Journal of Information Engineering and Applications
ISSN 2224-5782 (print) ISSN 2225-0506 (online)
Vol.5, No.10, 2015

A Comparative Performance Evaluation of Hive and Map Reduce for Big-Data

Ahmad Shaker Abdalrada
Wasit University, College of Arts, Center of computer and informatics, Wasit, Iraq

Dr.Naseer Ali Husieen
Wasit University, College of Education, Department of Computer Science

Abstract:
Advances in information stockpiling and mining advances make it conceivable to safeguard expanding measures of information created specifically or in a roundabout way by clients and break down it to yield important new bits of knowledge. Huge information can uncover individuals' shrouded behavioral examples and even revealed insight into their expectations. All the more absolutely, it can overcome any and all hardships between what individuals need to do and what they really do and how they connect with others and their surroundings. This data is valuable to government offices and in addition privately owned businesses to bolster choice making in zones going from law requirement to social administrations to country security. One of the proficient advancements that arrangement with the Big Data is Hadoop, which will be talked about in this paper. Hadoop, for preparing extensive information volume employments utilizes MapReduce programming model. Hadoop makes utilization of diverse schedulers for executing the occupations in parallel. The default scheduler is FIFO (First In First Out) Scheduler. Different schedulers with need, pre-emption and non-pre-emption alternatives have likewise been produced. As the time has passed the MapReduce has come to few of its restrictions. So keeping in mind the end goal to beat the constraints of MapReduce, the up and coming era of MapReduce has been produced called as YARN (Yet Another Resource Negotiator). Along these lines, this paper gives a review on Hadoop, few booking strategies it uses and a brief prologue to YARN.
Keywords: Big-Data, Hive, Map Reduce

Introduction

It's especially of enthusiasm to connected territories of situational mindfulness and the expectant methodologies needed for close constant disclosure. While huge information can yield to a great degree helpful data, it additionally gives new difficulties appreciation to the amount of information to store, the amount of this will cost, whether the information will be secure, and to what extent it must be kept up. There is an awesome interest for enhancing the Big Data administration procedures. The preparing of this gigantic can be best done utilizing appropriated figuring and parallel handling components. Hadoop [1] is an appropriated processing stage written in Java which fuses highlights such as MapReduce programming and Google File System standard. Hadoop structure assuages the engineers from parallelization issues while permitting them to concentrate on their processing issue and these parallelization issues are taken care of characteristically by the system.

Hadoop is a structure intended to work with enormous measure of information sets which is much bigger in extent than the ordinary frameworks can deal with. Hadoop disseminates this information over an arrangement of machines. The genuine force of Hadoop originates from the actuality its ability to versatile to hundreds or a great many PCs every containing a few processor centers. Numerous huge endeavors trust that inside of a couple of years more than a large portion of the world's information will be put away in Hadoop [2, 26, 27]. Moreover, Hadoop consolidated with Virtual Machine gives more doable results. Hadoop for the most part comprises of i) Hadoop Distributed File System (HDFS): a disseminated record framework to accomplish stockpiling and adaptation to non-critical failure and ii) Hadoop MapReduce an intense parallel programming model which forms immeasurable amount of information by means of dispersed figuring over the bunches.

Hadoop Distributed File System [3, 4, 25] is an opensource document framework that has been outlined particularly to handle expansive records that customary record framework can't deal with. The substantial measure of information is part, recreated and scattered on various machines. The replication of information encourages fast reckoning and unwavering quality. That is the reason HDFS can likewise be called as self-recuperating dispersed document framework implying that, if a specific duplicate of the information gets degenerate or all the more particularly to say if the DataNode on which the information was living falls flat then reproduced duplicate can be utilized. This guarantees that the on-going work proceeds with no disturbance [5, 23, 24].

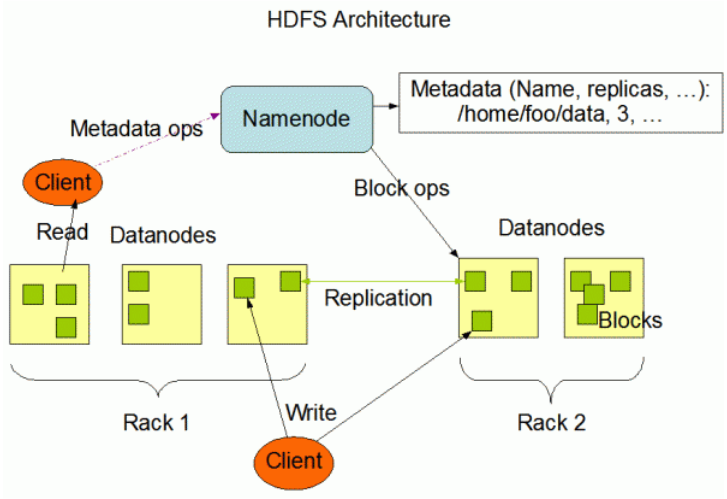

Figure 1: Hadoop Distributed File System

HDFS has ace and slave structural planning. The building design of HDFS is demonstrated above in Figure 1 [7, 20, 21]. In figure letters in order A, B, C speaks to information square and D fallowed by a number speaks to a numbered DataNode. HDFS gives circulated and exceedingly blame tolerant biological system. One Single NameNode alongside different DataNodes is available in a regular HDFS group. The NameNode, an expert server handles the obligation of dealing with the namespace of filesystem and oversees the entrance by customers to records. The namespace records the creation, cancellation and alteration of documents by clients. NameNode maps information squares to DataNodes and oversees record framework operations like opening, shutting and renaming of documents and registries. It is all upon the bearings of NameNode, the DataNodes performs operations on pieces of information, for example, creation, erasure and replication. The piece size is of 64MB and is reproduced into 3 duplicates. The second duplicate is put away on the nearby rack itself while the third on remote rack. A rack is only a gathering of information hubs .

MapReduce [5] [6] is a vital innovation which was proposed by Google. MapReduce is a disentangled programming model and is a noteworthy part of Hadoop for parallel preparing of endless measure of information. It diminishes software engineers from the weight of parallelization issues while permitting them to uninhibitedly focus on application improvement. The chart of Hadoop MapReduce is demonstrated in Figure 2 beneath. Two vital information preparing capacities contained in MapReduce writing computer programs are Map and Reduce.

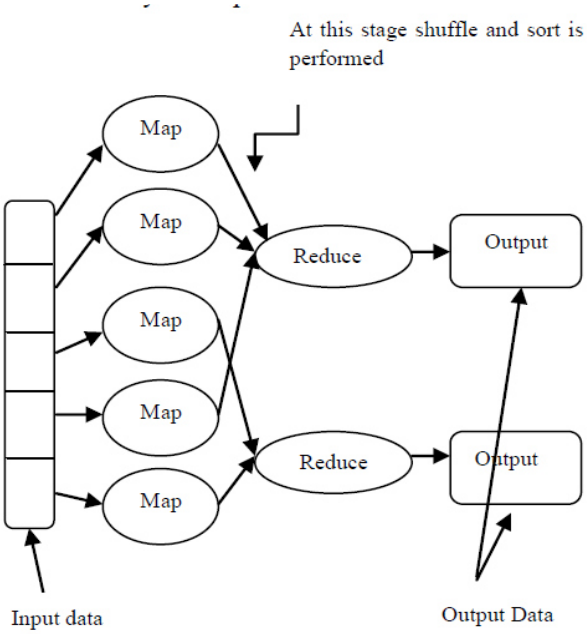

Figure 2: Hadoop MapReduce

The first information will be given as data to the Map stage which performs handling according to the programming done by the software engineers to create halfway results. Parallel Map errands will keep running at once. Firstly, the info information is part into altered measured pieces on which parallel Map undertakings are

run. The yield of the Map technique is a gathering of key/worth sets which is still a middle yield. These sets experience a rearranging stage crosswise over lessen undertakings. One and only key is acknowledged by each decrease undertaking and taking into account this key the preparing will be finished. At last the yield will be as key/worth sets.

The Hadoop MapReduce structure comprises of one Master hub termed as JobTracker and numerous Worker hubs called as TaskTrackers. The client submitted occupations are given as information to the JobTracker which changes them into a quantities of Map and Reduce assignments. These undertakings are doled out to the TaskTrackers. The TaskTrackers investigates the execution of these assignments and toward the end when all errands are proficient; the client is informed about occupation finish. HDFS accommodates adaptation to non-critical failure and dependability by putting away and duplicating the inputs and yields of a Hadoop work.

This paper shows the Hive and Map Reduce comparison in Big-Data analysis for real applications.

Motivation

Advances in electronic sensors, exchanges, figuring, and limit have made huge aggregations of data, getting information of worth to business, science, government, and society. A valid example, web searcher associations, for instance, Google, Yahoo!, and Microsoft have made a by and large new business by getting the information wholeheartedly available on the World Wide Web and offering it to people in accommodating ways. These associations accumulate trillions of bytes of data reliably and always incorporate new organizations, for instance, satellite pictures, driving direction, and picture recuperation. The societal preferences of these organizations are inconceivable, having changed how people find and make use of information once per day [8, 9].

Hive request will run speedier, thusly upgrading customer experience. Besides, customers will have passage to a generous, non-MR execution engine that has authoritatively wound up being a principle elective for data planning furthermore spilling, and which is among the most element ventures over all of Apache from supplier and submit points. Hive-on-Spark will be astoundingly critical for customers who are starting now using Spark for other data taking care of and machine-adjusting needs. Systematizing on one execution back end is in like manner worthwhile for operational organization, making it less requesting to examine issues and make overhauls [9, 10].

Over the span of the latest couple of years, the MapReduce perfect model has found developed accomplishment in various web associations (e.g. Google, Yahoo, Facebook, et cetera) that need to record or for the most part research various Terabytes to Petabytes of data consistently. This accomplishment has realized the release or overhaul of a couple open-source structures for adaptable stockpiling and planning through MapReduce. Apache Hadoop is one of the most likely seen, comprehensively valuable ventures around there, giving an extensible MapReduce structure close by a blemish tolerant passed on filesystem for far reaching data called HDFS (Hadoop Distributed File System). The availability and extensibility of such open-source frameworks has provoked amazing interest and experimentation in the academic world [11, 12].

Literature Review

Hadoop structure is prominent for HDFS and MapReduce. The Hadoop Ecosystem likewise contains diverse ventures which are talked about underneath [7] [8]:

- Apache HBase: A section situated, non-social dispersed key/esteem information store which is based to keep running on top of the HDFS stage. It is intended to scale out evenly in disseminated register groups.
- Apache Hive: An information distribution center base based on top of Hadoop for giving information synopsis, questioning, and investigating expansive datasets put away on Hadoop documents. It is not intended to offer ongoing questions, but rather it can bolster content documents, and arrangement records.
- Apache Pig: It gives an abnormal state parallel instrument for the programming of MapReduce employments to be executed on Hadoop bunches. It utilizes a scripting dialect known as Pig Latin, which is an information stream dialect outfitted towards handling of information in a parallel way.
- Apache Zookeeper: It is an Application Program Interface (API) that permits appropriated preparing of information in expansive frameworks to synchronize with one another so as to give predictable information to customer solicitations.
- Apache Sqoop: An apparatus intended for proficiently exchanging mass information in the middle of Hadoop and organized information stores, for example, social databases.
- Apache Flume: An appropriated administration for gathering, collecting, and moving expansive measure of log information.
- DataNode: A DataNode stores information as required File System of Hadoop. A utilitarian document framework has many DataNode, together the information imitated crosswise over them.
- NameNode: The NameNode is the centerpiece of a HDFS document framework. It keeps the catalog of all records in the document framework, and tracks where over the group the document information is kept. It doesn't store the information of these record itself.
- Jobtracker: The Jobtracker is the administration inside hadoop that ranches out MapReduce to particular

hubs in the bunch, in a perfect world the hubs that have the information, or at least are in the same rack.

- TaskTracker: A TaskTracker is a hub in the bunch that acknowledges undertakings Reduce, Map and operations with shuffle on a Job Tracked.
- Secondary Namenode: Auxiliary Namenode entire design is to have a checkpoint in HDFS. It is only a partner hub for namenode.

Few of the applications of Hadoop are given below [9, 10, 11]:

Table 1: Applications of Hadoop

Log Analysis	:	Log and/or clickstream examination of different sorts
Market and Advertise Analysis	:	Advertising investigation to think about past method
Travel and tour analysis	:	Online travel booking examinations for best arrangement
Energy Analysis	:	Vitality disclosure and vitality funds investigation
Infrastructure Analysis	:	Foundation administration investigation
Fraud Analysis	:	Extortion identification and investigation
Health Analysis	:	Human services examination for patients
Geo-Location Analysis	:	Tracks different sorts of information, for example, Geo-area information, machine and sensor information, online networking information.

Comparison of Hive and Map reduce

Table 2: Comparison of Hive and MapReduce [41,42,43]

	Mapreduce	Hive
Strength	• works both on organized and unstructured information. • good for composing complex business rationale. • Don't concur with "difficult to accomplish join usefulness", on the off chance that you comprehend what sort of go along with you need to execute you can actualize with few lines of code. • Most of the times MR yields better execution. • MR support for various leveled information is incredible particularly execute tree like structures. Better control at apportioning/indexing the information. Employment	• less improvement time. • suitable for specially appointed investigation. • easy for joins • Sql like Data-base gentlemen love that. • Good support for organized information. • Currently bolster database blueprint and perspectives like structure Support simultaneous multi clients, multi session situations. Greater group support. • Hive , Hiver server , Hiver Server2, Impala, Centry etc.
Weakness	• long advancement sort • hard to accomplish join usefulness • Need to know programming interface exceptionally well to show signs of improvement execution and so forth Code/troubleshoot/maintain	• not simple for complex business rationale. • deals just organized information. • Performance debases as information becomes greater very little to do, memory over stream issues. Can't do much with it. • Hierarchical information is a test. • Unstructured information obliges udf like part Combination of different strategies could be a bad dream element bits with UTDF if there should be an occurrence of enormous in

Methodology

Hadoop has some vital design records [19, 20, 40] that should be considered while arranging the Hadoop. Couple of modest bunch noteworthy well known design records are given beneath with little portrayal for each:

- hadoop-env.sh: It is used as Environment variables utilized for a part of scripts to run Hadoop.
- core-site.xml: It is used to contains settings of Configuration for Core Hadoop, for example, Input - Output settings which give basic on HDFS and MapReduce.
- hdfs-site.xml: It is used for confiugration and design daemon sets for HDFS: the datanodes, the namenode, with their auxiliary namenode.
- mapred-site.xml: It is used to store Setup daemons sets for MapReduce: the tasktrackers with their jobtracker.
- masters: It is set of running machines that every run an auxiliary namenode.
- slaves: It is set of running machines that every run a datanode and a tasktracker.

Hadoop structure is transcribed for successively submissions in Java on expansive groups of thing equipment and consolidates highlights i.e. MapReduce and of the GFS (Google File System) processing ideal model. HDFS can be profoundly blame easy-going appropriated document framework such as Hadoop when all is said in done, intended to be sent on ease equipment. It gives great quantity right to use on applications information which are appropriate used for claims for expansive information collections [37, 38, 39].
The fundamental objective of this paper is establishment for basic Hadoop for analyzing its application with Hive and MapReduce to take in real world.
Above all else prerequisite java development kit on the operating system (Ubuntu) working framework later present and actualize a pseudo-disseminated mode in Hadoop. Subsequently the Hadoop is completed effectively, watch its related all hubs i.e. datanode, optional namenode, namenode, asset supervisor and errand director are supporting effectively or not to Hadoop. It can likewise patterned its web interfaces. In the event that all supports well then it is using Book Crossing downloaded Datasets that are online accessible [15, 36, 37].
This database made out of 3 tables in dataset as illustrated below -

- BX-Users – This is containing clients data that incorporates 'Client ID' or User IDs in addition mapping to whole numbers. On the off chance that any demographic information is accessible then it is likewise given as 'Area', 'Age'. On the off chance that this cannot accessible which give NULL qualities in the contained field.
- BX-Books – This is containing Books complete dataset that are recognized utilizing ISBN number as unique identity. This likewise comprises additional data about Books such as yearofpublication, bookauthor,'booktitle, distributer, imageurll', imageurlm, imageurls. Pictures size can accessible in different senses i.e. huge, little and medium.
- BX-Book-Ratings – This is containing Books evaluation data. This dataset may be communicated unequivocal or understood.

From the over 3 tables dataset of BX-Books are used in this paper. Subsequent to copying the Books dataset, establishment for hive execution can be used and after that execution question for utilizing order line on hive running. This is demonstrating near investigation of MapReduce and Hive. What's more, at last demonstrates that hive is superior to anything for performing on MapReduce information investigation [16, 33, 34].

Database Description

The Dataset of Book Crossing is used as database for our exploration. Cai-Nicolas Zeigler have done to accumulate these dataset with proper support of Ron Hornbaker working as Humankind frameworks Chief Technical Officer. These datasets are containing 278,858 clients data, in addition having silent and unambiguous evaluations that speak the truth of 1,149,780 ratings around 271,379 books. This can accessible online for the examination [17, 31, 32].
These datasets are containing 3 tables as discussed are below -

- BX-Books
- BX-Book-Ratings
- BX-Users

Be that as it may, for our exploration we utilize datasets of BX-Books. This includes Books information such as yearofpublication, booktitles, bookauthors, ISBN numbers and distributers. Likewise it incorporate 3 unique flavors of size for pictures such as imageurl-l, imageurl-m, imageurl-s.

Execution steps

For the execution of hive, we oblige the accompanying essentials [19, 29, 30]-

- Understanding of java and Operating system(Ubuntu)
- Check for java installation.
- Check for stable release of hadoop.
- Download dataset for Book Crossing for analysis performance [4, 28].

The paper here will depict the obliged strides for setting up a solitary hub Hadoop group upheld on Hadoop's

HDFS, administration on Linux Operating System such as Ubuntu. HDFS can be profoundly blame easy-going circulated record framework in addition to Hadoop when all is said in done, intended to be sent on minimal effort equipment.

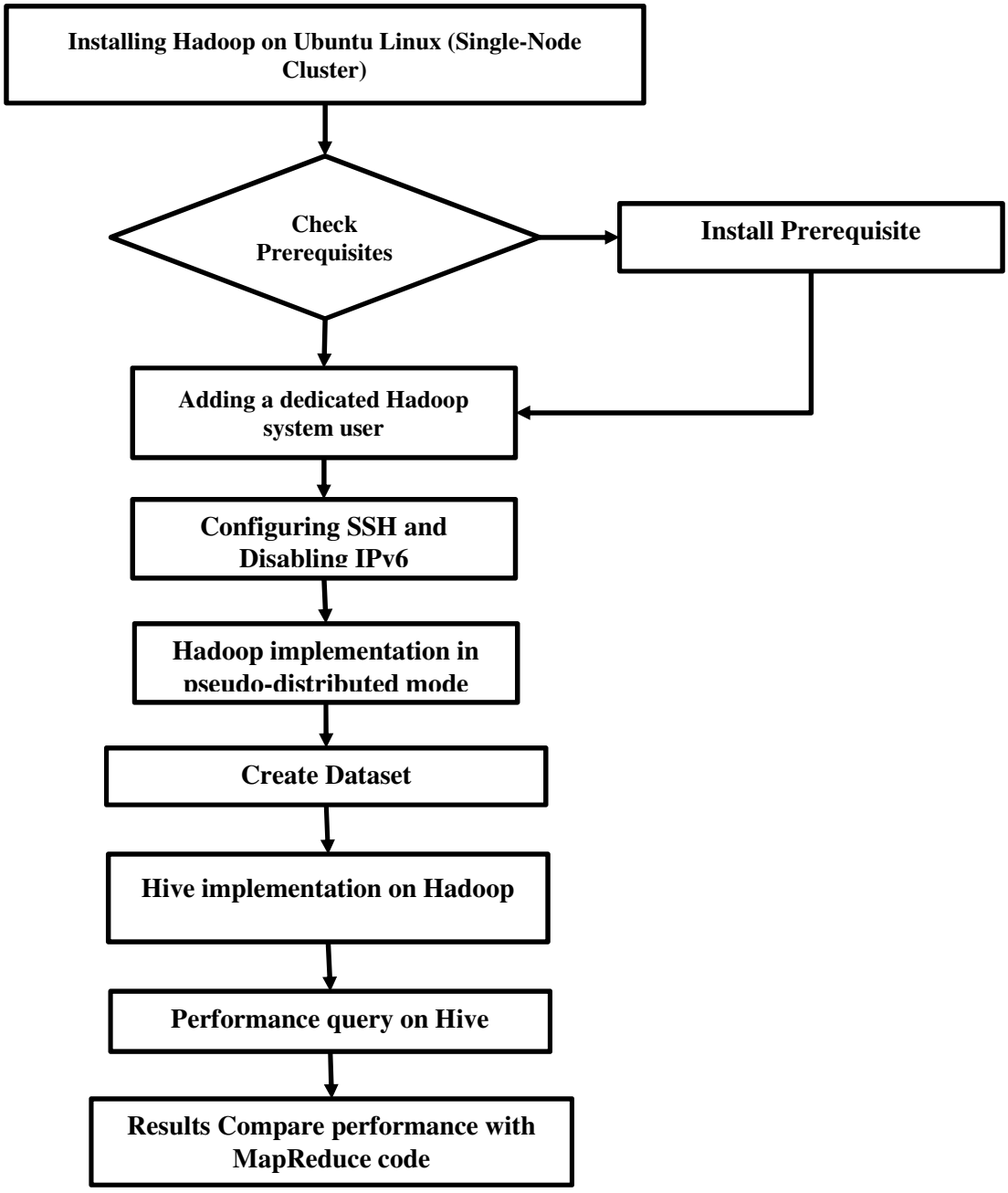

Figure 3: Design of Implementation Hive and MapReduce

Prerequisites and Installation of Hadoop

(1) Java Development Kit(JDK) 6

Hadoop obliges an operational Java 1.5+ (otherwise known as Java 5) establishment.

(2) Update the source list

```
-$  sudo apt-get update
```

(3) Install Sun Java 6 JDK

```
user@ubuntu:~$ sudo apt-get install sun-java6-jdk
```


The complete Java Development Kit which can be set to following directory /usr/lib/jvm/java-6-openjdk-amd64 afterwards establishment, java JDK is effectively introduced or not for rechecking, with the accompanying full path.

```
user@ubuntu:~$ java -version
```

(4) Require to add a system user for dedicated Hadoop
This resolve to utilize a devoted Hadoop client represent running environments Hadoop.

```
user@ubuntu:~$ sudo addgroup hadoop_group
user@ubuntu:~$ sudo adduser --ingroup hadoop_group hduser1
```

This will include the client hduser1 and the gathering hadoop_group to the nearby machine. Add hduser1 to the sudo bunch:

```
user@ubuntu:~$ sudo adduser hduser1 sudo
```

(5) Configuring SSH
The hadoop control scripts depend on SSH to perform bunch wide operations. For instance, there is a script for halting and beginning every one of the daemons in the groups. To work flawlessly, SSH should be setup to permit on hadoop client with secret key for less login towards bunch of other machines. The most straight forward technique to accomplish by creating a private / public pair of key, which will be shared over the bunch.
Hadoop obliges SSH admittance to deal with their hubs, i.e. end computer in addition to other neighborhood computer. For one-node working set on Hadoop, this way essential to design SSH admittance for the hduser client to localhost this is made by earlier instruction.
We need to create a SSH key for the hduser client.

```
user@ubuntu:~$ su - hduser1
hduser1@ubuntu:~$ ssh-keygen -t rsa -P ""
```

The next instruction is making a RSA pair for crypt key using a vacant watchword.
We need to empower SSH admittance to other nearby computer using recently made key pair which is finished by the accompanying instruction.

```
hduser1@ubuntu:~$   cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

The last instruction is testing setup of SSH by interfacing with the neighborhood machine with the hduser1 client. The stride is additionally expected to spare other nearby host key of that machine's unique finger impression with hduser client's identified by hosts document.

```
hduser@ubuntu:~$ ssh localhost
```

(6) Main Hadoop Installation
We will begin by changing to Hadoop hduser.

```
hduser@ubuntu:~$ su - hduser1
```

We download and extricate Hadoop 1.2.0 and Setup Environment Variables for Hadoop in .bashrc record by including after instruction.

```
# Set Hadoop-related environment variables
export HADOOP_HOME=/usr/local/hadoop
# Add Hadoop bin/ directory to PATH
export PATH= $PATH:$HADOOP_HOME/bin
```

(7) Configuration hadoop-env.sh
Change the file: conf/hadoop-env.sh

```
#export JAVA_HOME=/usr/lib/j2sdk1.5-sun

# export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64  (for 64 bit)
# export JAVA_HOME=/usr/lib/jvm/java-6-openjdk-amd64  (for 32 bit)
```

(8) Configuration conf/core -site.xml
We make the registry and set the obliged proprietorships and consents.


```
hduser@ubuntu:~$ sudo mkdir -p /app/hadoop/tmp
hduser@ubuntu:~$ sudo chown hduser:hadoop /app/hadoop/tmp
hduser@ubuntu:~$ sudo chmod 750 /app/hadoop/tmp
```

We paste the following between <configuration> in file conf / core -- site . xml

```
<property>
    <name>hadoop.tmp.dir</name>
    <value>/app/hadoop/tmp</value>
    <description>A base for other temporary directories.</description>
</property>

<property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:54310</value>
    <description>The name of the default file system.  A URI whose
    scheme and authority determine the FileSystem implementation.  The
    uri's scheme determines the config property (fs.SCHEME.impl) naming
    the FileSystem implementation class.  The uri's authority is used to
    determine the host, port, etc. for a filesystem.</description>
</property>
```

(9) Configuration conf / mapred -- site . xml

```
<property>
<name>mapred.job.tracker</name>
    <value>localhost:54311</value>
    <description>The host and port that the MapReduce job tracker runs
    at.  If "local", then jobs are run in-process as a single map
    and reduce task.
    </description>
</property>
```

(10) Configuration conf / hdfs – site . xml

```
<property>
    <name>dfs.replication</name>
    <value>1</value>
    <description>Default block replication.
    The actual number of replications can be specified when the file is created.
    The default is used if replication is not specified in create time.
    </description>
</property>
```

(11) Formatting the file system as HDFS via requirement of the NameNode

To arrange the filesystem (which just introduces the catalog indicated variable as dfs . name . dir). Follow the execution in this order:

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode -format
```

(12) Starting your single-node cluster

Before beginning the bunch, we have to give the obliged authorizations to the registry with the accompanying order.

```
hduser@ubuntu:~$ sudo chmod -R 777 /usr/local/hadoop

hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
```



```
hduser@ubuntu:/usr/local/hadoop$ jps
```

(13) This is providing additionally Hadoop's web interfaces using localhost. Along these lines, this will likewise entrance on Hadoop program via browser. Hadoop will be accessed by 50070 port number. Applying http://localhost:50070 on web browser, this will give administrations on Hadoop program as demonstrated in Figure 15.

(14) Hive installation and implementation

Hive is an Apache data distribution center foundation which constructed upper layer of Hadoop to support data rundown, investigation, and question. Hive underpins exploration of expansive set of data for HDFS to put away and decent record structures or frameworks such as Amazon S3 file system. This will make a HiveQL(Hive Query Language) as similar to SQLwhile keeping up full backing for guide/diminish.

```
user@ubuntu:~$  cd  /usr/lib/
user@ubuntu:~$  sudo mkdir hive
user@ubuntu:~$  cd Downloads
user@ubuntu:~$  sudo mv apache-hive-0.13.0-bin /usr/lib/hive
```

(15) Setting Hive environment variable

```
user@ubuntu:~$  cd
user@ubuntu:~$  sudo gedit  ~/.bashrc
```

We add the following lines at end of the file

```
# Set HIVE_HOME
export HIVE_HOME="/usr/lib/hive/apache-hive-0.13.0-bin"
PATH=$PATH:$HIVE_HOME/bin
export PATH
```

(16) Setting HADOOP_PATH in HIVE config.sh

```
user@ubuntu:~$ cd  /usr/lib/hive/apache-hive-0.13.0-bin/bin
user@ubuntu:~$ sudo gedit hive-config.sh

# Allow alternate conf dir location.
HIVE_CONF_DIR="${HIVE_CONF_DIR:-$HIVE_HOME/conf"
export HIVE_CONF_DIR=$HIVE_CONF_DIR
export HIVE_AUX_JARS_PATH=$HIVE_AUX_JARS_PATH

export HADOOP_HOME=/usr/local/hadoop
```

(17) Create Hive directories within HDFS

```
user@ubuntu:~$   hadoop fs -mkdir /usr/hive/warehouse
```

(18) Setting READ/WRITE permission for table

```
user@ubuntu:~$  hadoop fs -chmod g+w /usr/hive/warehouse
```

(19) HIVE launch

```
user@ubuntu:~$  hive
```

(20) Creating a sample database

```
hive> create database mydb;
```

Output

```
OK
Time taken: 0.369 seconds
hive>
```

(21) Configuring hive-site.xml:

Open with text-editor and change the following property


```
<property>
    <name>hive.metastore.local</name>
    <value>TRUE</value>
    <description>controls whether to connect to remove metastore server or open a new metastore s
</property>

<property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://usr/lib/hive/apache-hive-0.13.0-bin/metastore_db? createDatabaseIfNotExis
    <description>JDBC connect string for a JDBC metastore</description>
</property>

<property>
    <name>javax.jdo.option.ConnectionDriverName</name>
    <value>com.mysql.jdbc.Driver</value>
    <description>Driver class name for a JDBC metastore</description>
</property>

<property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/usr/hive/warehouse</value>
    <description>location of default database for the warehouse</description>
 </property>
```

(22) Create tmp and warehouse directory for Hive

```
~$ hadoop fs –mkdir –p /tmp
~$ hadoop fs –mkdir –p /usr/hive/warehouse
~$ hadoop fs – chmod g + w  / tmp
~$ hadoop fs – chmod g + w  / usr / hive /
```

(23) Copy dataset to Hive directory

```
~$ sudo mkdir /usr/local/hive/data
~$  sudo  cp BX-CSV-Dump.zip /usr/local/hive/data
~$ sudo unzip BX-CSV-Dump.zip
```

(24) Extract dataset into Hive directory

```
~$  sed ' s/ \&amp; / \ &/ g'  BX-Books.csv |  sed – e  '1d' | sed
's / ; / $ $ $ / g' | sed  's / " $ $ $ " / " ; " / g' |  sed  ' s
/ " / / g' >  BX-BooksCorrected1.txt
```

This command will remove header line, “characters and also replace “&” to “&”, “;” to “$$$”, “$$$” to “;”.

(25) Create Input directory into Hive directory

```
~$ hadoop fs –mkdir –p input
~$  hadoop  fs  –put  /usr/local/hive/data/BX-BooksCorrected1.txt
input
```

(26) Run Hive

```
Logging initialized using configuration in jar:file:/usr/local/
hive/lib/hive-common-1.0.1.jar!/hive-log4j.properties
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop-2.7.0/share
/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/Sta
ticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/hive-jdbc
-1.0.1-standalone.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings fo
r an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFac
tory]
hive>
```

Figure 4: Running Hive in HDFS

(27) Create Dataset for Hive


```
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings fo
r an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFac
tory]
hive> create table if not exists bxdataset
    > (isbn string,
    > booktitle string,
    > bookauthor string,
    > yearofpublication string,
    > publisher string)
    > row format delimited fields terminated by '\;'
    > stored as textfile;
OK
Time taken: 1.315 seconds
hive> load data inpath '/user/hduser/input/BX-BooksCorrected1.t
xt' overwrite into table bxdataset;
Loading data to table default.bxdataset
Table default.bxdataset stats: [numFiles=1, numRows=0, totalSiz
e=77832312, rawDataSize=0]
OK
Time taken: 1.708 seconds
hive> select yearofpublication, count(booktitle) from bxdataset
 group by yearofpublication;
```

Figure 5: Creating dataset for Hive

Result:

Hadoop can be utilized for quantify of the large dataset to CPU run-time, scale-out, scale-up. This will provide impact of CPU scedular on runtime when it provide performance tunning to the quantify for each dataset for reducers.

```
hive> select yearofpublication, count(booktitle) from bxdataset
 group by yearofpublication;
Query ID = hduser_20150629162929_66b535ac-29aa-407a-bba0-0060a3
29e61a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data
 size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Job running in-process (local Hadoop)
Hadoop job information for Stage-1: number of mappers: 0; numbe
r of reducers: 0
2015-06-29 16:29:05,615 Stage-1 map = 0%,  reduce = 0%
2015-06-29 16:29:08,990 Stage-1 map = 100%,  reduce = 100%
Ended Job = job_local122125797_0001
MapReduce Jobs Launched:
Stage-Stage-1:  HDFS Read: 155664638 HDFS Write: 0 SUCCESS
Total MapReduce CPU Time Spent: 0 msec
```

Figure 6: Display result dataset using Hive

```
Total MapReduce CPU Time Spent: 0 msec
OK
"0"     4619
"1376"  1
"1378"  1
"1806"  1
"1897"  1
"1900"  3
"1901"  7
"1902"  2
"1904"  1
"1906"  1
"1908"  1
"1909"  2
"1910"  1
"1911"  19
"1914"  1
"1917"  1
"1919"  1
"1920"  33
"1921"  2
"1922"  2
"1923"  11
```

Figure 7: Display MapReduce result

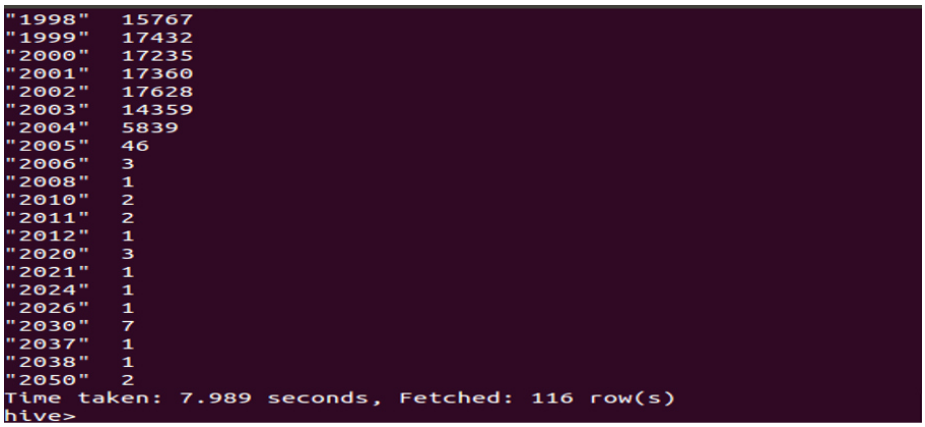

Figure 8: Display MapReduce result for CPU time

Comparison of MapReduce with Hive code

When we execute the similar job with Java based code of MapReduce for books project. The impact of result will be achieved on above Figures. Here we add package books and import many existing package such as java(util.Iterator, io.IOException), org.apache.hadoop(fs.Path, io(IntWritable,Text, LongWritable), mapred(FileInputFormat, FileOutputFormat, JobClient, JobConf, MapReduceBase, JobConf, OutputCollector, Reducer, Mapper, Reporter, TextOutputFormat, TextInputFormat). We create a class with name of BookFrequency , sub static class BooksXMapper which extends MapReduceBase with implements Mapper. We override map function with OutputCollector(Text, IntWriteable), value as Text, key as LongWritable, The Java code is used to compute the CPU Time as below -

```
public class Books_Frequency
{  public static class Books_XMapper extends MapReduceBase implementsMapper<LongWritable,  Text,  Text,
IntWritable>
    {  @Override
        public void map(LongWritable _key, Text _value,
        OutputCollector<Text, IntWritable> _output, Reporter _reporter)
                throws IOException {
            String _temp = value.toString();
            String[] _SingleBookdata = temp.split("\";\"");
            output.collect(new Text(SingleBookdata[3]),new IntWritable(1));
        }
    }
    public static class Books_XReducer extends MapReduceBase implements Reducer<Text,  IntWritable,  Text,
IntWritable>
    {
        @Override
        public void reduce(Text _key, Iterator<IntWritable> _values,
                OutputCollector<Text, IntWritable> _output, Reporter _reporter)
                throws IOException {
            int _bookfreq = 0;
            while(_values.hasNext())
            {
                IntWritable _value = (IntWritable)values.next();
                _bookfreq = _bookfreq + _value.get();
            }
            output.collect(_key, new IntWritable(_bookfreq));
        }
    }
    public static void main(String[] args) throws Exception
    {
        JobConf _newconf = new JobConf();
        _newconf.setJarByClass(BooksFrequency.class);
 _newconf.setJobName("bookfrequency");
```



```
  _newconf.setOutputKeyClass(Text.class);
  _newconf.setOutputValueClass(IntWritable.class);
  _newconf.setMapperClass(BooksXMapper.class);
  _newconf.setReducerClass(BooksXReducer.class);
 //_newconf.setCombinerClass(BooksXReducer.class);
   _newconf.setInputFormat(TextInputFormat.class);
 _newconf.setOutputFormat(TextOutputFormat.class);
 FileInputFormat.setInputPaths(_newconf, new Path(args[0]));
 FileOutputFormat.setOutputPath(_newconf, new Path(args[1]));
 JobClient.runJob(_newconf);
      }
}
```

This code will give clear exertion about Hive to take less effort time due to less effort for MapReduce programming in both case of learning and writing code. When Hive code is compared above code, this give the less effort of coding as three lines required whereas MapReduce take about 25 lines code.

Conclusion

Big-Data is presented as a brief beginning in this paper. Enormous information can convey significant advantages to the business. At that point the paper talks about around one of the advances that handle the Big Data, the Hadoop. At that point paper discusses HDFS and MapReduce programming model. At that point we discuss a few utilizations of Hadoop. At that point the different schedulers utilized as a part of Hadoop are talked about to sum things up. We investigate specialized parts of Hadoop where some imperative setup records of Hadoop are talked about. Since MapReduce experiences some critical confinements if the quantities of hubs are expanded, the innovation that conquers this constraint is YARN which is quickly talked about. MapReduce code is bigger in terms of number of code line. MapReduce and Hive comparison's show the best running performance. Hive line code is less number of line in compare to MapReduce. Hive take less execution time in compare to MapReduce. This is proving that Hive is better than MapReduce.

References

[1] Andrew Becherer, "Hadoop Security Design", iSEC Partners, Inc.
[2] Hrishikesh Karambelkar, "Scaling Big Data with Hadoop and Solr", Birmingham B3 2PB, UK, 2013.
[3] "Big Data : getting Started with Hadoop, Sqoop & Hive, available" , [online] available on http://cloudacademy.com/blog/big-data-getting-started-with-hadoop-sqoop-hive/
[4] David Floyer, "Enterprise Big-data" , [online] available on http://wikibon.org/wiki/v/Enterprise_Big-data.
[5] Benjamin Bengfort, Jenny kim, "Creating a Hadoop Pseudo-Distributed Environment" , [online] available on https://districtdatalabs.silvrback.com/creating-a-hadoop-pseudo-distributed-environment
[6] Koen Vlaswinkel, "How to install java on Ubuntu with Apt-Get" , [online] available on https://www.digitalocean.com/community/tutorials/how-to-install-java-on-ubuntu-with-apt-get
[7] Mathias Kettner, "SSH login without password" , [online] available on http://www.linuxproblem.org/art_9.html
[8] Dustin Kirkland, "Ubuntu manuals" , [online] available on http://manpages.ubuntu.com/manpages/raring/man1/nvi.1.html
[9] Varad Meru, "Single-Node hadoop tutorial" , [online] available on http://www.orzota.com/single-node-hadoop-tutorial/
[10] Michael G. Noll, "Running Hadoop on Ubuntu Linux(Single-Node Cluster)" , [online] available on http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/
[11] "Hadoop 2.6 Installing on Ubuntu 14.04 (Single-Node Cluster) – 2015" , [online] available on http://www.bogotobogo.com/Hadoop/BigData_hadoop_Install_on_ubuntu_single_node_cluster.php
[12] Apache hadoop, "Single Node Setup" , [online] available on https://hadoop.apache.org/docs/r1.2.1/single_node_setup.html
[13] "Apache Hadoop (CDH 5) Hive Introduction – 2015" , [online] available on http://www.bogotobogo.com/Hadoop/BigData_hadoop_CDH5_Hive_Introduction.php
[14] Oshin Prem, "HIVE Installation" , [online] available on http://doctuts.readthedocs.org/en/latest/hive.html
[15] Edureka, "Apache Hive Installation on Ubuntu" , [online] available on http://www.edureka.co/blog/apache-hive-installation-on-ubuntu/
[16] "Hive Installation On Ubuntu" , [online] available on https://archanaschangale.wordpress.com/2013/08/24/hive-installation-on-ubuntu/
[17] Safari, "Programming Hive" , [online] available on https://www.safaribooksonline.com/library/view/programming-hive/9781449326944/ch04.html
[18] "Hive – Introduction" , [online] available on http://www.tutorialspoint.com/hive/hive_introduction.htm
[19] Jason Dere, "LanguageManualDDL" , [online] available on https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL
[20] "Hadoop material", [online] available on http://www.hadoopmaterial.com/2013/10/find-frequency-of-books-published-in.html
[21] B.Thirumala Rao, Dr.L.S.S.Reddy, "Survey on Improved Scheduling in Hadoop MapReduce in Cloud Environments", International Journal of Computer Applications (0975 – 8887) Volume 34– No.9, November 2011.
[22] Kala Karun. A, Chitharanjan.K, "A Review on Hadoop – HDFS Infrastructure Extensions", Proceedings of 2013 IEEE Conference on Information and Communication Technologies (ICT 2013).
[23] Lizhe Wang, Jie Tao, Rajiv Ranjan , Holger Marten, AchimStreit , Jingying Chen, DanChen, "G-Hadoop: MapReduce across distributed data centers for data-intensive computingl, 2013.
[24] WeijiaXu, Wei Luo, Nicholas Woodward, "Analysis and Optimization of Data Import with Hadoop", 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum.
[25] HUANG Lan, WANG Xiao-wei, ZHAI Yan-dong, YANG Bin, "Extraction of User Profile Based on the Hadoop Framework".
[26] Jeffy Dean,Sanjay Ghemawat. MapReduce, "Simplified Data Processing on Large Clusters", OSDI04: Sixth Symposium on Operating System Design and Implemention, Ssn Francisco, CA,December, 2004.
[27] What is Apache Hadoop? https://hadoop.apache.org
[28] Introduction to the Hadoop Software Ecosystem. http://www.revelytix.com/?q=content/hadoop-ecosystem
[29] 10 ways companies are using Hadoop (for more than ads). https://gigaom.com/2012/06/05/10-ways-companies-are-usinghadoop-
[30] to-do-more-than-serve-ads/
[31] Business Applications of Hadoop. http://www.edureka.in/blog/business-applications-of-hadoop/
[32] Job Scheduling. Hadoop: The Definitive Guide, Third Edition, Textbook.

[33] Hadoop's Fair Scheduler. https://hadoop.apache.org/docs/r1.2.1/fair_scheduler.
[34] Jagmohan Chauhan, Dwight Makaroff and Winfried Grassmann, "The Impact of Capacity Scheduler Configuration Settings on MapReduce Jobs".
[35] Hadoop's Capacity Scheduler. http://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html.
[36] M. Zaharia, A. Kowinski, A. Joseph, R. Katz and I. Stoica, "Improving mapreduce performance in heterogeneous environments." USENIX OSDI, 2008
[37] DongjinYoo, Kwang Mong Sim, "A comparative review of job scheduling for mapreduce,‖ Multi-Agent and Cloud Computing Systems Laboratory, Proceedings of IEEE CCIS2011.
[38] Li Liu, Yuan Zhou, Ming Liu, Guandong Xu, Xiwei Chen, Dangping Fan, Qianru Wang,"Premptive Hadoop Jobs Scheduling under a Deadline", IEEE 2012 Eighth International Conference on Semantics, Knowledge and Grids.
[39] Hadoop Administrative Guide. http://caen.github.io/hadoop/administration-hadoop.html.
[40] Hadoop Cluster Configuration Files. http://www.edureka.in/blog/hadoop-cluster-configuration-files.
[41] Arinto Murdopo, Jim Dowling, "Next Generation Hadoop: High Availability for YARN."
[42] Vinod Kumar Vavilapalli, Arun C Murthy, Chris Douglas,SharadAgarwal, MahadevKonar, Robert Evans, Thomas Graves, Jason Lowe, Hitesh Shah, Siddharth Seth, BikasSaha, Carlo Curino, Owen O'Malley, Sanjay Radia, Benjamin, Reed, Eric Baldeschwieler, "Apache Hadoop YARN: Yet Another Resource Negotiator."
[43] Hortonworks Hadoop YARN. http://hortonworks.com/hadoop/yarn/.

Appendix-A : Screenshot of steps for Hadoop implementation.

```
Ign http://in.archive.ubuntu.com trusty/main Translation-en_IN
Ign http://in.archive.ubuntu.com trusty/multiverse Translation-en_IN
Ign http://in.archive.ubuntu.com trusty/restricted Translation-en_IN
Ign http://in.archive.ubuntu.com trusty/universe Translation-en_IN
Ign http://in.archive.ubuntu.com trusty-updates/main Translation-en_IN
Ign http://in.archive.ubuntu.com trusty-updates/multiverse Translation-en_IN
Ign http://in.archive.ubuntu.com trusty-updates/restricted Translation-en_IN
Ign http://in.archive.ubuntu.com trusty-updates/universe Translation-en_IN
Ign http://in.archive.ubuntu.com trusty-backports/main Translation-en_IN
Ign http://in.archive.ubuntu.com trusty-backports/multiverse Translation-en_IN
Ign http://in.archive.ubuntu.com trusty-backports/restricted Translation-en_IN
Ign http://in.archive.ubuntu.com trusty-backports/universe Translation-en_IN
Fetched 1,206 kB in 1min 32s (13.0 kB/s)
Reading package lists... Done
```

Figure A1: Update the source list

```
sunick@sunick-HP-Pavilion-15-Notebook-PC:~$ java -version
java version "1.6.0_31"
OpenJDK Runtime Environment (IcedTea6 1.13.3) (6b31-1.13.3-1ubuntu1)
OpenJDK 64-Bit Server VM (build 23.25-b01, mixed mode)
sunick@sunick-HP-Pavilion-15-Notebook-PC:~$
```

Figure A2: Check java version

```
sunick@sunick-HP-Pavilion-15-Notebook-PC:~$ sudo addgroup hadoop_group
Adding group `hadoop_group' (GID 1003) ...
Done.
sunick@sunick-HP-Pavilion-15-Notebook-PC:~$
```

Figure A3: Adding the indicated Hadoop group

```
sunick@sunick-HP-Pavilion-15-Notebook-PC:~$ sudo adduser --ingroup hadoop_group hduser1
Adding user `hduser1' ...
Adding new user `hduser1' (1003) with group `hadoop_group' ...
Creating home directory `/home/hduser1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for hduser1
Enter the new value, or press ENTER for the default
        Full Name []: oshin
        Room Number []: 23
        Work Phone []: 23456
        Home Phone []: 234567
        Other []: 89
Is the information correct? [Y/n] Y
sunick@sunick-HP-Pavilion-15-Notebook-PC:~$
```

Figure A4: Adding dedicated user

```
hduser1@sunick-HP-Pavilion-15-Notebook-PC:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser1/.ssh/id_rsa):
Created directory '/home/hduser1/.ssh'.
Your identification has been saved in /home/hduser1/.ssh/id_rsa.
Your public key has been saved in /home/hduser1/.ssh/id_rsa.pub.
The key fingerprint is:
79:8a:40:cc:67:3e:71:ad:c6:44:b4:b7:e2:fa:77:0e hduser1@sunick-HP-Pavilion-15-Notebook-PC
The key's randomart image is:
+--[ RSA 2048]----+
|       .o        |
|   o   . o       |
|    + + + o      |
|   . + = + .     |
|    . o S o      |
|     . = +       |
|      . o E      |
|       .  ...    |
|      .... o.    |
+-----------------+
hduser1@sunick-HP-Pavilion-15-Notebook-PC:~$
```

Figure A5: Configuring SSH


```
hduser1@sunick-HP-Pavilion-15-Notebook-PC:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
hduser1@sunick-HP-Pavilion-15-Notebook-PC:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is b0:29:0c:ee:0f:4a:f5:ee:84:70:4a:c3:ad:62:b2:70.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

13 packages can be updated.
0 updates are security updates.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

hduser1@sunick-HP-Pavilion-15-Notebook-PC:~$
```

Figure A6: SSH login on Localhost

```
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
or implied.
  See the License for the specific language governing permissio
ns and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
     <name>hadoop.tmp.dir</name>
     <value>/app/hadoop/tmp</value>
</property>
<property>
     <name>fs.default.name</name>
     <value>hdfs://localhost:54310</value>
</property>

</configuration>
~
```

Figure A7: Core property configuration in conf directory

```
  Unless required by applicable law or agreed to in writing, so
ftware
  distributed under the License is distributed on an "AS IS" BA
SIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
or implied.
  See the License for the specific language governing permissio
ns and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
    <name>mapreduce.jobtracker.address</name>
    <value>localhost:54311</value>
</property>

</configuration>
```

Figure A8: mapred property configuration in conf directory

```
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

<property>
  <name>dfs.replication</name>
  <value>3</value>
</property>
<property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
</property>
<property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
</property>

</configuration>
~
```

Figure A9: hdfs property configuration in conf directory

```
usr/local/hadoop/share/hadoop/common/lib/protobuf-java-2.5.0.ja
r:/usr/local/hadoop/share/hadoop/common/lib/jettison-1.1.jar:/u
sr/local/hadoop/share/hadoop/common/lib/curator-recipes-2.7.1.j
ar:/usr/local/hadoop/share/hadoop/common/lib/java-xmlbuilder-0.
4.jar:/usr/local/hadoop/share/hadoop/common/lib/commons-math3-3
.1.1.jar:/usr/local/hadoop/share/hadoop/common/lib/hadoop-auth-
2.7.0.jar:/usr/local/hadoop/share/hadoop/common/lib/snappy-java
-1.0.4.1.jar:/usr/local/hadoop/share/hadoop/common/lib/jackson-
core-asl-1.9.13.jar:/usr/local/hadoop/share/hadoop/common/lib/j
unit-4.11.jar:/usr/local/hadoop/share/hadoop/common/lib/htrace-
core-3.1.0-incubating.jar:/usr/local/hadoop/share/hadoop/common
/lib/jackson-mapper-asl-1.9.13.jar:/usr/local/hadoop/share/hado
op/common/lib/commons-codec-1.4.jar:/usr/local/hadoop/share/had
oop/common/lib/commons-compress-1.4.1.jar:/usr/local/hadoop/sha
re/hadoop/common/lib/api-asn1-api-1.0.0-M20.jar:/usr/local/hado
```


Figure A10: Formatting NameNode in HDFS environmnet

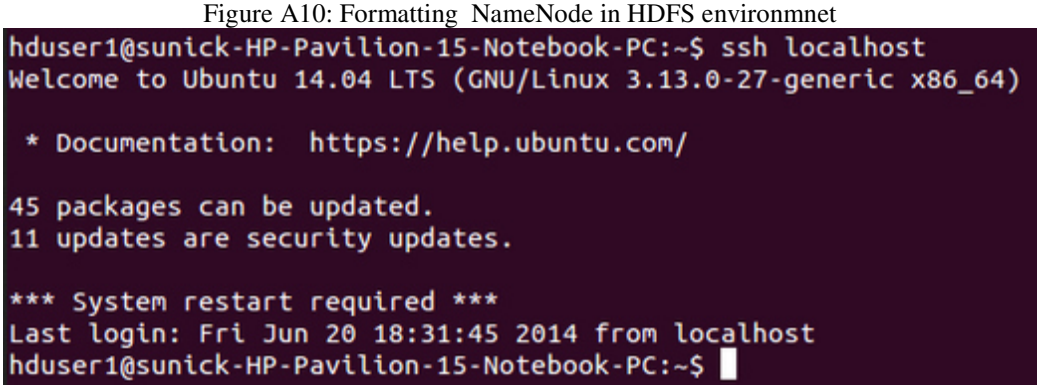

Figure A11: Connecting single node via SSH

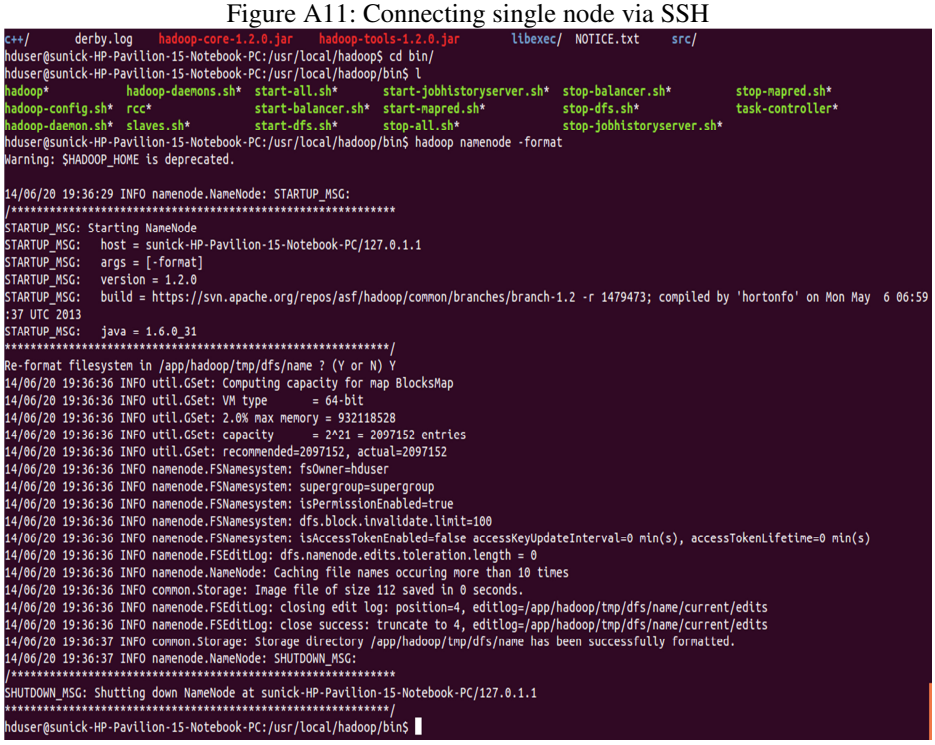

Figure A12: Starting single node of HDFS

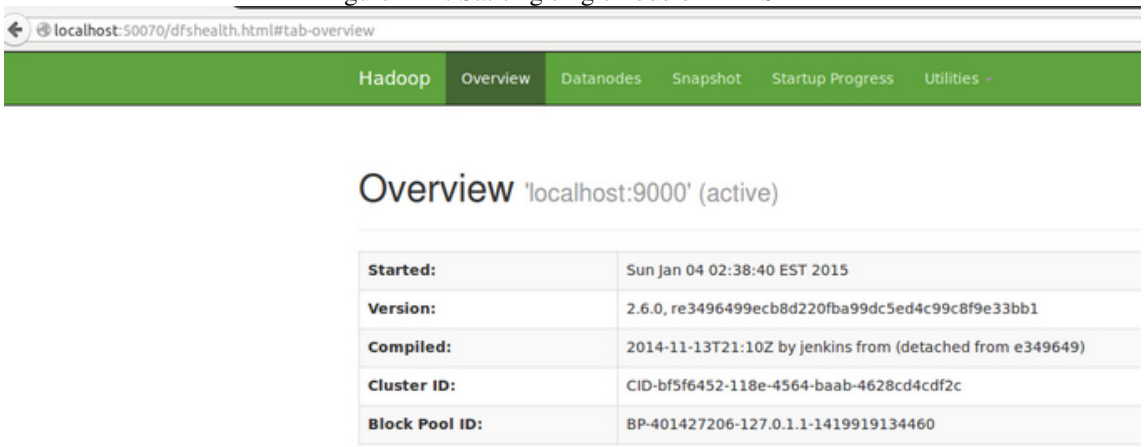

Figure A13: Checking namenode health information