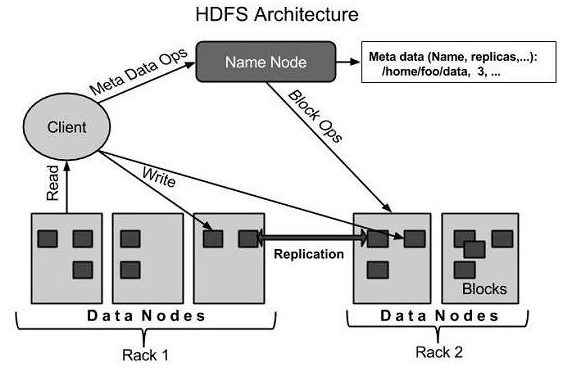
**Architecture of Hadoop Distributed File System**

HDFS stores filesystem metadata and application data separately. An HDFS cluster consists of a single Name node, a master server that manages the file system namespace and regulates access to files by clients. In addition, there are a number of Data nodes, usually one per node

in the cluster, which manage storage attached to the nodes that they run on. HDFS exposes a file system namespace and allows user data to be stored in files.



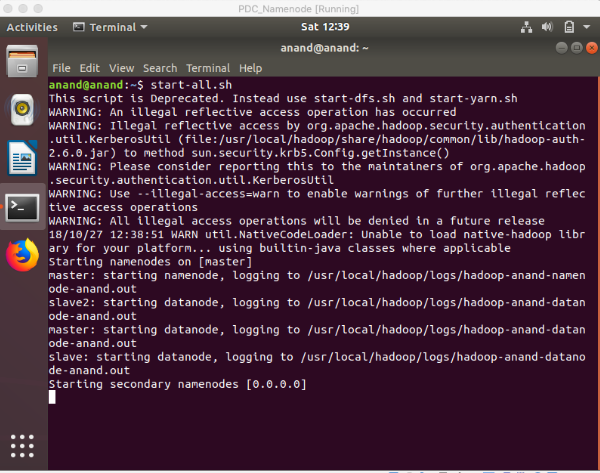
**Implementation of Project**

Unlike other Hadoop projects, we tried to implement the HDFS on three nodes and further use HDFS in our next objective to analyse the system behaviour on the case of fault tolerance. So before implementing HDFS we know that Hadoop is a framework written in Java for running applications on large clusters of commodity hardware and incorporates features similar to those of the [Google File](http://en.wikipedia.org/wiki/Google_File_System) [System (GFS)](http://en.wikipedia.org/wiki/Google_File_System) and of the [MapReduce](http://en.wikipedia.org/wiki/MapReduce) computing paradigm, So implantation of Hadoop HDFS system start with the installation of JAVA 6 on the machine itself.

* Adding a dedicated Hadoop system user
* Configuring SSH-Hadoop requires SSH access to manage its nodes
* Disabling IPV6
* Updating bashrc and JAVA\_HOME configurations
* Configuring Core-Site.xml and mapred-site.xml according to the requirements and the size of the cluster.
* Format the NameNode
* Starting the NameNode and connecting the Datanodes to the Namenodes.

So this part is the beginning of connecton of DataNodes with the NameNode and creating a multi node cluster of three nodes. So, these are the following steps that we had follow:

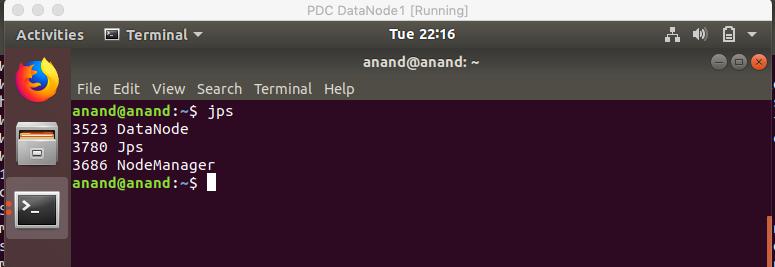
* Multi- Node Hadoop cluster has been created, So to create this we have created three Virtual machines one for DataNode and one for Namenode. So after the creation of **HDFS** in NameNode we have started a command **start-all.sh** in order to start theNamenode and all the other essential components in the Namenode of machine and to establish connection with the DataNode



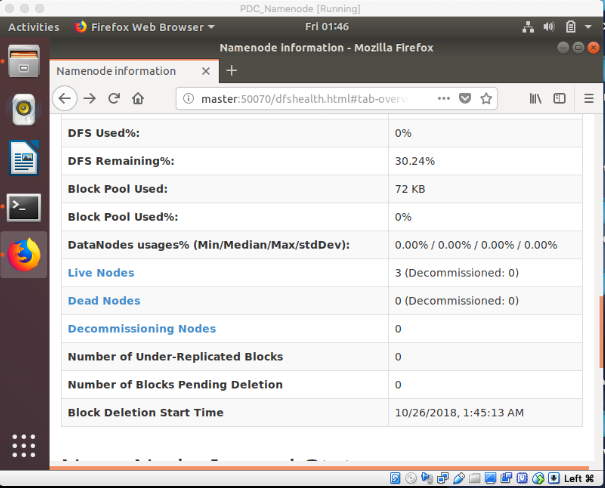
2. As our NameNode started the all other components in the one node i.e



3. In DataNode the components that has been started are:

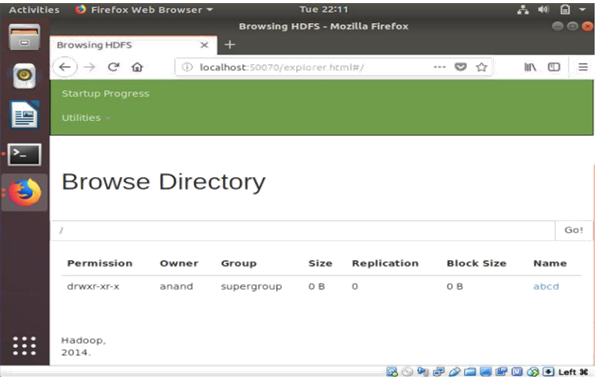


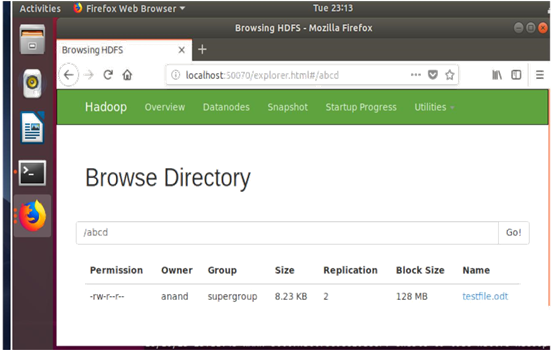
1. So now our NameNode and DataNode has been connected and we have connected 3 nodes here it is:



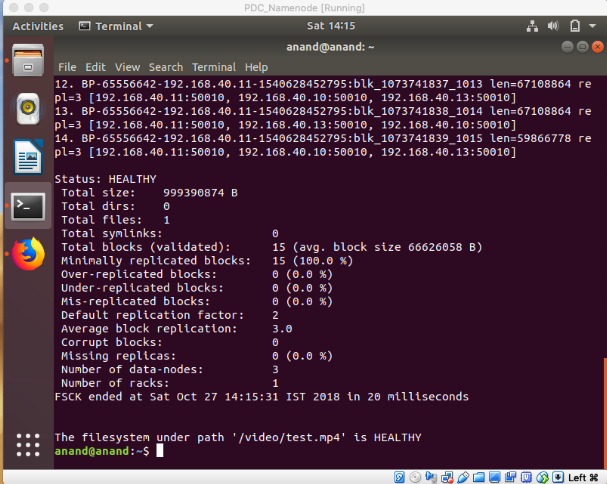
Here Name Node information in the localhost:50070/ is showing that we have connected two node and we have two live nodes at the moment and HDFS implementation is done so far.

1. Now we will transfer file from the NameNode to the DataNode with the help of commands using terminal:
   1. First in the NameNode I will create directory in the DataNode using this command: **hadoop fs** **–mkdir /abcd** which will in hence create the directory in the DataNode which can be shown in the HDFS information in the localhost
   2. Second in the NameNode I will transfer file from the NameNode to the DataNode in the same directory which I have created earlier on and transfer file name **testfile** using this command: **hadoop fs** **–put /home/anand/home/Desktop/testfile.odt** **/abcd**
   3. This will transfer the file in the DataNode directory.

****

****

1. Now as our Nodes are completely connected and transfer of file has been done, So Now our next target is to check the status of nodes and know what happens when Fault tolerance occurs.



**Fault Tolerance:**

Fault tolerance is defined as, when the system functions properly without any data loss even if some hardware components of the system has failed. It is very hard to reach cent percent fault tolerance but faults can be tolerated up to some extent. HDFS provide high throughput to access data application and suitable to have large data sets as their input. The main purpose of this fault tolerance is to remove frequently taking place failures, which occurs commonly and disturbs the ordinary functioning of the system. Single point failure nodes occur when a single node failure causes the entire system to crashes. The primary duty of fault tolerance is to remove such node which disturbs the entire normal functioning of the system. Fault tolerance is one of the major advantages of using Hadoop. The three main solutions which are used to produce fault tolerance are data replication, heartbeat messages and checkpoint and recovery.

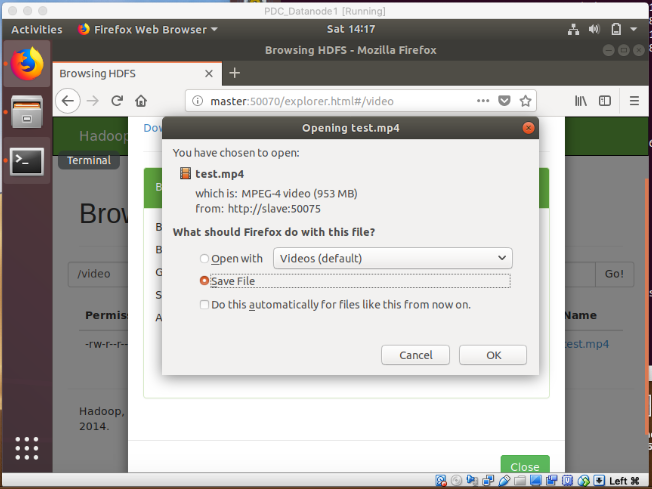
**Replication Factor-**

The **replication factor** is a property that can be set in the HDFS configuration file that will allow you to adjust the global replication factor for the entire cluster. For each block stored in HDFS, there will be **n – 1** duplicated blocks distributed across the cluster. For example, if the replication factor was set to 3 (default value in HDFS) there would be one original block and two replicas.

1. Implementing HDFS with Fault Tolerance using Replication Factor-

Data in HDFS is divided into multiple blocks which have multiple copies on different nodes. So if any node goes down, data retrieval can be done through other nodes. When a file is stored in HDFS it is divided into blocks which is stored in the different nodes of the cluster, By default the replication factor for Hadoop is 3 which means that there are 3 copies of a single block of files stored in different nodes that are selected and managed by HDFS. So as soon as the node goes down the blocks present in that node get replicated in another nodes which is taken care by the name node. However we can change the replication factor to be a value of our choice but since the replication factor is selected to be 1, HDFS makes a single copy of the particular file block which won’t reciprocate fault tolerance. In our demonstration we are able to see that the file system divides the files into blocks of size of 128 MB and distributes them into different salves of the cluster. So even if one of the nodes shits down we can retrieve the file from a different node of the cluster as whole without any loss in the file. However, one of the major complication or a disadvantage in this could be: if suppose the NameNode is corrupted or shuts down unexpectedly we might not be able to retrieve the data since the link between the slave/DataNodes is lost.

Data in HDFS is divided into multiple blocks which have multiple copies on different nodes. So if any node goes down, data retrieval can be done through other nodes. When a file is stored in HDFS it is divided into blocks which is stored in the different nodes of the cluster, By default the replication factor for Hadoop is 3 which means that there are 3 copies of a single block of files stored in different nodes that are selected and managed by HDFS. So as soon as the node goes down the blocks present in that node get replicated in another nodes which is taken care by the name node. However we can change the replication factor to be a value of our choice but since the replication factor is selected to be 1, HDFS makes a single copy of the particular file block which won’t reciprocate fault tolerance. In our demonstration we are able to see that the file system divides the files into blocks of size of 128 MB and distributes them into different salves of the cluster. So even if one of the nodes shits down we can retrieve the file from a different node of the cluster as whole without any loss in the file. However, one of the major complication or a disadvantage in this could be: if suppose the NameNode is corrupted or shuts down unexpectedly we might not be able to retrieve the data since the link between the slave/DataNodes is lost.

****

This figure shows data from Hadoop can be downloaded when Fault Tolerance occurs in HDFS

**References**

[1] Apache Hadoop. http://hadoop.apache.org.

[2] Hadoop Distributed File System <http://hadoop.apache.org/hdfs>.

[3] Borthakur, D. 2007. The Hadoop Distributed File System: Architecture and Design. http://hadoop.apache.org/common/docs/r0.18.0/hdfs\_ design.pdf.

[4] Shvachko, K., et al. 2010. The Hadoop Distributed File System. IEEE. http://storageconference.org/2010/Papers/MSST/Shva chko.pdf.

[5] Apache Hadoop. <http://en.wikipedia.org/wiki/Apache_Hadoop>.

[6] Joey Joblonski. Introduction to Hadoop. A Dell technical white paper. http://i.dell.com/sites/content/business/solutions/white papers/en/Documents/hadoop-introduction.pdf

[7] An introduction to the hadoop distributed file Systemhttp://www.ibm.com/developerworks/library/w a-introhdfs.

[8] Selic, B. 2004. Fault tolerance techniques for distributed systems. IBM. http://www.ibm.com/developerworks/rational/library/ 114.htm.

[9] Jared Evans CSCI B534 Survey Paper. “Fault Tolerance in Hadoop for Work Migration” http://salsahpc.indiana.edu/b534projects/sites/default/files/public/0\_Fault%20Tolerance%20in%20Hadoop% 20for %20Work%20Migration\_Evans, %20Jared%20Matthew.pdf.

[10]The Hadoop Distributed File System: Architecture and Design” by Dhruba Borthakur, http://hadoop.apache.org/docs/r0.18.0/hdfs\_design.pdf .

[11]Hadoop - Advantages and Disadvantages http://www.j2eebrain.com/java-J2ee-hadoopadvantages-and-disadvantages.html. [12]Shivaraman Janakiraman ‘’Fault Tolerance in Hadoop for Work Migration” http://salsahpc.indiana.edu/b534projects/sites/default/files/public/8\_Fault%20Tolerance%20in%20Hadoop%20for%20Work%20Migration\_Janakiraman, % 20Shivaraman.pdf.

[13]Wang, F. et al. 2009. Hadoop High Availability through Metadata Replication. ACM.

[14]T. White. Hadoop: The Definitive Guide. O'Reilly, 2009.

[15] T. Cowsalya and S.R. Mugunthan: Hadoop architecture and fault tolerance based Hadoop clusters in geographically distributed data center, 2015 ISSN 1819-6608