Volume 6, Issue 12, December 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Efficiency of Hadoop Distributed File System using CloudSim

C. Gazala Akthar*, K. Shouryadhar, S. Nuzhath Pasha
Asst. Professor, Dept. of CSE, Ravindra College of Engineering for Women, Kurnool, Andhra Pradesh, India

Abstract*— HDFS is an open source which uses the distributed user-level file system store reliably and to stream those analysis data sets. It also promises users to have efficient HDFS with Quality of Service. Hence in order to perform the HDFS, it is necessary to know the user preferences and provide best file system under which the HDFS should be performed. This paper proposes a HDFS to select as best file system with the objective of maximizing the user satisfaction and investigates the performance of HDFS by using cloudSim.

Keywords*— HDFS, Name node, Data node, CloudSim.

I. INTRODUCTION

1.1 HDFS

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. Hadoop is an Apache top-level project being built and used by a global community of contributors and users. [1] .The Hadoop distributed file system is one of the most popular file system for fetching the data or files from the file system efficiently. HDFS is master/slave architecture, where master is **Name node** and slave is **data node**. Data node stores the data and Name node is in charge of file system operations. If Name node failure occurs then Hadoop provides duplicating file i.e. Secondary Name node. Data node replicates the data into multiple copies of data around the cluster.

Figure 1: The Workflow diagram of HDFS

1.2 NameNode*:*

The NameNode is responsible for storing the files metadata and for tracing all the operations realized on the file system. All these data are stored on the local file system of the NameNode in two files called "EditLog" and "FSImage"

- ***EditLog****:* Stores all changes that occur into the file system metadata's. For instance, adding a new file in HDFS is traced in this file.
- ***FSImage****:* Stores all the file system namespace: the location of files (or blocks) in the cluster

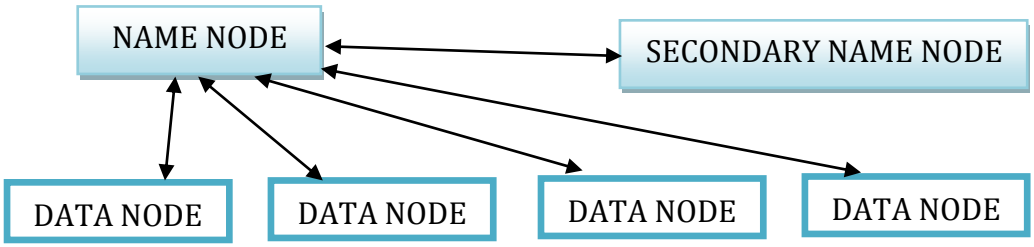

Figure 2: The NameNode diagram of HDFS

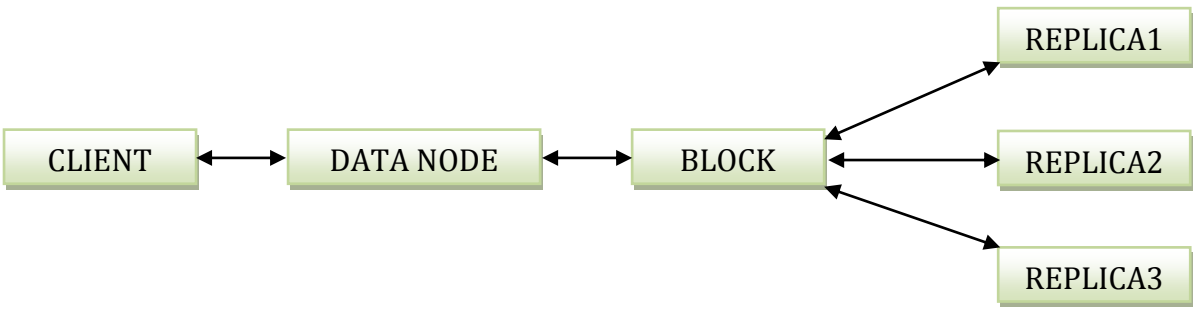

Figure 3: The DataNode diagram of HDFS

1.3 DataNodes:

The DataNodes simply store the data i.e. the files blocks without any knowledge of HDFS. The files blocks are stored on the local file system of the DataNodes and are replicated among the other DataNodes based on the configured replication factor.[2]

II. RELATED WORK

Analyzing Hadoop is an emerging hot field in academic research. Our work covers both Data node analysis and Name node analysis. A distributed file system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster. All modules in Hadoop are designed with a fundamental hardware failures are common and then they are automatically handled by software framework. HDFS are derived from Google file system and Gfs Map reduce. Hadoop is hosted in cloud and deployed in traditional on site data center as well as in cloud. So that it does not acquire any hardware[3]. Data in Hadoop cluster broken into smaller pieces (block), they are distributed through cluster. Map reduce functions can be executed on smaller subsets of larger data sets. It provides scalability that is needed for big data processing. Unlike other file systems it works with any distributed file system. Hadoop is rack aware file system. Hdfs is an open source which provides many interfaces and libraries for different file systems. Gfs and Hdfs has its own working style, the difference is in Hdfs uses Name node and Data node and Gfs uses chunk servers and chunks. An advantage of using HDFS is data awareness between the Job tracker and Task tracker.[4]

III. PROPOSED SYSTEM

We have proposed a HDFS, a File framework where a document is been overhauled in the Hadoop Distributed file system, it naturally make obstructs, copy's of the document in distinctive information hubs which mainly includes the name node and data node. In the event that customer needs to perform a few operations on documents, for example, altering, changing, erasing is possible by this record framework. We additionally utilize guide diminish strategies within this record framework. How HDFS is simulated over cloudSim, which is not done before. For efficient access of HDFS we carry out simulation by using the CloudSim and Hadoop.[5][7]

3.1 Comparisons of HDFS with Local File System Performance

The average writing/reading performance measured with the program for the small (512 MB) and big file (4 GB) on the testing cluster is

Table 1: writing and Reading performance of Local file system

LOCAL FILE SYSTEM	512MB	4GB
Write	34.145	43.122
Read	63.054	47.384

Compare with the HDFS performance

Table 2: Writing and reading performance of HDFS

HDFS	512MB	4GB
Write	34.145	32.205
Read	63.054	47.384

3.2 Problems That Are Solved By HDFS

Table3: Performance of HDFS

Problem	Solution
1. Data is so enormous it is not possible store in computer.	1. Data is put away in multi machine.
2. Top of the line machines are expensive.	2. Run on thing Hardware.
3. Product fittings will fail.	3.Programming is astute enough to manage fittings disappointment.
4. Equipment disappointment lead to loss.	4. Repeat information.
5. In what capacity will the disseminated hubs co-ordinate among them.	5. There is expert.

3.3 CloudSim

CloudSim is actually a software framework which supports several core functionalities

- Queuing & processing of events,
- Creation of CloudSim entities,
- Communication between components and management of simulation clock

Therefore this tool allows to[6]

1. Test application services in repeatable and controllable environment.
2. Tune the system bottlenecks before deploying apps in actual cloud.
3. Experiment with different workload mix and resource performance scenarios on simulated infrastructure for developing and testing adaptive application provisioning techniques..

IV. STUDY AND ANALYSIS

In this paper, we intend to associate the information hub and name hub with the HDFS document framework in action and authorize the execution along expected yield. At whatever point a customer overhauls a document in the Hdfs framework first Data hub gets initiated and takes the record from the customer it makes squares and imitations in it. At that point it sends a criticism to Name hub that is ace that it have gotten a document, the squares and reproductions are made. Name hub accordingly send to Data hub the piece got message furthermore adds that document to Hdfs as it specifically corresponds with Hdfs. The optional name hub has the al rights that a name hub has however it is just utilized when primary name disappointments happens. The Big data is simulated with the cloudsim where all the data or files that are to be stored, from HDFS they directly stores in cloudsim (data center). The following diagram represents the design and workflow of the project.[8]

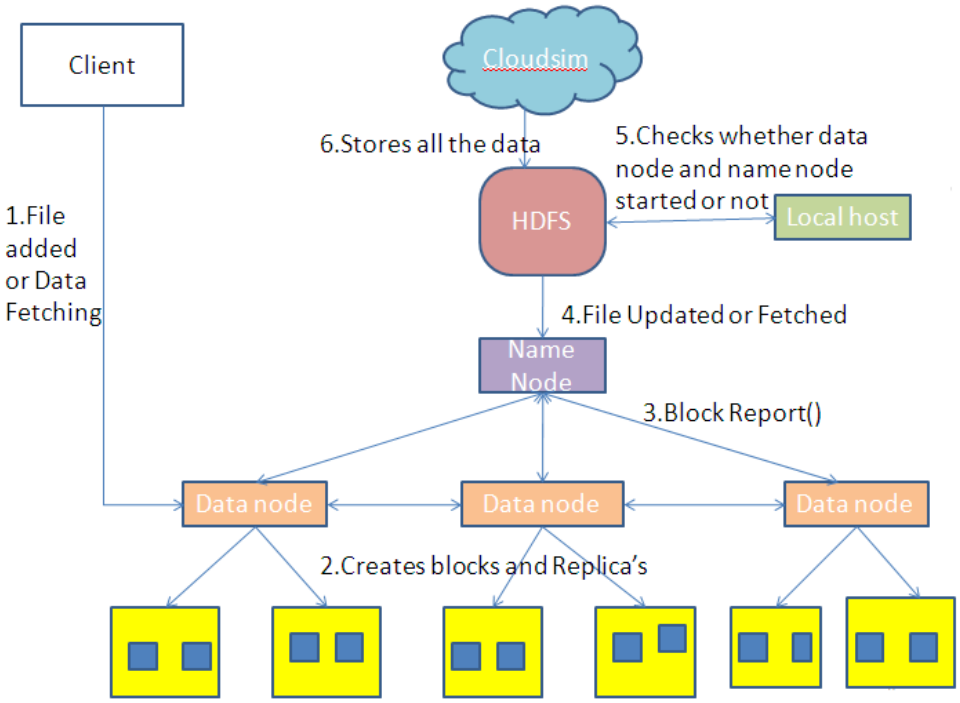

Figure 4: System design

4.2 Operations in HDFS

Performance of Write Opearaton is Graphed in HDFS

When certain numbers of files are added, automatically number of nodes gets increased. In HDFS writing performance scales better for small data sets that is been shown in the following graph clearly, where x-axis shows number of files and y-axis shows number of nodes.

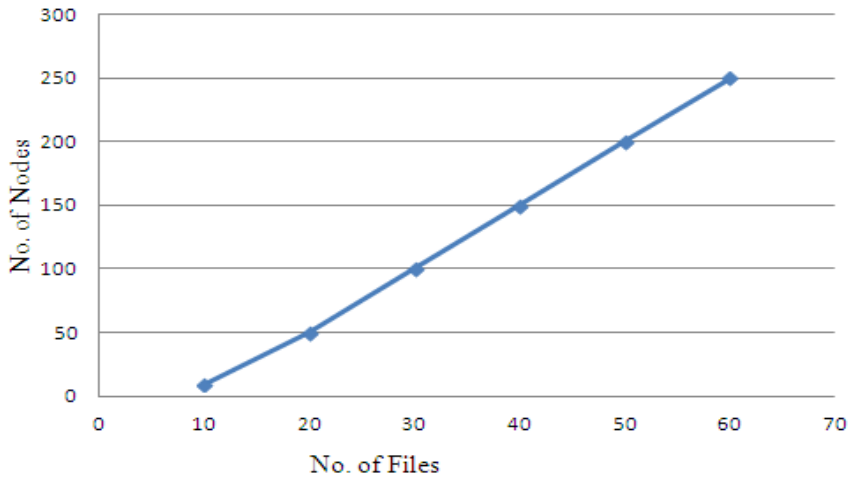

Figure 5: Performance of write operation in HDFS

When nodes are changed and files are constant, then it shows as below

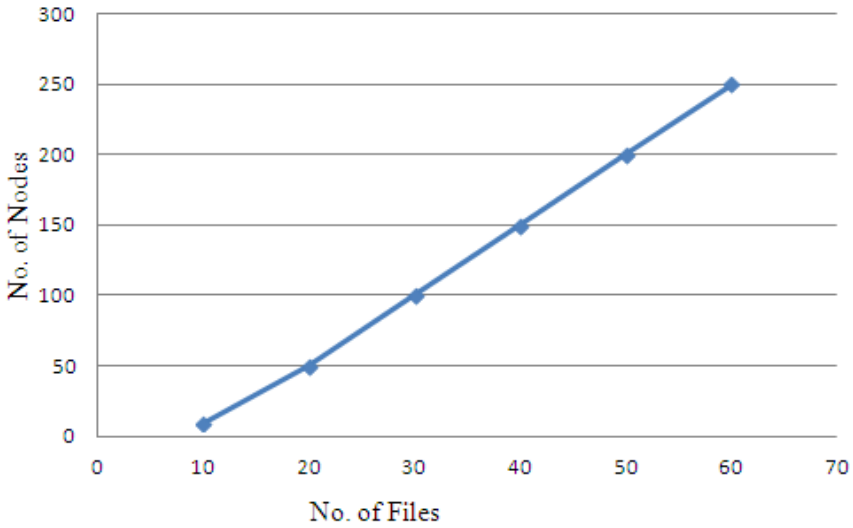

Figure 6: Performance of write operation in HDFS, when nodes are changed

When again nodes are changed

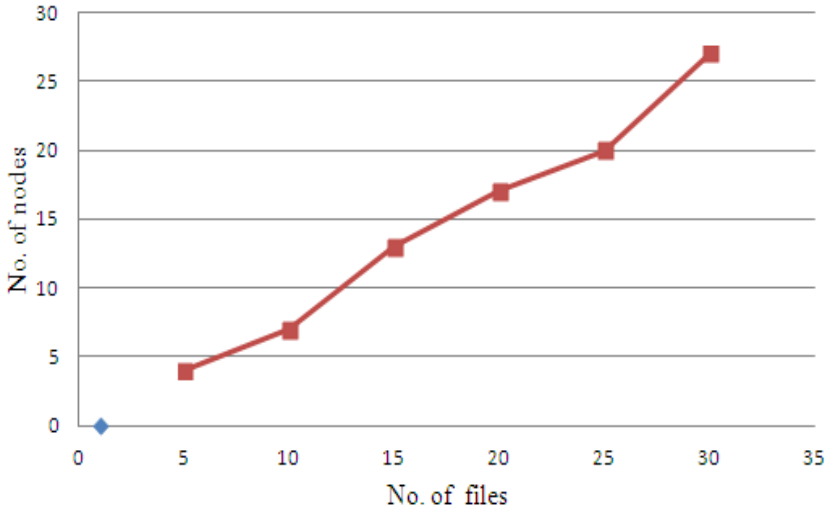

Figure 7: Performance of write operation in HDFS, when nodes are changed

When nodes are constant and files are changed

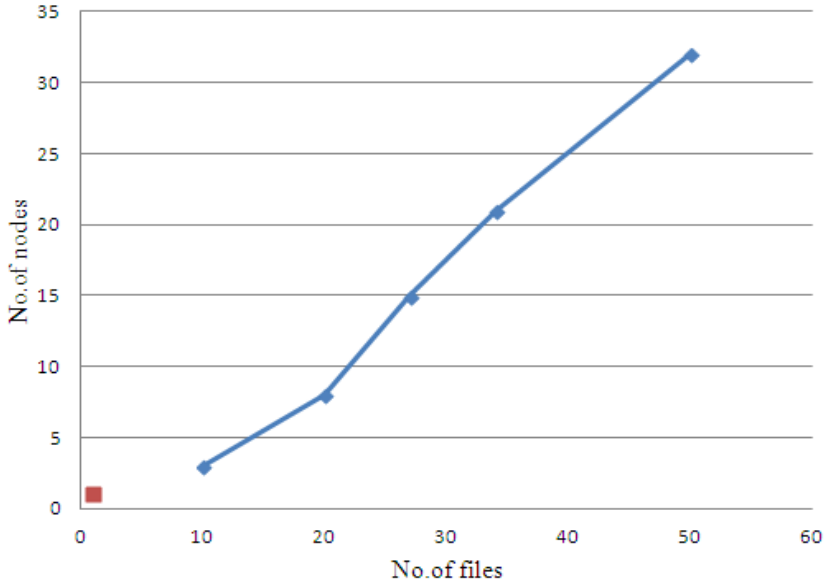

Figure 8: Performance of write operation in HDFS, when nodes are constant and files are changed

Other graph when files are changed

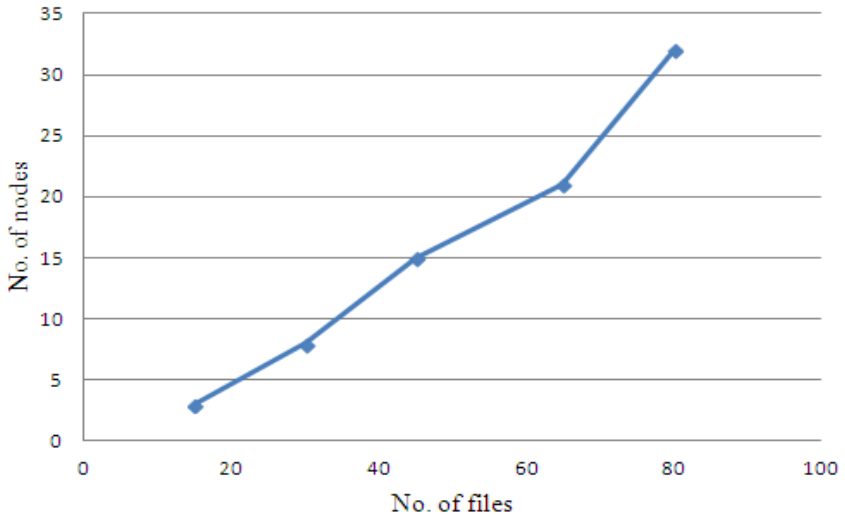

Figure 9: Performance of write operation in HDFS, when nodes are constant and files are changed

Both when files and nodes are changed, then the graph represents as follows

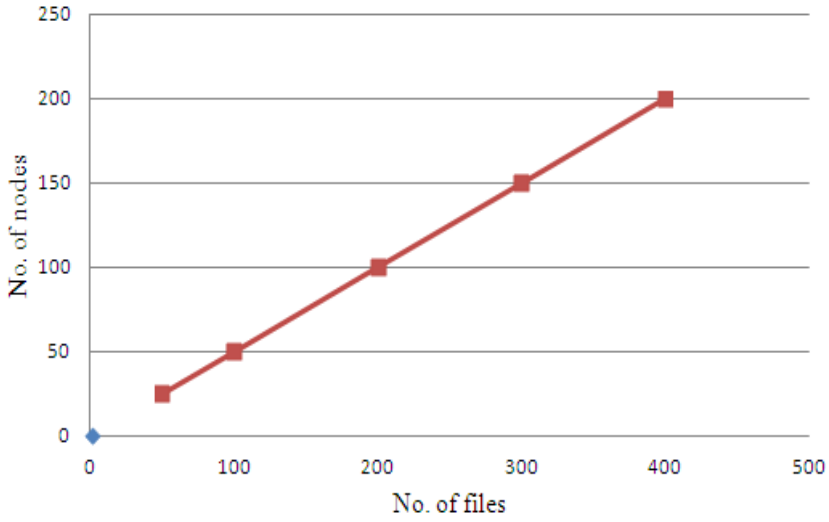

Figure 10: Performance of write operation in HDFS, when nodes are and files are changed

Other graph of files and nodes and changed

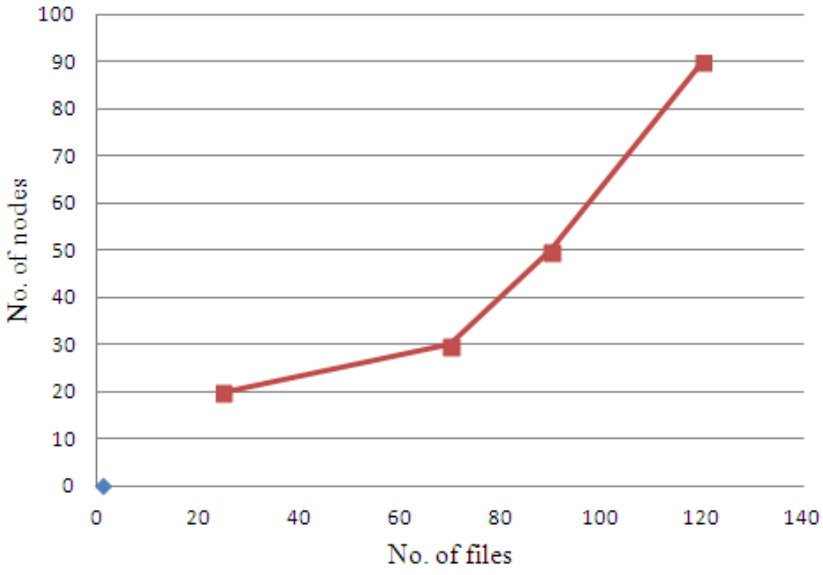

Figure 11: Performance of write operation in HDFS, when nodes are and files are changed

Performance of Read Opearaton is Graphed in HDFS

When certain files are to be read in HDFS then, if client wants to read less number of files as the nodes will be more then, performance decreases. If number of reading files are increased then nodes will be equally read in the same time as, when lesser number files are been read. This can be clearly understood by following graph.

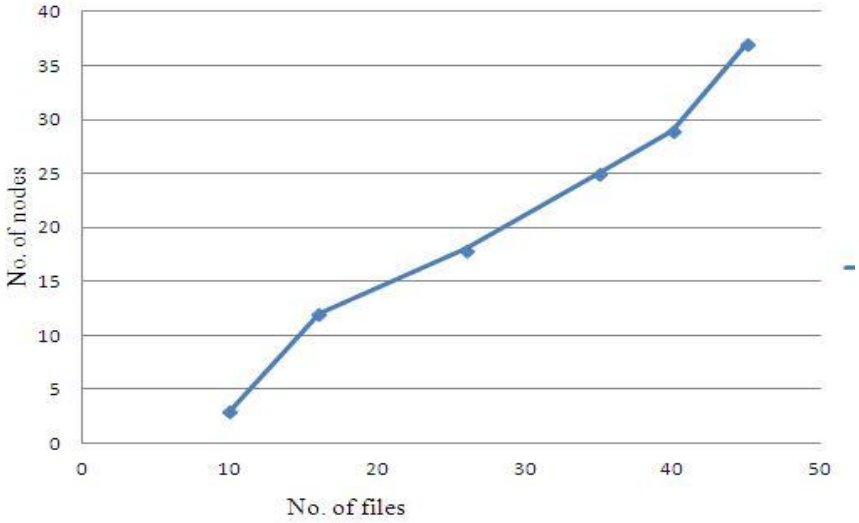

Figure 12: Performance of read operation in HDFS

When nodes are changed and files are constant, then graph represents as follows

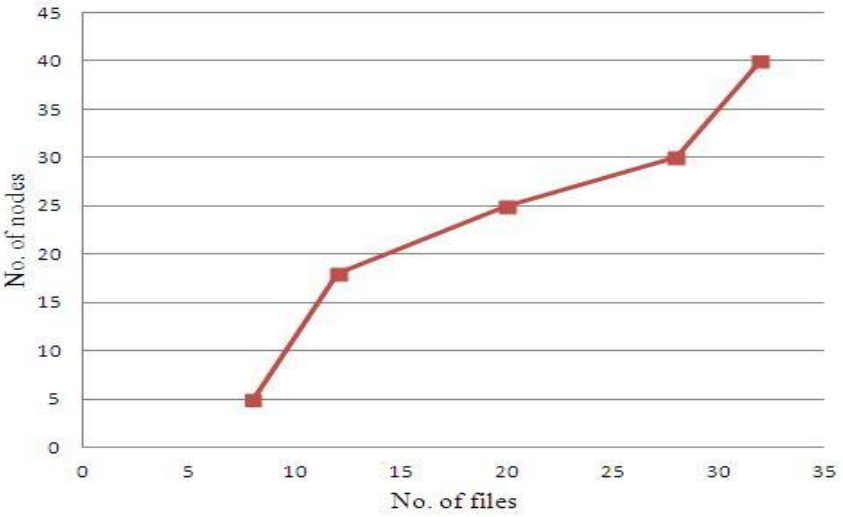

Figure 13: Performance of read operation in HDFS, when nodes are changed and files are constant

Other graph when nodes are changed, represents as follows

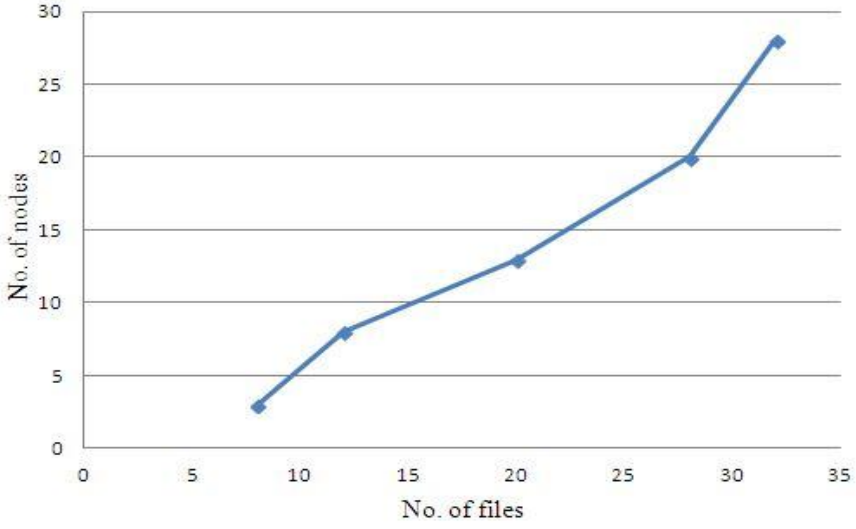

Figure 14: Performance of read operation in HDFS, when nodes are changed again and files are constant

When nodes are constant and files are changed, then graph represents as follows

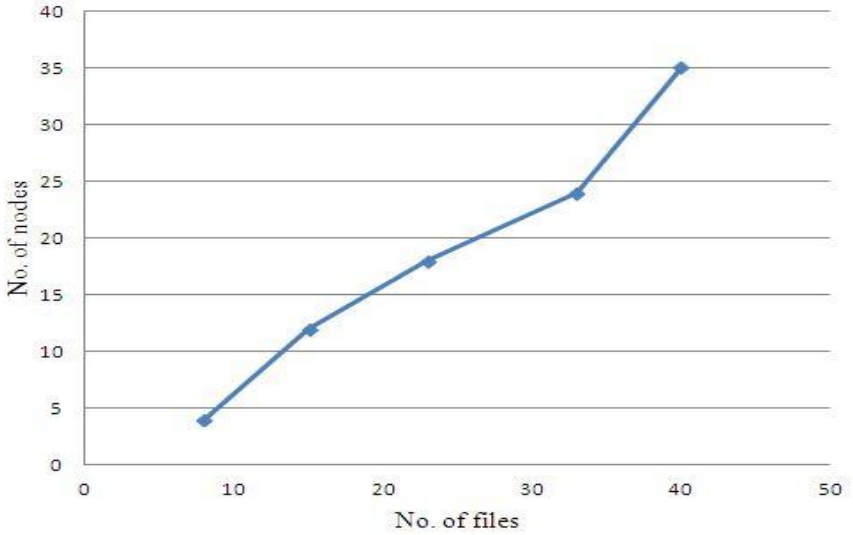

Figure 15: Performance of read operation in HDFS, when nodes are constant and files are changed

Other graph when files are changed, represents as follows

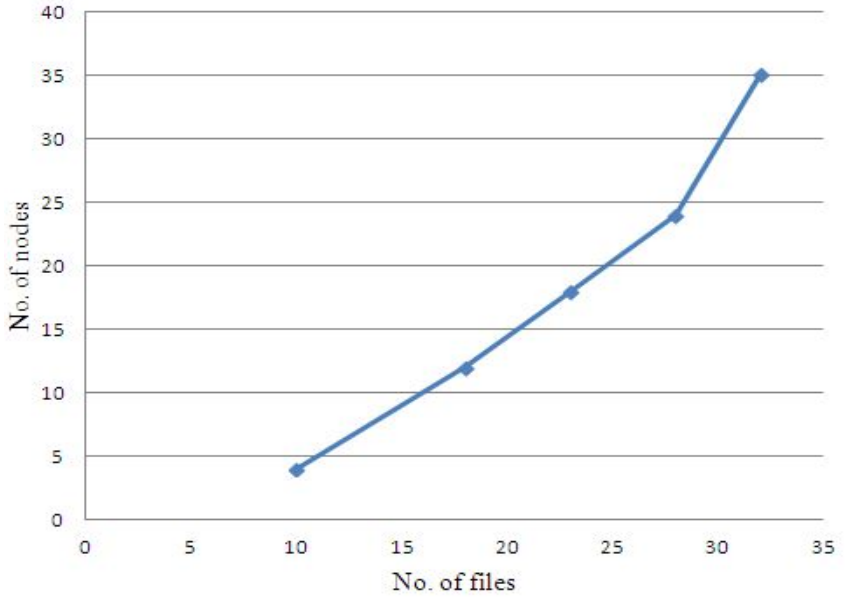

Figure 16: Performance of read operation in HDFS, when nodes are constant and files are changed again.

When both nodes and files are changed, then graph represents as follows

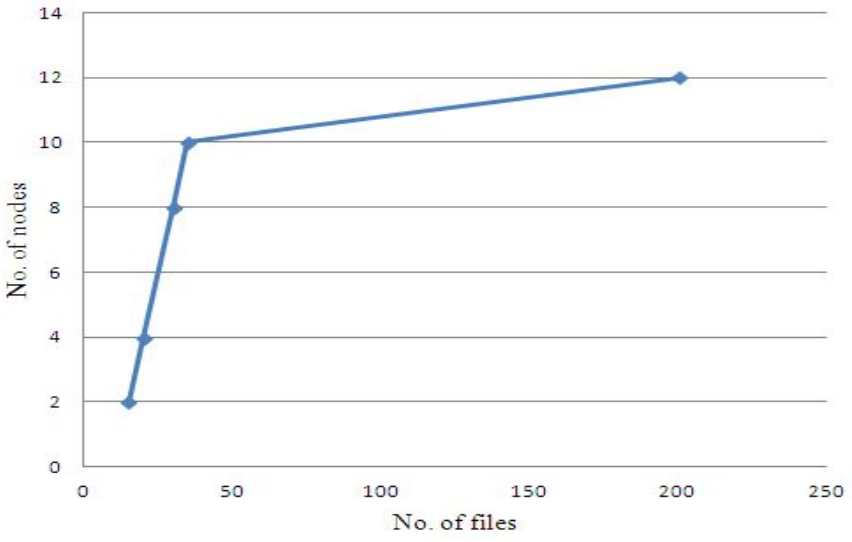

Figure 17: Performance of read operation in HDFS, when nodes and files are changed.

When nodes and files are again changed

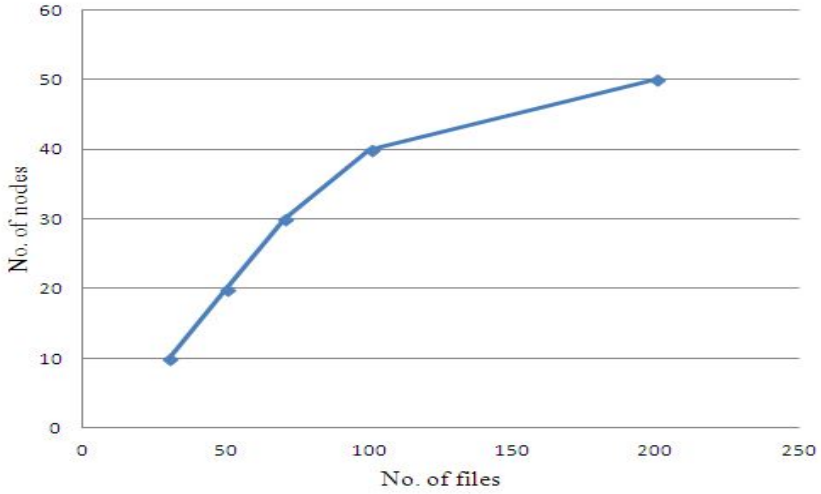

Figure 18: Performance of read operation in HDFS, when nodes and files are changed again.

By analyzing the above graphs, when read operation is performed in smaller data sets i.e less number of files, the performance is decreased .where as when number of files are to be increases then performance increased.

Performance of Access / Retrieve Opearation is Graphed in HDFS

When the client wants to access particular file from the HDFS then performance increases and decreases depending on the file that is been fetched or retrieved from the block that is stored depending on that the performance and decreases . The following graph represents it clearly.[9]

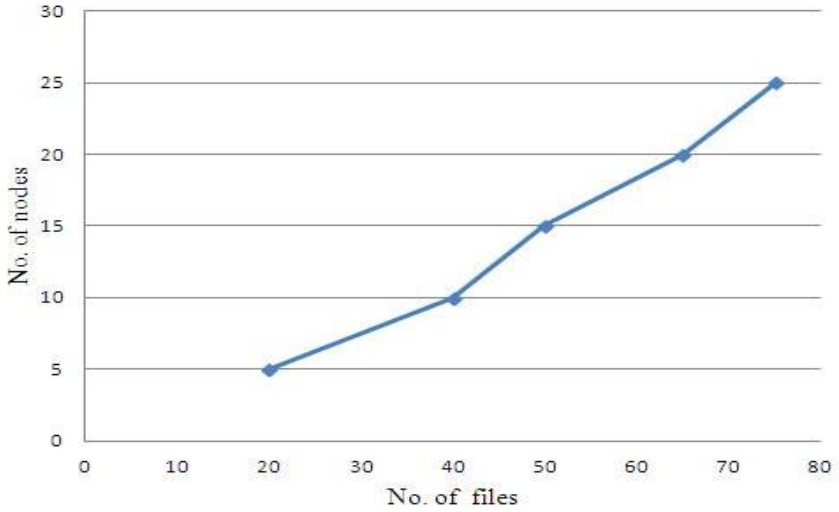

Figure 19: Performance of Access/ retrieve file operation in HDFS

When more files are present in HDFS then, how graph represents is shown below

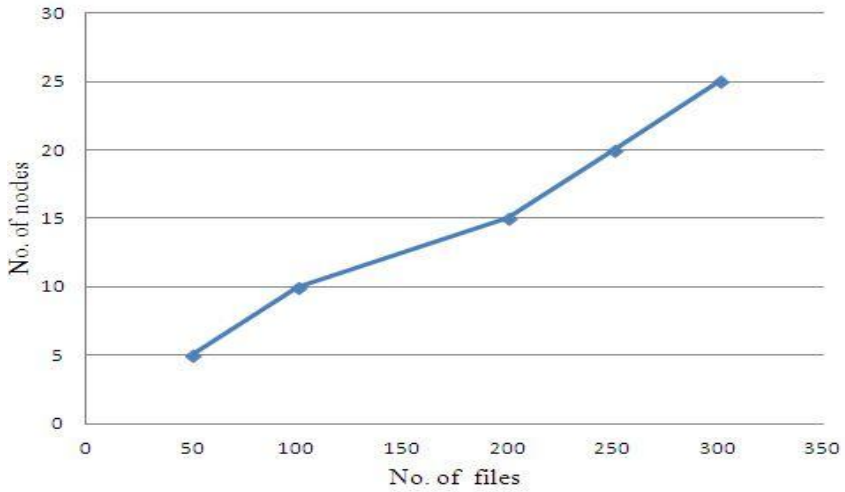

Figure 20: Performance of Access/ retrieve file operation in HDFS, when more files are present

The above graph shows, we conclude that as HDFS is parallel runner on different machines the file is accessed faster than the other file systems. While accessing some files may take lesser time to access because , as replicas may present in different blocks. So, in the lesser time the file is accessed.

Analyzing Result of Opearations

Hadoop is designed for clients, which don't run on a Hadoop daemon itself. If a client performs a writing operation, normally this operation will run on the NameNode and it will split the input data set and spread the split parts across the DataNodes. Otherwise if we want to perform the writing operation for some reason on any DataNodes internally, this operation will only performed locally to avoid congestion on the network. The writing performance of Hadoop scales better than reading with small data sets. But it doesn't matter because Hadoop is designed for the batch processing on huge data sets. So in this case it's quite fine with the scalability. Furthermore the writing and reading performance are fast.[10][11][12]

V. CONCLUSIONS

In this paper, we have proposed a simple Hadoop file which uses the HDFS as it is secured because we use that in Linux operating system and implement the file system using java. If client adds a particular file to the Hadoop file system then automatically data node and name node gets activated then blocks and replicas are created in the file system. In addition, future work includes responsible file system where the performance of a file system is increased using same commodity inexpensive hardware**.**

REFERENCES

[1] Sachin P Bappalige analyzied Hadoop opensource.com/life/14/8/intro-apache-hadoop-big-data.
[2] IBM, developer works.
[3] Applications and organizations using Hadoop". Wiki.apache.org. 2013-06-19. Retrieved 2013-10-17.
[4] Sanjay ghemawat, Harward gobioff and sheen tak bung In *Proceedings of the sosp 2003.*
[5] HDFS Architecture Guide:hadoop.apache.org/docs/stable1/hdfs_design.html
[6] For latest and up to date information, visit hadoop.apache.org
[7] The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne
[8] "HDFS Design". Hadoop.apache.org. Retrieved 2014-09-04.
[9] "Hdfs Working".hadoopilluminated.com/hadoop_illuminated/HDFS_Intro.html. Retrieved 2014-09-03.
[10] Data node wiki.apache.org/hadoop/DataNode . Retrieved on 2014-08-31
[11] Name node design bigdataplanet.info/2013/10/Hadoop-Tutorial-Part-2-HadoopDistributed-File-System
[12] Hadoop growth hadoop.apache.org/core/.

