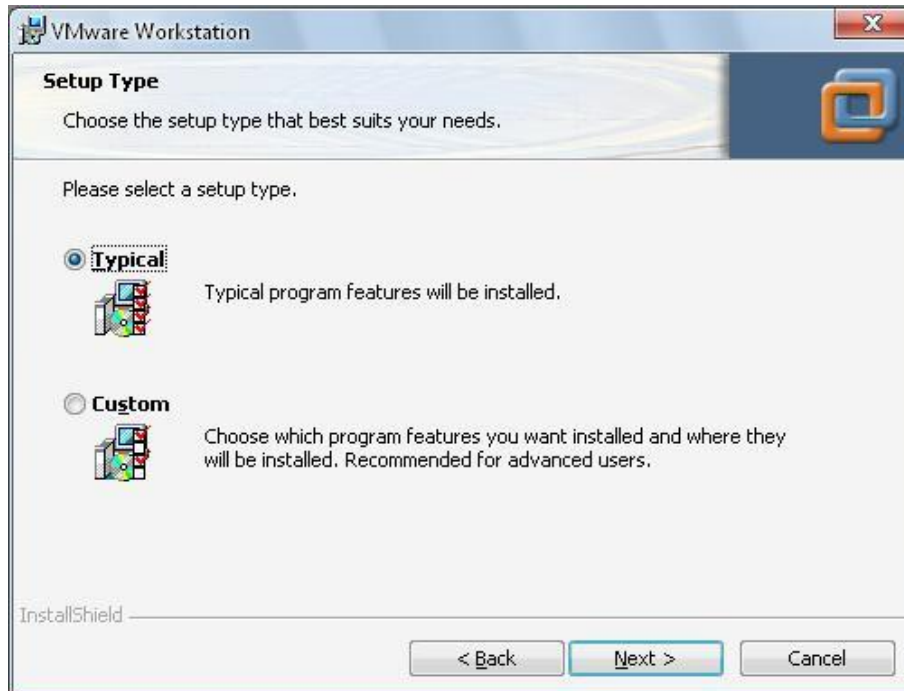
STEP 2. After downloading VMWare workstation, install it on your PC



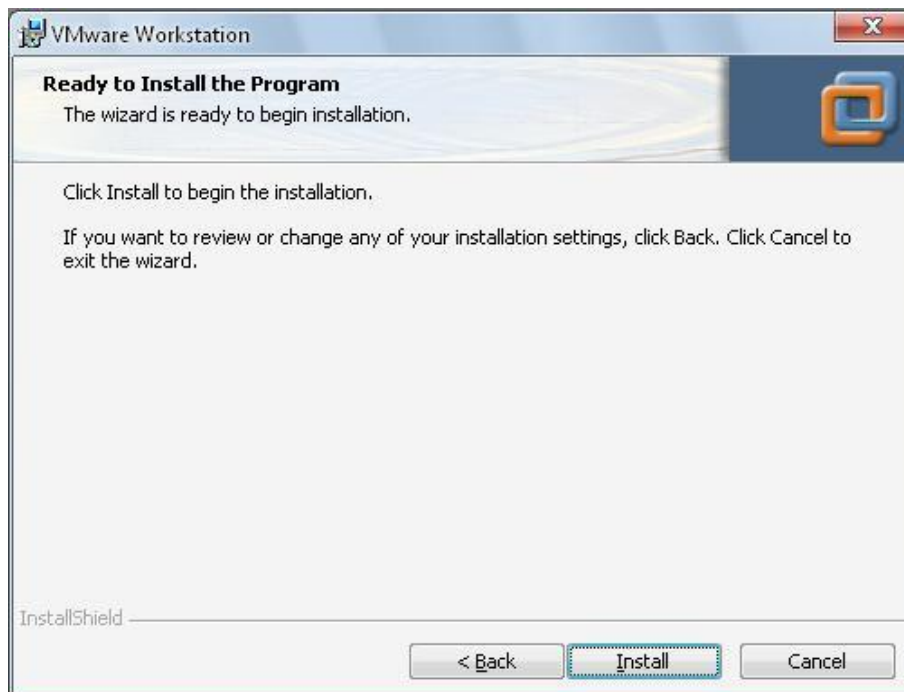
STEP 3. Setup will open Welcome Screen



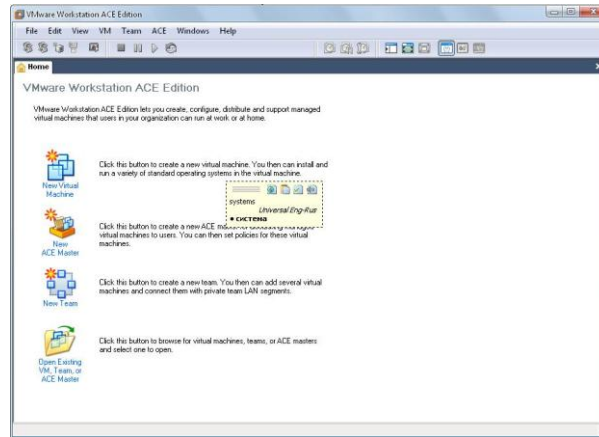
Click on **Next** button and choose **Typical** option



STEP 4. By clicking “Next” buttons, to begin the installation, click on **Install** button at the end

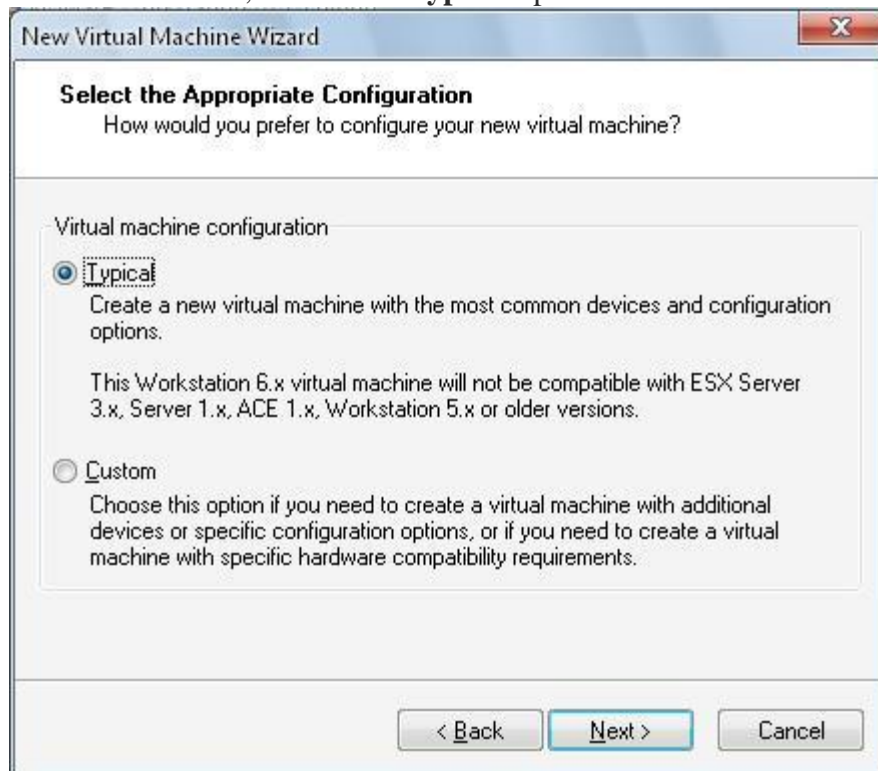


STEP 5. This will install VMware Workstation software on your PC, After installation complete, click on **Finish** button. Then restart your PC. Then open this software



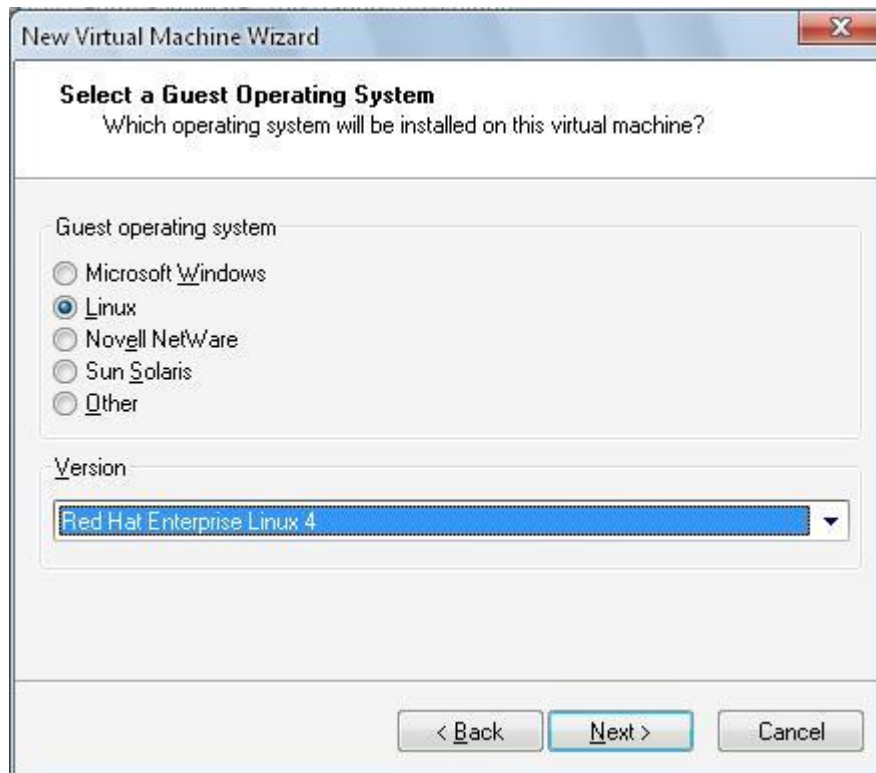
6. In this step we try to create new “virtual machine”. Enter to File menu, then New-> Virtual Machine

Click on **Next** button, then check **Typical** option as below

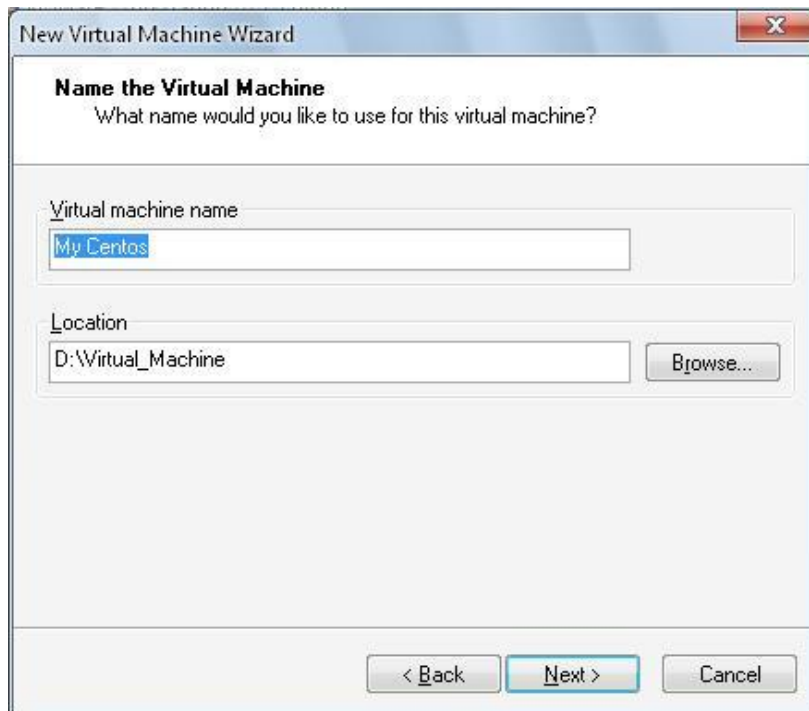


Then click **Next** button, and check your OS version. In this example, as we’re going to setup Oracle server on CentOS, we’ll check **Linux** option and from “*version*” option we’ll check **Red Hat**

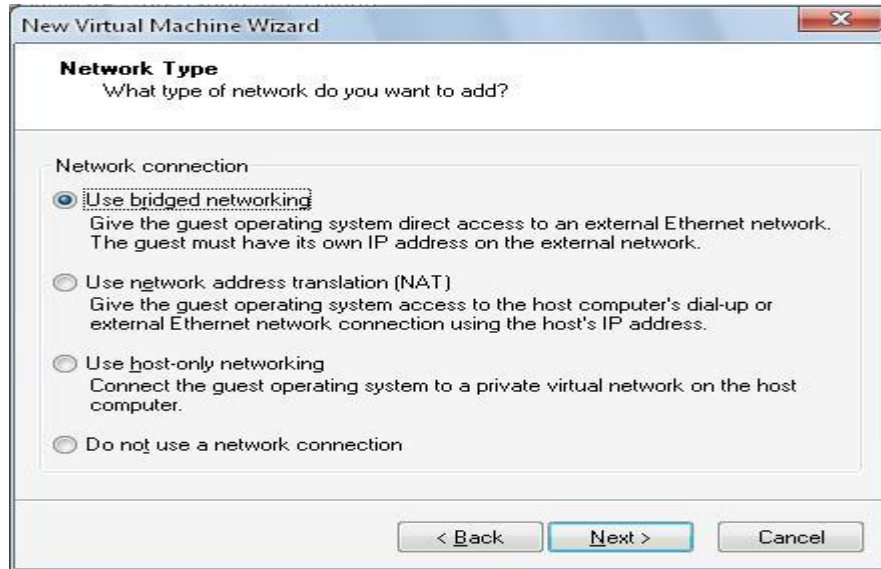
## Enterprise Linux 4



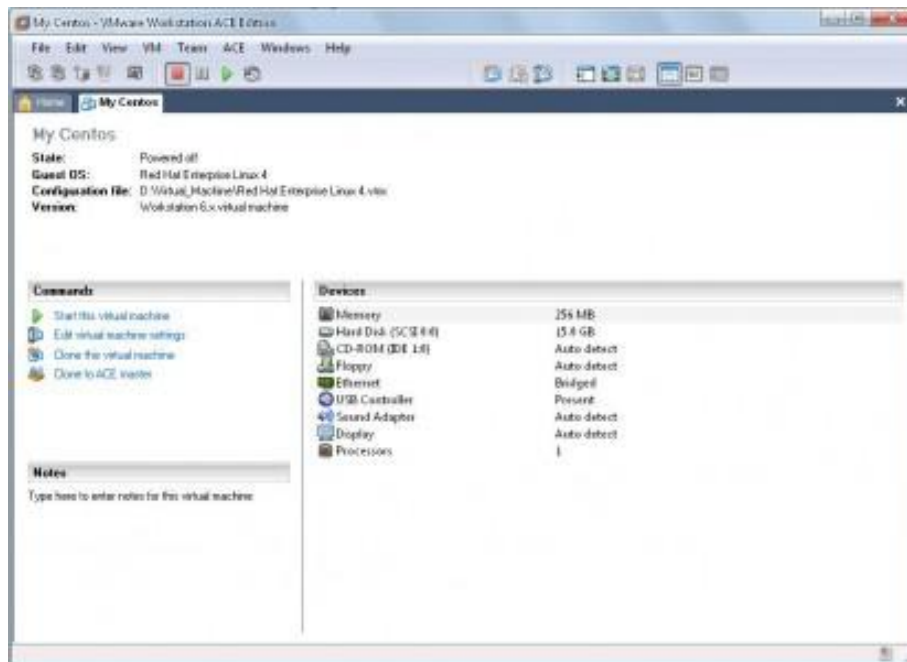
By clicking **Next** button, we'll give a name to our virtual machine, and give directory to create this new virtual machine



Then select **Use bridged networking** option and click **Next**.

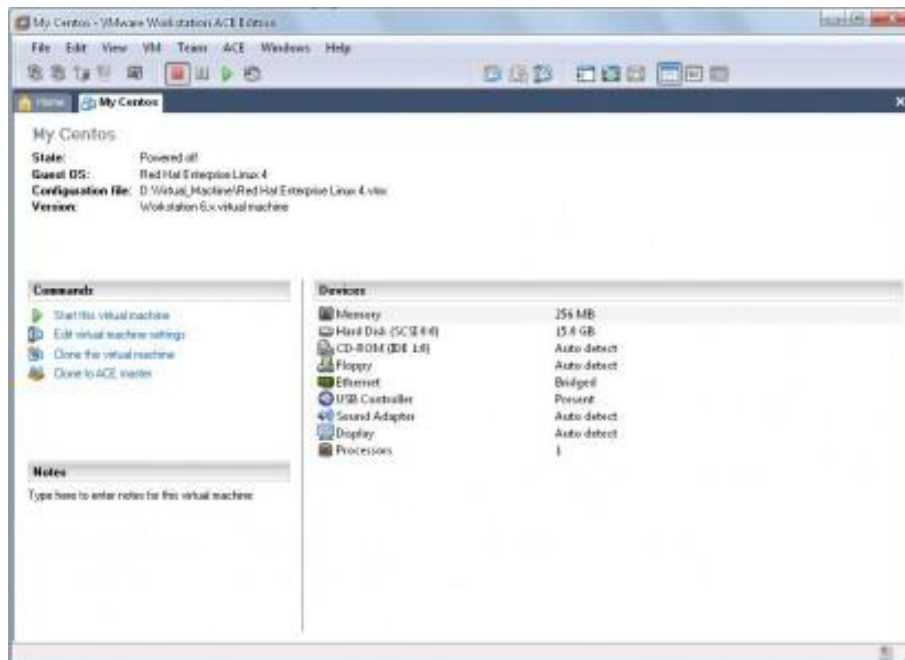


Then you've to define size of hard disk by entering its size. I'll give 15 GB hard disk space and please check **Allocate all disk space now** option



Here, you can delete **Sound Adapter, Floppy and USB Controller** by entering "Edit virtual machine settings". If you're going to setup Oracle Server, please make sure you've increased your Memory (RAM) to 1GB.

## 1.4 INPUT/OUTPUT



## 1.5 PRE LAB VIVA QUESTIONS:

1. What is VMWare stack?
2. List out various data formats?
3. List out the characteristics of big data?

## 1.6 LAB ASSIGNMENT:

1. Install Pig?
2. Install Hive?

## 1.7 POST LAB VIVA QUESTIONS:

1. List out various terminologies in Big Data environments?
2. Define big data analytics?

## WEEK - 2

### HADOOP MODES

#### 2.1 OBJECTIVE:

- 1) Perform setting up and Installing Hadoop in its three operating modes.  
Standalone.  
Pseudo distributed  
Fully distributed.
- 2) Use web based tools to monitor your Hadoop setup.

#### 2.2 RESOURCES:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

#### 2.3 PROGRAM LOGIC:

##### a) STANDALONE MODE:

- Installation of jdk 7

**Command:** sudo apt-get install openjdk-7-jdk

- Download and extract Hadoop

**Command:** wget <http://archive.apache.org/dist/hadoop/core/hadoop-1.2.0/hadoop-1.2.0.tar.gz>

**Command:** tar -xvf hadoop-1.2.0.tar.gz

**Command:** sudo mv hadoop-1.2.0 /usr/lib/hadoop

- Set the path for java and hadoop

**Command:** sudo gedit \$HOME/.bashrc

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
```

```
export PATH=$PATH:$JAVA_HOME/bin
```

```
export HADOOP_COMMON_HOME=/usr/lib/hadoop
```

```
export HADOOP_MAPRED_HOME=/usr/lib/hadoop
```

```
export PATH=$PATH:$HADOOP_COMMON_HOME/bin
```

```
export PATH=$PATH:$HADOOP_COMMON_HOME/Sbin
```

- Checking of java and hadoop

**Command:** java -version

**Command:** hadoop version

##### b) PSEUDO MODE:

Hadoop single node cluster runs on single machine. The namenodes and datanodes are performing on the one machine. The installation and configuration steps as given below:

- Installation of secured shell

**Command:** sudo apt-get install openssh-server



- Create a ssh key for passwordless ssh configuration

**Command:** ssh-keygen -t rsa -P ""

- Moving the key to authorized key

**Command:** cat \$HOME/.ssh/id\_rsa.pub >> \$HOME/.ssh/authorized\_keys  
/\*\*\*\*\*RESTART THE COMPUTER\*\*\*\*\*/

- Checking of secured shell login

**Command:** ssh localhost

- Add JAVA\_HOME directory in hadoop-env.sh file

**Command:** sudo gedit /usr/lib/hadoop/conf/hadoop-env.sh  
export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-i386

- Creating namenode and datanode directories for hadoop

**Command:** sudo mkdir -p /usr/lib/hadoop/dfs/namenode

**Command:** sudo mkdir -p /usr/lib/hadoop/dfs/datanode

- Configure core-site.xml

**Command:** sudo gedit /usr/lib/hadoop/conf/core-site.xml

```
<property>  
<name>fs.default.name</name>  
<value>hdfs://localhost:8020</value>  
</property>
```

- Configure hdfs-site.xml

**Command:** sudo gedit /usr/lib/hadoop/conf/hdfs-site.xml

```
<property>  
<name>dfs.replication</name>  
<value>1</value>  
</property>  
<property>  
<name>dfs.permissions</name>  
<value>>false</value>  
</property>  
<property>  
<name>dfs.name.dir</name>  
<value>/usr/lib/hadoop/dfs/namenode</value>  
</property>  
<property>  
<name>dfs.data.dir</name>  
<value>/usr/lib/hadoop/dfs/datanode</value>  
</property>
```

- Configure mapred-site.xml

**Command:** sudo gedit /usr/lib/hadoop/conf/mapred-site.xml

```
<property>  
<name>mapred.job.tracker</name>
```

<value>localhost:8021</value>

</property>

- Format the name node

**Command:** hadoop namenode -format

- Start the namenode, datanode

**Command:** start-dfs.sh

- Start the task tracker and job tracker

**Command:** start-mapred.sh

- To check if Hadoop started correctly

**Command:** jps

namenode

secondarynamenode

datanode

jobtracker

tasktracker

### c) FULLY DISTRIBUTED MODE:

All the demons like namenodes and datanodes are runs on different machines. The data will replicate according to the replication factor in client machines. The secondary namenode will store the mirror images of namenode periodically. The namenode having the metadata where the blocks are stored and number of replicas in the client machines. The slaves and master communicate each other periodically. The configurations of multinode cluster are given below:

- Configure the hosts in all nodes/machines

**Command:** sudo gedit /etc/hosts/

192.168.1.58 pcetcse1

192.168.1.4 pcetcse2

192.168.1.5 pcetcse3

192.168.1.7 pcetcse4

192.168.1.8 pcetcse5

- Passwordless Ssh Configuration

Create ssh key on namenode/master.

**Command:** ssh-keygen -t rsa -p ""

Copy the generated public key all datanodes/slaves.

**Command:** ssh-copy-id -i ~/.ssh/id\_rsa.pub huser@pcetcse2

**Command:** ssh-copy-id -i ~/.ssh/id\_rsa.pub huser@pcetcse3

**Command:** ssh-copy-id -i ~/.ssh/id\_rsa.pub huser@pcetcse4

**Command:** ssh-copy-id -i ~/.ssh/id\_rsa.pub huser@pcetcse5

/\*\*\*\*\*RESTART ALL NODES/COMPUTERS/MACHINES \*\*\*\*\*/

**NOTE:** Verify the passwordless ssh environment from namenode to all datanodes as “huser” user.

➤ Login to master node

**Command:** ssh pcetcse1

**Command:** ssh pcetcse2

**Command:** ssh pcetcse3

**Command:** ssh pcetcse4

**Command:** ssh pcetcse5

➤ Add JAVA\_HOME directory in hadoop-env.sh file in all nodes/machines

**Command:** sudo gedit /usr/lib/hadoop/conf/hadoop-env.sh  
export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-i386

➤ Creating namenode directory in namenode/master

**Command:** sudo mkdir -p /usr/lib/hadoop/dfs/namenode

➤ Creating namenode directory in datanodes/slaves

**Command:** sudo mkdir -p /usr/lib/hadoop/dfs/datanode Close HTML tag.

Use web based tools to monitor your Hadoop setup.

HDFS Namenode on UI

<http://localhost:50070/>

## 2.4 INPUT/OUTPUT:

ubuntu @localhost> jps

Data node, name nodem

Secondary name node, NodeManager, Resource Manager

Hadoop NameNode localhost:8020 - Mozilla Firefox

localhost:50070/dfshealth.jsp

## NameNode 'localhost:8020'

**Started:** Fri May 08 12:09:25 IST 2015  
**Version:** 1.2.0, r1479473  
**Compiled:** Mon May 6 06:59:37 UTC 2013 by hortonfo  
**Upgrades:** There are no upgrades in progress.

[Browse the filesystem](#)  
[Namenode Logs](#)

---

### Cluster Summary

6 files and directories, 1 blocks = 7 total. Heap Size is 60 MB / 889 MB (6%)

Configured Capacity	: 161.33 GB
DFS Used	: 28.01 KB
Non DFS Used	: 16.01 GB
DFS Remaining	: 145.32 GB
DFS Used%	: 0 %
DFS Remaining%	: 90.07 %
Live Nodes	: 1
Dead Nodes	: 0
Decommissioning Nodes	: 0
Number of Under-Replicated Blocks	: 0

---

### NameNode Storage:

Storage Directory	Type	State
-------------------	------	-------

Hadoop NameNode localhost:8020 - Mozilla Firefox

localhost:50070/dfsnodeList.jsp?whatNodes=LIVE

## NameNode 'localhost:8020'

**Started:** Fri May 08 12:09:25 IST 2015  
**Version:** 1.2.0, r1479473  
**Compiled:** Mon May 6 06:59:37 UTC 2013 by hortonfo  
**Upgrades:** There are no upgrades in progress.

[Browse the filesystem](#)  
[Namenode Logs](#)  
[Go back to DFS home](#)

---

### Live Datanodes : 1

Node	Last Contact	Admin State	Configured Capacity (GB)	Used (GB)	Non DFS Used (GB)	Remaining (GB)	Used (%)	Used (%)	Remaining (%)	Blocks
dn2	0	In Service	161.33	0	16.01	145.32	0		90.07	1

This is Apache Hadoop release 1.2.0

## localhost Hadoop Machine List

### Active Task Trackers

Task Trackers												
Name	Host	# running tasks	Max Map Tasks	Max Reduce Tasks	Task Failures	Directory Failures	Node Health Status	Seconds Since Node Last Healthy	Total Tasks Since Start	Succeeded Tasks Since Start	Total Tasks Last Day	Succeeded Tasks Last Day
<a href="#">tracker_dn2:localhost/127.0.0.1:49820</a>	dn2	0	2	2	0	0	N/A	0	0	0	0	0

This is [Apache Hadoop](#) release 1.2.0

HDFS Jobtracker  
<http://localhost:50030/>

HDFS Logs  
<http://localhost:50070/logs/>

Directory: /logs/ - Mozilla Firefox  
 Hadoop NameNode loc... x Directory: /logs/ x  
 localhost:50070/logs/

### Directory: /logs/

<a href="#">hadoop-sudheer-datanode-dn2.log</a>	6487 bytes	8 May, 2015 12:10:13 PM
<a href="#">hadoop-sudheer-datanode-dn2.log.2015-05-07</a>	301426 bytes	7 May, 2015 9:23:03 PM
<a href="#">hadoop-sudheer-datanode-dn2.out</a>	719 bytes	8 May, 2015 12:09:25 PM
<a href="#">hadoop-sudheer-datanode-dn2.out.1</a>	719 bytes	7 May, 2015 9:00:26 PM
<a href="#">hadoop-sudheer-datanode-dn2.out.2</a>	719 bytes	7 May, 2015 8:55:58 PM
<a href="#">hadoop-sudheer-jobtracker-dn2.log</a>	22631 bytes	8 May, 2015 12:09:39 PM
<a href="#">hadoop-sudheer-jobtracker-dn2.log.2015-05-07</a>	678885 bytes	7 May, 2015 9:22:52 PM
<a href="#">hadoop-sudheer-jobtracker-dn2.out</a>	719 bytes	8 May, 2015 12:09:28 PM
<a href="#">hadoop-sudheer-jobtracker-dn2.out.1</a>	719 bytes	7 May, 2015 9:00:28 PM
<a href="#">hadoop-sudheer-jobtracker-dn2.out.2</a>	719 bytes	7 May, 2015 8:56:01 PM
<a href="#">hadoop-sudheer-namenode-dn2.log</a>	17042 bytes	8 May, 2015 12:11:36 PM
<a href="#">hadoop-sudheer-namenode-dn2.log.2015-05-07</a>	17446 bytes	7 May, 2015 9:00:28 PM
<a href="#">hadoop-sudheer-namenode-dn2.out</a>	719 bytes	8 May, 2015 12:09:24 PM
<a href="#">hadoop-sudheer-namenode-dn2.out.1</a>	719 bytes	7 May, 2015 9:00:24 PM
<a href="#">hadoop-sudheer-namenode-dn2.out.2</a>	719 bytes	7 May, 2015 8:55:57 PM
<a href="#">hadoop-sudheer-secondarynamenode-dn2.log</a>	2085 bytes	8 May, 2015 12:09:32 PM
<a href="#">hadoop-sudheer-secondarynamenode-dn2.log.2015-05-07</a>	296453 bytes	7 May, 2015 9:23:08 PM
<a href="#">hadoop-sudheer-secondarynamenode-dn2.out</a>	719 bytes	8 May, 2015 12:09:27 PM
<a href="#">hadoop-sudheer-secondarynamenode-dn2.out.1</a>	719 bytes	7 May, 2015 9:00:27 PM
<a href="#">hadoop-sudheer-secondarynamenode-dn2.out.2</a>	719 bytes	7 May, 2015 8:56:00 PM
<a href="#">hadoop-sudheer-tasktracker-dn2.log</a>	4969 bytes	8 May, 2015 12:09:35 PM
<a href="#">hadoop-sudheer-tasktracker-dn2.log.2015-05-07</a>	60226 bytes	7 May, 2015 9:22:57 PM
<a href="#">hadoop-sudheer-tasktracker-dn2.out</a>	719 bytes	8 May, 2015 12:09:29 PM
<a href="#">hadoop-sudheer-tasktracker-dn2.out.1</a>	719 bytes	7 May, 2015 9:00:30 PM
<a href="#">hadoop-sudheer-tasktracker-dn2.out.2</a>	719 bytes	7 May, 2015 8:56:02 PM
<a href="#">history/</a>	4096 bytes	7 May, 2015 8:56:08 PM

## tracker\_dn2:localhost/127.0.0.1:49820 Task Tracker Status



Version: 1.2.0, r1479473

Compiled: Mon May 6 06:59:37 UTC 2013 by hortonfo

### Running tasks

Task Attempts	Status	Progress	Errors
---------------	--------	----------	--------

### Non-Running Tasks

Task Attempts	Status
---------------	--------

### Tasks from Running Jobs

Task Attempts	Status	Progress	Errors
---------------	--------	----------	--------

### Local Logs

[Log directory](#)

---

This is [Apache Hadoop](#) release 1.2.0



localhost Hadoop Map/Reduce Administration - Mozilla Firefox

Hadoop NameNode loc... x localhost Hadoop Map/... x

localhost:50030/jobtracker.jsp

## localhost Hadoop Map/Reduce Administration

State: RUNNING  
 Started: Fri May 08 12:09:33 IST 2015  
 Version: 1.2.0, r1479473  
 Compiled: Mon May 6 06:59:37 UTC 2013 by hortonfo  
 Identifier: 201505081209  
 SafeMode: OFF

### Cluster Summary (Heap Size is 55.5 MB/889 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Exc N
0	0	0	1	0	0	0	0	2	2	4.00	0	0	0

### Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)

Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

### Running jobs

## 2.5 PRE LAB VIVA QUESTIONS:

1. What does `_jps` command do?
2. How to restart Namenode?
3. Differentiate between Structured and Unstructured data?

## 2.6 LAB ASSIGNMENT:

- 1 How to configure the daemons in the browser.

## 2.7 POST LAB VIVA QUESTIONS:

1. What are the main components of a Hadoop Application?
2. Explain the difference between NameNode, Backup Node and Checkpoint NameNode.

## WEEK- 3

### USING LINUX OPERATING SYSTEM

#### 3.1 OBJECTIVE:

1. Implementing the basic commands of LINUX Operating System – File/Directory creation, deletion, update operations.

#### 3.2 RESOURCES:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

#### 3.3 PROGRAM LOGIC:

1. cat > filename
2. Add content
3. Press 'ctrl + d' to return to command prompt.

To remove a file use syntax - rm filename

#### 1.4 INPUT/OUTPUT:

List current contents of directory

```
guru99@VirtualBox:~$ ls
Desktop  Downloads  Music  |Public  sample1  Templates
Documents  examples.desktop  Pictures  sample  sample2  Videos
```

Remove the file sample1

```
guru99@VirtualBox:~$ rm sample1
```

List directory , to check file has been deleted

```
guru99@VirtualBox:~$ ls
Desktop  Downloads  Music  Public  sample2  Videos
Documents  examples.desktop  Pictures  sample  Templates
guru99@VirtualBox:~$
```

#### 3.5 PRE-LAB VIVA QUESTIONS:

1. What is ls command?
2. What are the attributes of ls command?

#### 3.6 LAB ASSIGNMENT:

- 1 Write a linux commands for Sed operations?
- 2 Write the linux commands for renaming a file?

#### 3.7 POST-LAB VIVA QUESTIONS:

1. What is the purpose of rm command?
2. What is the difference between Linux and windows commands?



## WEEK-4

### FILE MANAGEMENT IN HADOOP

#### 4.1 OBJECTIVE:

Implement the following file management tasks in Hadoop:

- i. Adding files and directories
- ii. Retrieving files
- iii. Deleting files

Hint: A typical Hadoop workflow creates data files (such as log files) elsewhere and copies them into HDFS using one of the above command line utilities.

#### 4.2 RESOURCES:

VMWare stack, 4 GB RAM, Hard Disk 80 GB.

#### 4.3 PROGRAM LOGIC:

##### Adding Files and Directories to HDFS

Before you can run Hadoop programs on data stored in HDFS, you'll need to put the data into HDFS first. Let's create a directory and put a file in it. HDFS has a default working directory of /user/\$USER, where \$USER is your login user name. This directory isn't automatically created for you, though, so let's create it with the mkdir command. For the purpose of illustration, we use chuck. You should substitute your user name in the example commands.

```
hadoop fs -mkdir /user/chuck
```

```
hadoop fs -put
```

```
hadoop fs -put example.txt /user/chuck
```

##### Retrieving Files from HDFS

The Hadoop command get copies files from HDFS back to the local filesystem. To retrieve example.txt, we can run the following command:

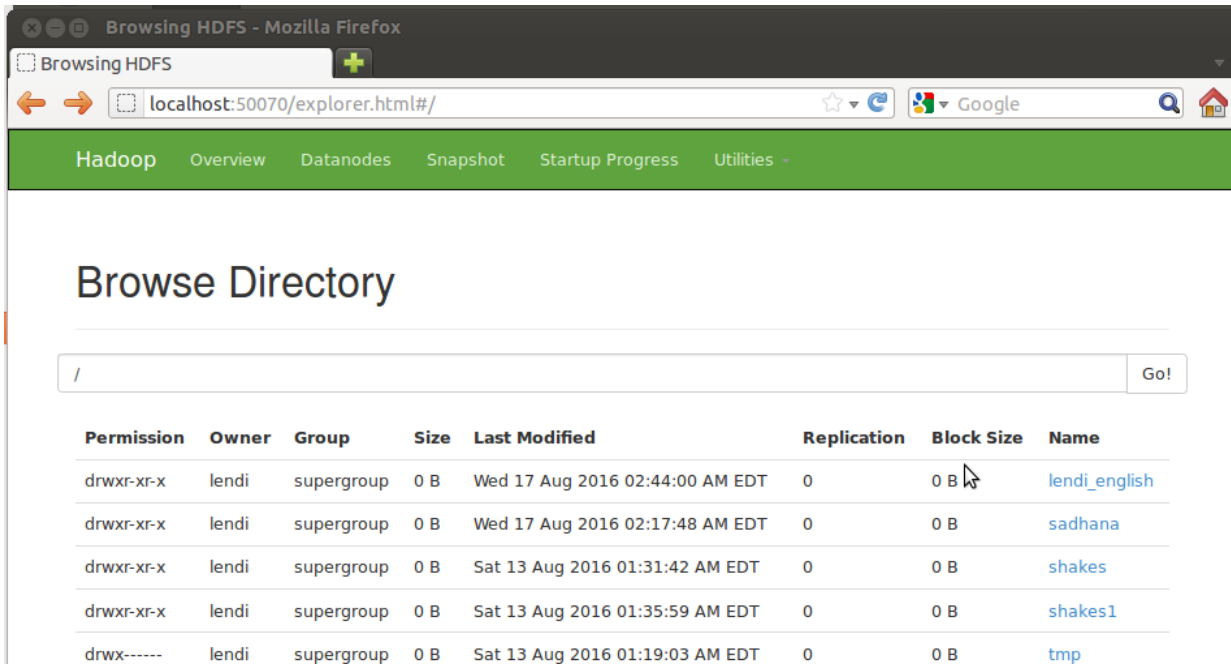
```
hadoop fs -cat example.txt
```

##### Deleting files from HDFS

```
hadoop fs -rm example.txt
```

- Command for creating a directory in hdfs is “hdfs dfs –mkdir /lencise”.
- Adding directory is done through the command “hdfs dfs –put lendi\_english /”.

#### 4.4 INPUT/OUTPUT:



#### 4.5 PRE LAB VIVA QUESTIONS:

- 1) Define Hadoop?
- 2) List out the various use cases of Hadoop?

#### 4.6 LAB ASSIGNMENT

- 1) What is command used to list out directories of Data Node through web tool

#### 4.7 POST LAB VIVA QUESTIONS:

1. Distinguish the Hadoop Ecosystem?
2. Demonstrate divide and conquer philosophy in Hadoop Cluster?

## WEEK-5

### MAPREDUCE PROGRAM 1

#### 5.1 OBJECTIVE:

Run a basic word count Map Reduce program to understand Map Reduce Paradigm.

#### 5.2 RESOURCES:

VMWare stack, 4 GB RAM, Web browser, Hard Disk 80 GB.

#### 5.3 PROGRAM LOGIC:

WordCount is a simple program which counts the number of occurrences of each word in a given text input data set. WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. Our implementation consists of three main parts:

1. Mapper
2. Reducer
3. Driver

#### Step-1. Write a Mapper

A Mapper overrides the `map()` function from the Class "org.apache.hadoop.mapreduce.Mapper" which provides `<key, value>` pairs as the input. A Mapper implementation may output `<key,value>` pairs using the provided Context .

Input value of the WordCount Map task will be a line of text from the input data file and the key would be the line number `<line_number, line_of_text>` . Map task outputs `<word, one>` for each word in the line of text.

#### *Pseudo-code*

```
void Map (key, value){  
    for each word x in value:  
        output.collect(x,1);  
}
```

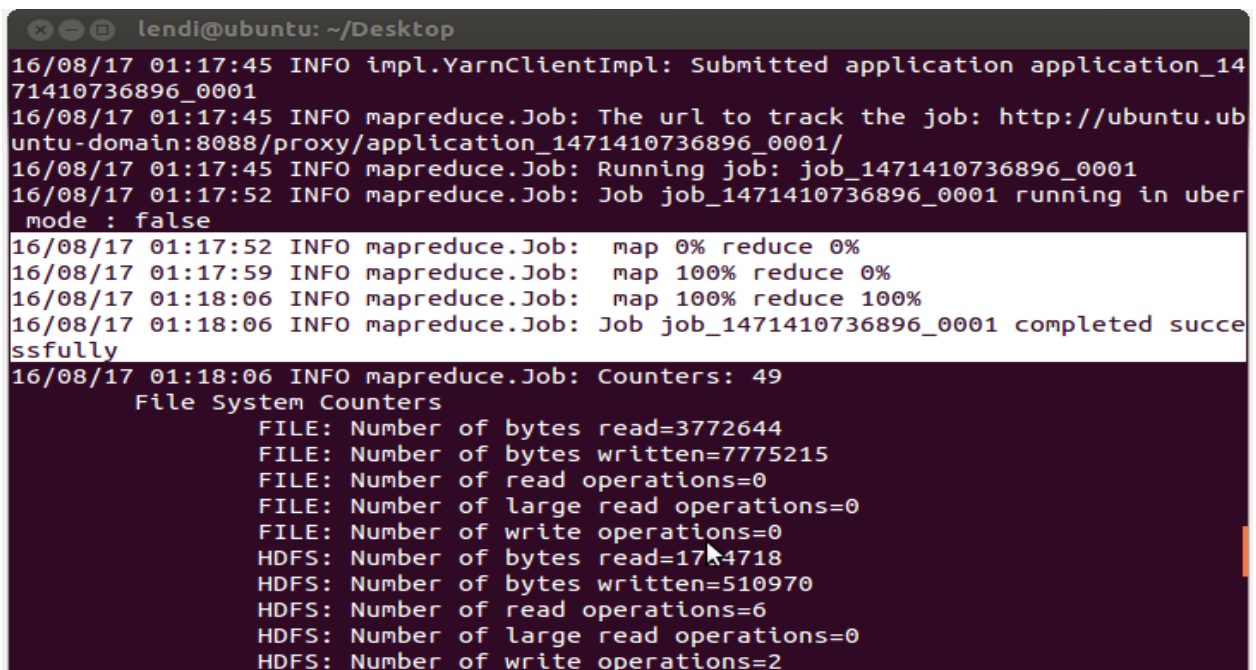
## Step-2. Write a Reducer

A Reducer collects the intermediate <key,value> output from multiple map tasks and assemble a single result. Here, the WordCount program will sum up the occurrence of each word to pairs as <word, occurrence>.

Pseudo-code

```
void Reduce (keyword, <list of value>){ for
    each x in <list of value>:
        sum+=x;
    final_output.collect(keyword, sum);
}
```

## 5.4 INPUT/OUTPUT:

A terminal window showing the execution of a Hadoop job. The window title is 'lendi@ubuntu: ~/Desktop'. The logs show the submission of application 'application\_1471410736896\_0001' at 01:17:45, the job starting at 01:17:52, and completion at 01:18:06. The final output shows counters for the job, including file system and HDFS statistics.

```
lendi@ubuntu: ~/Desktop
16/08/17 01:17:45 INFO impl.YarnClientImpl: Submitted application application_1471410736896_0001
16/08/17 01:17:45 INFO mapreduce.Job: The url to track the job: http://ubuntu.ubuntu-domain:8088/proxy/application_1471410736896_0001/
16/08/17 01:17:45 INFO mapreduce.Job: Running job: job_1471410736896_0001
16/08/17 01:17:52 INFO mapreduce.Job: Job job_1471410736896_0001 running in uber mode : false
16/08/17 01:17:52 INFO mapreduce.Job:  map 0% reduce 0%
16/08/17 01:17:59 INFO mapreduce.Job:  map 100% reduce 0%
16/08/17 01:18:06 INFO mapreduce.Job:  map 100% reduce 100%
16/08/17 01:18:06 INFO mapreduce.Job: Job job_1471410736896_0001 completed successfully
16/08/17 01:18:06 INFO mapreduce.Job: Counters: 49
File System Counters
  FILE: Number of bytes read=3772644
  FILE: Number of bytes written=7775215
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=174718
  HDFS: Number of bytes written=510970
  HDFS: Number of read operations=6
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=2
```

## 5.5 PRE-LAB VIVA QUESTIONS:

1. Justify how hadoop technology satisfies the business insights now -a -days?
2. Define Filesystem?

**5.6 LAB ASSIGNMENT:**

Run a basic word count Map Reduce program to understand Map Reduce Paradigm.

**5.7 POST-LAB VIVA QUESTIONS:**

1. Define what is block in HDFS?
2. Why is a block in HDFS so large?

## WEEK-6

### MAPREDUCE PROGRAM 2

#### 6.1 OBJECTIVE:

Write a Map Reduce program that mines weather data. Hint: Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with Map Reduce, since it is semi structured and record-oriented.

#### 6.2 RESOURCES:

VMWare, Web browser, 4 GB RAM, Hard Disk 80 GB.

#### 6.3 PROGRAM LOGIC:

WordCount is a simple program which counts the number of occurrences of each word in a given text input data set. WordCount fits very well with the MapReduce programming model making it a great example to understand the Hadoop Map/Reduce programming style. Our implementation consists of three main parts:

1. Mapper
2. Reducer
3. Main program

#### Step-1. Write a Mapper

A Mapper overrides the `map` function from the Class "org.apache.hadoop.mapreduce.Mapper" which provides `<key, value>` pairs as the input. A Mapper implementation may output `<key,value>` pairs using the provided Context .

Input value of the WordCount Map task will be a line of text from the input data file and the key would be the line number `<line_number, line_of_text>` . Map task outputs `<word, one>` for each word in the line of text.

#### Pseudo-code

```
void Map (key, value){
for each max_temp x in value:
output.collect(x, 1);
}
void Map (key, value){
    for each min_temp x in value:

output.collect(x, 1);
}
```

### **Step-2 Write a Reducer**

A Reducer collects the intermediate <key,value> output from multiple map tasks and assemble a single result. Here, the WordCount program will sum up the occurrence of each word to pairs as <word, occurrence>.

#### **Pseudo-code**

```
void Reduce (max_temp, <list of value>){
for each x in <list of value>:
sum+=x;
final_output.collect(max_temp, sum);
}
void Reduce (min_temp, <list of value>){
for each x in <list of value>:
sum+=x;
final_output.collect(min_temp, sum);
}
```

### **3. Write Driver**

The Driver program configures and run the MapReduce job. We use the main program to perform basic configurations such as:

Job Name : name of this Job Executable (Jar)

Class: the main executable class. For here, WordCount.

Mapper Class: class which overrides the "map" function. For here, Map.

Reducer: class which override the "reduce" function. For here , Reduce.

Output Key: type of output key. For here, Text.Output

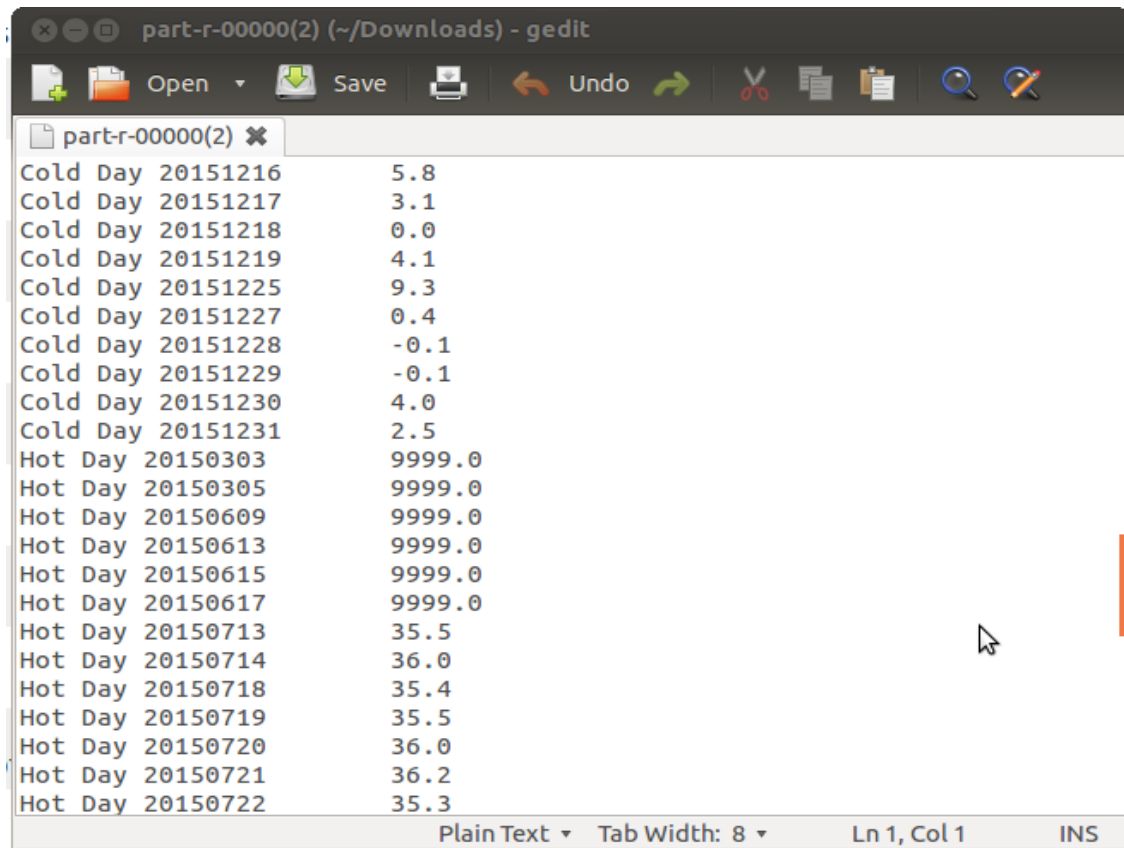
Value: type of output value. For here, IntWritable.

File Input Path

File Output Path

## 6.4 INPUT/OUTPUT:

Set of Weather Data over the years



```
part-r-00000(2) x
Cold Day 20151216      5.8
Cold Day 20151217      3.1
Cold Day 20151218      0.0
Cold Day 20151219      4.1
Cold Day 20151225      9.3
Cold Day 20151227      0.4
Cold Day 20151228     -0.1
Cold Day 20151229     -0.1
Cold Day 20151230      4.0
Cold Day 20151231      2.5
Hot Day 20150303     9999.0
Hot Day 20150305     9999.0
Hot Day 20150609     9999.0
Hot Day 20150613     9999.0
Hot Day 20150615     9999.0
Hot Day 20150617     9999.0
Hot Day 20150713      35.5
Hot Day 20150714      36.0
Hot Day 20150718      35.4
Hot Day 20150719      35.5
Hot Day 20150720      36.0
Hot Day 20150721      36.2
Hot Day 20150722      35.3
Plain Text ▾ Tab Width: 8 ▾ Ln 1, Col 1 INS
```

## 6.5 PRE-LAB VIVA QUESTIONS:

- 1) Explain the function of MapReducer partitioner?
- 2) What is the difference between an Input Split and HDFS Block?
- 3) What is Sequencefileinputformat?

## 6.6 LAB ASSIGNMENT:

1. Using Map Reduce job to Identify language by merging multi language dictionary files into a single dictionary file.
2. Join multiple datasets using a MapReduce Job.

## 6.7 POST-LAB VIVA QUESTIONS:

- 1) In Hadoop what is InputSplit?
- 2) Explain what is a sequence file in Hadoop?



## WEEK-7

### MAPREDUCE PROGRAM 3

#### 7.1 OBJECTIVE:

Implement matrix multiplication with Hadoop Map Reduce.

#### 7.2 RESOURCES:

VMWare, Web browser, 4 GB RAM, Hard Disk 80 GB.

#### 7.3 PROGRAM LOGIC:

We assume that the input files for A and B are streams of (key,value) pairs in sparse matrix format, where each key is a pair of indices (i,j) and each value is the corresponding matrix element value. The output files for matrix  $C=A*B$  are in the same format.

#### We have the following input parameters:

The path of the input file or directory for matrix A.

The path of the input file or directory for matrix B.

The path of the directory for the output files for matrix C.

strategy = 1, 2, 3 or 4.

R = the number of reducers.

I = the number of rows in A and C.

K = the number of columns in A and rows in B.

J = the number of columns in B and C.

IB = the number of rows per A block and C block.

KB = the number of columns per A block and rows per B block.

JB = the number of columns per B block and C block.

In the pseudo-code for the individual strategies below, we have intentionally avoided factoring common code for the purposes of clarity. Note that in all the strategies the memory footprint of both the mappers and the reducers is flat at scale.

Note that the strategies all work reasonably well with both dense and sparse matrices. For sparse matrices we do not emit zero elements. That said, the simple pseudo-code for multiplying the individual blocks shown here is certainly not optimal for sparse matrices. As a learning exercise, our focus here is on mastering the MapReduce complexities, not on optimizing the sequential matrix multiplication algorithm for the individual blocks.

#### Steps

1. setup ()
2. var NIB = (I-1)/IB+1
3. var NKB = (K-1)/KB+1
4. var NJB = (J-1)/JB+1
5. map (key, value)

6. if from matrix A with key=(i,k) and value=a(i,k)
  7. for  $0 \leq j_b < N_{JB}$
  8. emit (i/IB, k/KB, j\_b, 0), (i mod IB, k mod KB, a(i,k))
  9. if from matrix B with key=(k,j) and value=b(k,j)
  10. for  $0 \leq i_b < N_{IB}$
  - emit (i\_b, k/KB, j/JB, 1), (k mod KB, j mod JB, b(k,j))
- Intermediate keys (i\_b, k\_b, j\_b, m) sort in increasing order first by i\_b, then by k\_b, then by j\_b, then by m. Note that m = 0 for A data and m = 1 for B data.

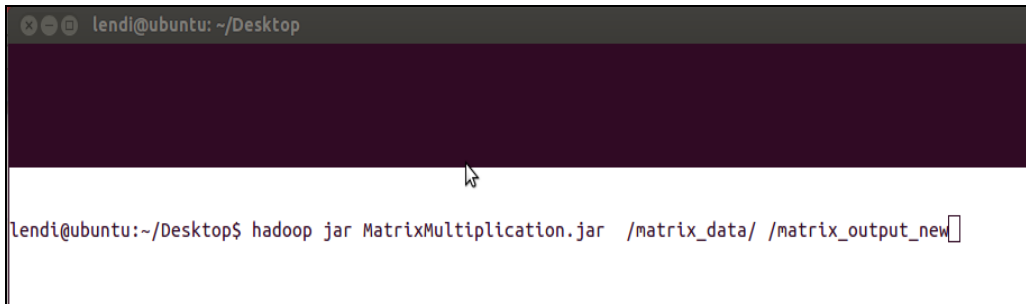
**The partitioner maps intermediate key (i\_b, k\_b, j\_b, m) to a reducer r as follows:**

11.  $r = ((i_b * JB + j_b) * KB + k_b) \bmod R$
12. These definitions for the sorting order and partitioner guarantee that each reducer R[i\_b, k\_b, j\_b] receives the data it needs for blocks A[i\_b, k\_b] and B[k\_b, j\_b], with the data for the A block immediately preceding the data for the B block.
13. var A = new matrix of dimension IBxKB
14. var B = new matrix of dimension KBxJB
15. var s\_i\_b = -1
16. var s\_k\_b = -1

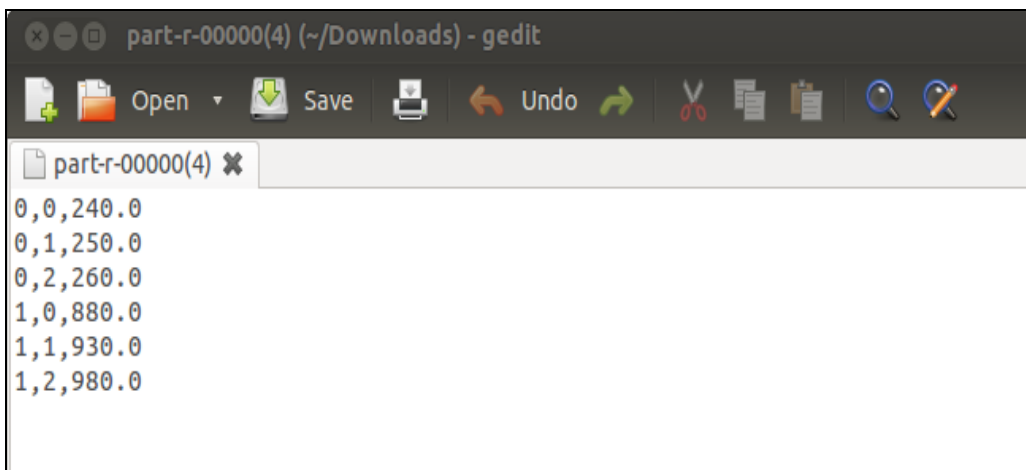
**Reduce (key, valueList)**

17. if key is (i\_b, k\_b, j\_b, 0)
  18. // Save the A block.
  19. s\_i\_b = i\_b
  20. s\_k\_b = k\_b
  21. Zero matrix A
  22. for each value = (i, k, v) in valueList A(i,k) = v
  23. if key is (i\_b, k\_b, j\_b, 1)
  24. if i\_b != s\_i\_b or k\_b != s\_k\_b return // A[i\_b, k\_b] must be zero!
  25. // Build the B block.
  26. Zero matrix B
  27. for each value = (k, j, v) in valueList B(k,j) = v
  28. // Multiply the blocks and emit the result.
  29. i\_base = i\_b \* IB
  30. j\_base = j\_b \* JB
  31. for  $0 \leq i < \text{row dimension of A}$
  32. for  $0 \leq j < \text{column dimension of B}$
  33. sum = 0
  34. for  $0 \leq k < \text{column dimension of A} = \text{row dimension of B}$
  - a. sum += A(i,k)\*B(k,j)
  35. if sum != 0 emit (i\_base+i, j\_base+j), sum
- Set of Data sets over different Clusters are taken as Rows and Columns

## 7.4 INPUT/OUTPUT:



```
lendi@ubuntu: ~/Desktop
lendi@ubuntu:~/Desktop$ hadoop jar MatrixMultiplication.jar /matrix_data/ /matrix_output_new
```



```
part-r-00000(4) (~/Downloads) - gedit
0,0,240.0
0,1,250.0
0,2,260.0
1,0,880.0
1,1,930.0
1,2,980.0
```

## 7.5 PRE-LAB VIVA QUESTIONS:

1. Explain what is “map” and what is “reducer” in Hadoop?
2. Mention what daemons run on a master node and slave nodes?
3. Mention what is the use of Context Object?

## 7.6 LAB ASSIGNMENT:

1. Implement matrix addition with Hadoop Map Reduce.

## 7.7 POST-LAB VIVA QUESTIONS:

1. What is partitioner in Hadoop?
2. Explain of RecordReader in Hadoop?

## WEEK-8

### PIG LATIN LANGUAGE - PIG

#### 8.1 OBJECTIVE:

1. Installation of PIG.

#### 8.2 RESOURCES:

VMWare, Web browser, 4 GB RAM, Hard Disk 80 GB.

#### 8.3 PROGRAM LOGIC:

##### STEPS FOR INSTALLING APACHE PIG

1) Extract the pig-0.15.0.tar.gz and move to home directory

2) Set the environment of PIG in bashrc file.

3) Pig can run in two modes

Local Mode and Hadoop Mode

Pig -x local and pig

4) Grunt Shell

Grunt >

5) LOADING Data into Grunt Shell

DATA = LOAD <CLASSPATH> USING PigStorage(DELIMITER) as (ATTRIBUTE :  
DataType1, ATTRIBUTE : DataType2.....)

6) Describe Data

Describe DATA;

7) DUMP Data

Dump DATA;

#### 8.4 INPUT/OUTPUT:

Input as Website Click Count Data

```
lendi@ubuntu: ~
grunt> ad1 = load '/home/lendi/Desktop/static_data/ad_data/ad_data1.txt' using PigStorage('\t') as (item:chararray,campaignId:chararray,date:chararray,time:chararray,display_site:chararray,was_clicked:int,cpc:int,country:chararray,placement:chararray);
2016-10-14 02:35:32,441 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-10-14 02:35:32,441 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> describe ad1;
ad1: {item: chararray,campaignId: chararray,date: chararray,time: chararray,display_site: chararray,was_clicked: int,cpc: int,country: chararray,placement: chararray}
grunt> ad2 = load '/home/lendi/Desktop/static_data/ad_data/ad_data2.txt' using PigStorage(',') as (campaignId:chararray,date:chararray,time:chararray,display_site:chararray,placement:chararray,was_clicked:int,cpc:int,item:chararray);
2016-10-14 02:36:08,732 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2016-10-14 02:36:08,732 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> describe ad2;
ad2: {campaignId: chararray,date: chararray,time: chararray,display_site: chararray,placement: chararray,was_clicked: int,cpc: int,item: chararray}
grunt>
```

### 8.5 PRE-LAB VIVA QUESTIONS:

- 1) What do you mean by a bag in Pig?
- 2) Differentiate between PigLatin and HiveQL
- 3) How will you merge the contents of two or more relations and divide a single relation into two or more relations?

### 8.6 LAB ASSIGNMENT:

1. Process baseball data using Apache Pig.

### 8.7 POST-LAB VIVA QUESTIONS:

1. What is the usage of foreach operation in Pig scripts?
2. What does Flatten do in Pig

## WEEK-9

### PIG COMMANDS

#### 9.1 OBJECTIVE:

Write Pig Latin scripts sort, group, join, project, and filter your data.

#### 9.2 RESOURCES:

VMWare, Web browser, 4 GB RAM, Hard Disk 80 GB.

#### 9.3 PROGRAM LOGIC:

##### **FILTER Data**

FDATA = FILTER DATA by ATTRIBUTE = VALUE;

##### **GROUP Data**

GDATA = GROUP DATA by ATTRIBUTE;

##### **Iterating Data**

FOR\_DATA = FOREACH DATA GENERATE GROUP AS GROUP\_FUN,  
ATTRIBUTE = <VALUE>

##### **Sorting Data**

SORT\_DATA = ORDER DATA BY ATTRIBUTE WITH CONDITION;

##### **LIMIT Data**

LIMIT\_DATA = LIMIT DATA COUNT;

##### **JOIN Data**

JOIN DATA1 BY (ATTRIBUTE1,ATTRIBUTE2....) , DATA2 BY  
(ATTRIBUTE3,ATTRIBUTE....N)

#### 9.4 INPUT / OUTPUT :



```
lendi@ubuntu: ~
grunt> join_data = join ad1 by (campaignId,display_site,cpc),ad2 by (campaignId,
display_site,cpc);
grunt> describe join_data;
join_data: {ad1::item: chararray,ad1::campaignId: chararray,ad1::date: chararray
,ad1::time: chararray,ad1::display_site: chararray,ad1::was_clicked: int,ad1::cp
c: int,ad1::country: chararray,ad1::placement: chararray,ad2::campaignId: charar
ray,ad2::date: chararray,ad2::time: chararray,ad2::display_site: chararray,ad2::
placement: chararray,ad2::was_clicked: int,ad2::cpc: int,ad2::item: chararray}
grunt> █
```

**9.5 PRE-LAB VIVA QUESTIONS:**

1. How will you merge the contents of two or more relations and divide a single relation into two or more relations?
2. What is the usage of foreach operation in Pig scripts?
3. What does Flatten do in Pig?

**9.6 LAB ASSIGNMENT:**

1. Using Apache Pig to develop User Defined Functions for student data.

**9.7 PRE-LAB VIVA QUESTIONS:**

1. What do you mean by a bag in Pig?
2. Differentiate between PigLatin and HiveQL

## WEEK-10

### PIG LATIN MODES, PROGRAMS

#### 10.1 OBJECTIVE:

- a. Run the Pig Latin Scripts to find Word Count.
- b. Run the Pig Latin Scripts to find a max temp for each and every year.

#### 10.2 RESOURCES:

VMWare, Web Browser, 4 GB RAM, 80 GB Hard Disk.

#### 10.3 PROGRAM LOGIC:

Run the Pig Latin Scripts to find Word Count.

```
lines = LOAD '/user/hadoop/HDFS_File.txt' AS (line:chararray);
words = FOREACH lines GENERATE FLATTEN(TOKENIZE(line)) as word;
grouped = GROUP words BY word;
wordcount = FOREACH grouped GENERATE group, COUNT(words);
DUMP wordcount;
```

Run the Pig Latin Scripts to find a max temp for each and every year

```
-- max_temp.pig: Finds the maximum temperature by year
records = LOAD 'input/ncdc/micro-tab/sample.txt'
AS (year:chararray, temperature:int, quality:int);
filtered_records = FILTER records BY temperature != 9999 AND
(quality == 0 OR quality == 1 OR quality == 4 OR quality == 5 OR quality == 9);
grouped_records = GROUP filtered_records BY year;
max_temp = FOREACH grouped_records GENERATE group,
MAX(filtered_records.temperature);
DUMP max_temp;
```

#### 10.4 INPUT / OUTPUT:

```
(1950,0,1)
(1950,22,1)
(1950,-11,1)
(1949,111,1)
(1949,78,1)
```

#### 10.5 PRE-LAB VIVA QUESTIONS:

1. List out the benefits of Pig?
2. Classify Pig Latin commands in Pig?



**10.6 LAB ASSIGNMENT:**

1. Analyzing average stock price from the stock data using Apache Pig

**10.7 POST-LAB VIVA QUESTIONS:**

1. Discuss the modes of Pig scripts?
2. Explain the Pig Latin application flow?

## WEEK-11

### HIVE

#### 11.1 OBJECTIVE:

Installation of HIVE.

#### 11.2 RESOURCES:

VMWare, Web Browser, 1GB RAM, Hard Disk 80 GB.

#### 11.3 PROGRAM LOGIC:

Install MySQL-Server

- 1) Sudo apt-get install mysql-server
- 2) Configuring MySQL UserName and Password
- 3) Creating User and granting all Privileges  
Mysql –uroot –proot  
Create user <USER\_NAME> identified by <PASSWORD>
- 4) Extract and Configure Apache Hive  
tar xvfz apache-hive-1.0.1.bin.tar.gz
- 5) Move Apache Hive from Local directory to Home directory
- 6) Set CLASSPATH in bashrc  
Export HIVE\_HOME = /home/apache-hive  
Export PATH = \$PATH:\$HIVE\_HOME/bin
- 7) Configuring hive-default.xml by adding My SQL Server Credentials  
<property>  
<name>javax.jdo.option.ConnectionURL</name>  
<value>  
jdbc:mysql://localhost:3306/hive?createDatabaseIfNotExist=true  
</value>  
</property>  
<property>  
<name>javax.jdo.option.ConnectionDriverName</name>  
<value>com.mysql.jdbc.Driver</value>  
</property>  
<property>  
<name>javax.jdo.option.ConnectionUserName</name>  
<value>hadoop</value>  
</property>  
<property>  
<name>javax.jdo.option.ConnectionPassword</name>  
<value>hadoop</value>  
</property>
- 8) Copying mysql-java-connector.jar to hive/lib directory.

## 11.4 INPUT/OUTPUT:

```
administrator@ubuntu: ~
d yet. Please use TIMESTAMP instead
hive> create table log_data(l_date string,l_time string,s_sitename string,s_comput
ername string,l_uri string,uri_query string,ip_address string,user_agent string,
status1 int,status2 int,s_bytes int,c_bytes int,time_taken int);
OK
Time taken: 0.331 seconds
hive> show tables;
OK
log_data
Time taken: 0.074 seconds, Fetched: 1 row(s)
hive> desc log_data;
OK
l_date          string          None
l_time          string          None
s_sitename      string          None
s_computername  string          None
l_uri           string          None
uri_query       string          None
ip_address      string          None
user_agent      string          None
status1         int             None
status2         int             None
s_bytes         int             None
c_bytes         int             None
```

## 11.5 PRE-LAB VIVA QUESTIONS:

1. In Hive, explain the term 'aggregation' and its uses?
2. List out the Data types in Hive?

## 11.6 LAB ASSIGNMENT:

1. Analyze twitter data using Apache Hive.

## 11.7 POST-LAB VIVA QUESTIONS:

1. Explain the Built-in Functions in Hive?
2. Describe the various Hive Data types?

## WEEK-12

### HIVE OPERATIONS

#### 12.1 OBJECTIVE:

Use Hive to create, alter, and drop databases, tables, views, functions, and indexes.

#### 12.2 RESOURCES:

VMWare, XAMPP Server, Web Browser, 1GB RAM, Hard Disk 80 GB.

#### 12.3 PROGRAM LOGIC:

##### **SYNTAX for HIVE Database Operations**

##### **DATABASE Creation**

CREATE DATABASE|SCHEMA [IF NOT EXISTS] <database name>

##### **Drop Database Statement**

DROP DATABASE Statement DROP (DATABASE|SCHEMA) [IF EXISTS]  
database\_name [RESTRICT|CASCADE];

##### **Creating and Dropping Table in HIVE**

CREATE [TEMPORARY] [EXTERNAL] TABLE [IF NOT EXISTS] [db\_name.]  
table\_name

[(col\_name data\_type [COMMENT col\_comment], ...)]

[COMMENT table\_comment] [ROW FORMAT row\_format] [STORED AS  
file\_format]

##### **Loading Data into table log\_data**

##### **Syntax:**

**LOAD DATA LOCAL INPATH '<path>/u.data' OVERWRITE INTO TABLE  
u\_data;**

##### **Alter Table in HIVE**

##### **Syntax**

ALTER TABLE name RENAME TO new\_name

ALTER TABLE name ADD COLUMNS (col\_spec[, col\_spec ...])

ALTER TABLE name DROP [COLUMN] column\_name

ALTER TABLE name CHANGE column\_name new\_name new\_type

ALTER TABLE name REPLACE COLUMNS (col\_spec[, col\_spec ...])

##### **Creating and Dropping View**

CREATE VIEW [IF NOT EXISTS] view\_name [(column\_name [COMMENT  
column\_comment], ...)] [COMMENT table\_comment] AS SELECT ...

##### **Dropping View**

##### **Syntax:**

DROP VIEW view\_name

##### **Functions in HIVE**

String Functions:- round(), ceil(), substr(), upper(), reg\_exp() etc

Date and Time Functions:- year(), month(), day(), to\_date() etc

Aggregate Functions :- sum(), min(), max(), count(), avg() etc

## **INDEXES**

```
CREATE INDEX index_name ON TABLE base_table_name (col_name, ...)
AS 'index.handler.class.name'
[WITH DEFERRED REBUILD]
[IDXPARTICTIONS (property_name=property_value, ...)]
[IN TABLE index_table_name]
[PARTITIONED BY (col_name, ...)]
[
[ ROW FORMAT ...] STORED AS ...
| STORED BY ...
]
[LOCATION hdfs_path]
[TBLPROPERTIES (...)]
```

### **Creating Index**

```
CREATE INDEX index_ip ON TABLE log_data(ip_address) AS
'org.apache.hadoop.hive.ql.index.compact.CompactIndexHandler' WITH DEFERRED
REBUILD;
```

### **Altering and Inserting Index**

```
ALTER INDEX index_ip_address ON log_data REBUILD;
```

### **Storing Index Data in Metastore**

```
SET
hive.index.compact.file=/home/administrator/Desktop/big/metastore_db/tmp/index_ipadd
ress_result;
SET
hive.input.format=org.apache.hadoop.hive.ql.index.compact.HiveCompactIndexInputFor
mat;
```

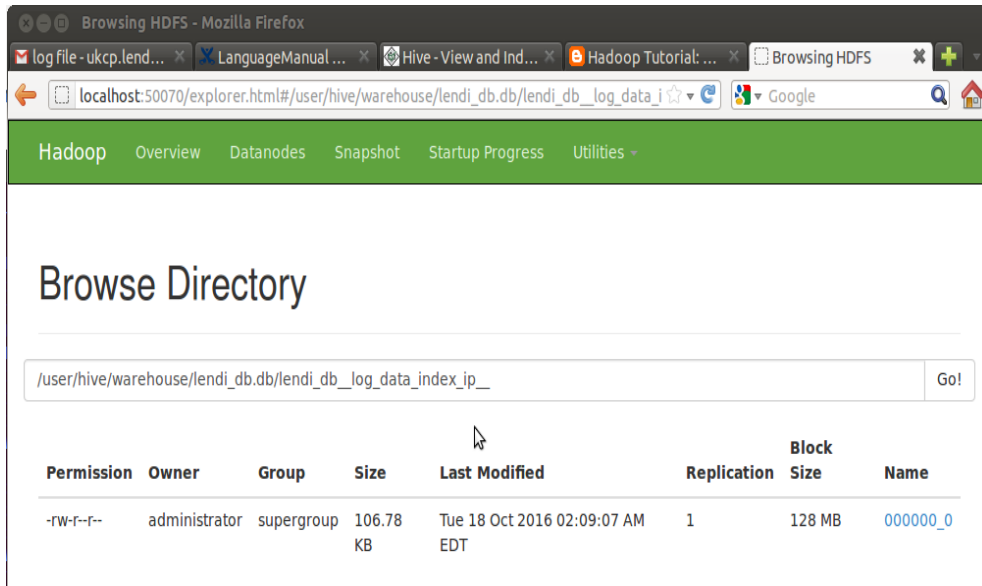
### **Dropping Index**

```
DROP INDEX INDEX_NAME on TABLE_NAME;
```

## 12.4 INPUT/OUTPUT:

```
administrator@ubuntu: ~
0.6.20.6 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.1;+Trident/4.0;+GTB7.5;+SLC
R+2.0.50727;+.NET+CLR+3.5.30729;+.NET+CLR+3.0.30729;+Media+Center+PC+6.0;+InfoPath.2) 304
11 498 0
2014-12-23 23:08:38 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pic3.jpg
0.6.20.6 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.1;+Trident/4.0;+GTB7.5;+SLC
R+2.0.50727;+.NET+CLR+3.5.30729;+.NET+CLR+3.0.30729;+Media+Center+PC+6.0;+InfoPath.2) 304
10 497 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/css/demo.css - 10.
ozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.50727;+.NET+CLR+3.0.0
CLR+1.1.4322;+InfoPath.2) 304 0 210 458 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/css/elasticlide.css -
0.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.50727;+.NET+
06;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 210 465 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pic11.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 211 469 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pic12.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 211 469 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pic10.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 211 469 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pic9.jpg
0.3.20.22 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+6.0;+SLCC1;+.NET+CLR+2.0.5072
+3.0.04506;+.NET+CLR+1.1.4322;+InfoPath.2) 304 0 210 467 0
2014-12-23 23:16:07 W3SVC1 NEWINTSERV2 /trf/elastic/images/small/pica.jpg
```

```
administrator@ubuntu: ~
hive> select * from index_ip;
FAILED: SemanticException [Error 10001]: Line 1:14 Table not found 'index ip'
hive> INSERT OVERWRITE DIRECTORY '/home/administrator/Desktop/hive_data/index_test_result' SELECT `
bucketname`, `_offsets` FROM lendi_db.lendi_db_log_data_index_ip__ where ip_address='141.0.11.19
9';
Total MapReduce jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1476764326039_0014, Tracking URL = http://ubuntu.ubuntu-domain:8088/proxy/applica
tion_1476764326039_0014/
Kill Command = /home/administrator/hadoop-2.7.1/bin/hadoop job -kill job_1476764326039_0014
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2016-10-18 02:16:23,240 Stage-1 map = 0%, reduce = 0%
2016-10-18 02:16:27,406 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
2016-10-18 02:16:28,442 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
2016-10-18 02:16:29,472 Stage-1 map = 100%, reduce = 0%, Cumulative CPU 1.32 sec
MapReduce Total cumulative CPU time: 1 seconds 320 msec
Ended Job = job_1476764326039_0014
Stage-3 is selected by condition resolver.
Stage-2 is filtered out by condition resolver.
Stage-4 is filtered out by condition resolver.
Moving data to: hdfs://localhost:9000/tmp/hive-administrator/hive_2016-10-18_02-16-17_425_5894975364
0454830/-ext-10000
Moving data to: /home/administrator/Desktop/hive data/index test result
```



### 12.5 PRE-LAB VIVA QUESTIONS:

1. How many types of joins are there in Pig Latin with an examples?
2. Write the Hive command to create a table with four columns: First name, last name, age, and income?

### 12.6 LAB ASSIGNMENT:

1. Analyze stock data using Apache Hive.

### 12.7 POST-LAB VIVA QUESTIONS:

1. Write a shell command in Hive to list all the files in the current directory?
2. List the collection types provided by Hive for the purpose a start-up company want to use Hive for storing its data.