Table 3: HDFS Cluster Test Cases

TID	*Description*	*Input*	*Expected Output*	*Actual Output*	*Pass /Fail*
1	Verify the starting of Name Node and Data Nodes	$bin/ start-all.sh	Master: starting name node,	Master: starting name node	Pass
			Slave1: starting Secondary name node	Slavel1: starting secondary name node	Pass
			Slave2: starting data node	Slave2: starting data node	Pass
			Slave2:starting data node	Slave2:starting data node	Pass
			Slave2: starting data node	Slave2: starting data node	Pass
2	Verify JPS comma-nd in Name Node	$jps	14399 Name-Node 12215 jps	14399 Name-Node 12215 jps	Pass
3	Verify JPS comma-nd in Secondary Name Node	$jps	11612 jps 16312 Secondary Name node	11612 jps 16312 secondary name node	pass
4	Verify JPS command in Data Node	$jps	11501 Data-Node 11612 jps	11501 Data-Node 11612 jps	Pass

6.1.2 Test Case 02

Test Case Name: Verifying Read Interface

Description:
The test case is designed to verify the read interface. It verifies whether the proper values or messages are displaying or not for the given source file or not. If the given file name and path are correct, the time taken to read will be displayed otherwise, it displays the error message.

Table 4: Read Test Cases

TID	*Description*	*Input*	*Expected Output*	*Actual output*	*Pass /Fail*
1	Verify the result for given wrong source file name in HDFS	Source File: HDFS:/ur/ 16 MB.zip	Message box should be displayed with error message	Message box displayed	Pass
2	Verify the result for given wrong file name is LFS	Source File:/HDF Scluster/d esktop/So urce/16 MB.zip	Message box should be displayed with error message	Message bodx display	pass
3	Verify the result for reading given source file name is HDFS	Source File:/HDF S:/User/16 MB.zip	Time in Milliseconds should be displayed	3812	Pass
4	Verify the result for given source file name is LFS	Source File:/ HDFSclus ter/deskto p/source/1 6MB.zip	Time in Milliseconds should be displayed	137824	Pass

6.1.3 Test Case 03

Test Case Name: Verifying the write interface.

Description:
This test case is designed to verify the write interface. It verifies whether the proper values or messages are displaying or not for the given source file and destination file or not. If the given file name and path are correct, the time taken to read will be displayed otherwise, it displays the error message.

Table 5: Write Test Case

TID	*Description*	*Input*	*Expected Output*	*Actual Output*	*Pass/ Fail*
1	Verify the result for given wrong source file name in HDFS	Source File:/hdfscluste r/desktop/sourc e/16MB.zip Destination file: HDFS:/ur/16M B.zip	Message box should be displayed with expectation	Message Box displayed	Pass
2	Verify the result for given wrong file name is LFS	Source File:/hdfscluste r/desktop/sourc e/16MB.zip Destination file: /HDFScluster/d esktop/destinati on/16MB.zip	Message box should be displayed with expectation	Message Box displayed	Pass
11	Verify the result for given source file name is HDFS	Source File:/hdfscluste r/desktop/sourc e/16MB.zip Destination file: HDFS:/ur/16M B.zip	Time in Milliseconds should be displayed	9737	Pass
3	Verify the result for given source file name in LFS	Source File:/hdfscluste r/desktop/sourc e/16MB.zip Destination file: /HDFScluster/d esktop/destinati on/16MB.zip	Time in Milliseconds should be displayed	51315	Pass

6.2 Conclusion on Testing

In this various test cases are designed for testing HDFS cluster, read interface and write interface. The HDFS is tested whether it is starting properly or not by using bin/start-all.sh and jps command. The read interface is tested by giving wrong file name in source and correct file names for both HDFS and LFS. The write interface is tested by giving wrong file names for source and destination for HDFS and LFS. My project is passed in all these tests.

7. Conclusion & Future Enhancement

7.1 Conclusion

Apache provides a distributed system hadoop Distributed File System for maintaining large data. HDFS gives an efficient environment for storing and handling large files in distributed manner compare to Local File System. LFS is able to handle small files efficiently. But for large files it takes lot of time for writing and reading. HDFS handles large files when compared to LFS. HDFS and LFS give the performance similar for writing small files and there is a notable difference in writing large files. The LFS gives better performance for small files compare to HDFS. Because the small file is available in disk and can be read well. HDFS gives less performance for reading small files because of extra load for java environment. HDFS gives better performance for reading large files because the data is divided into blocks which can be read in parallel fashion where LFS reads the data in Sequential fashion. All the read and write values obtain through interface are included and analyzed by generating graphs.

7.2 Future Enhancement

Even though Hadoop Distributed File System is fault tolerant, scalable and suitable for data intensive applications in this project I have evaluated the read and write performance of HDFS. HDFS follows **single-writer multiple-reader model** that means while write process is in progress, if any client wants to read the same file is not possible in the existing HDFS hence the future work is mostly to propose a model which supports concurrent file access.

References

[1] Konstantin Shvachko, Hairong kuand, Sanjay Radia, Robert Chansler, "The Hadoop Distributed File System" In MSST '10 Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies(MSST), IEEE Computer Society Washingtin, DC, USA 2010, pages 1-10

[2] S. Ghemawat, H. Gobioff, S. Leung. "The Google File System",In Proc.of ACM Symposium on Operating Systems Principles,Lake George ,NY,Oct 2003,pp 29-43

[3] P.H.Carms, W.B Ligon III, R.B.Ross, and R. Thakur, "PVFS: A Parallel File System for Linux Clusters" Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, GA, October 2000, pp. 317-327

[4] Russel Sandber, "The Sun Network File System: Design, Implementation and Experience", Sun Mircrosystems, Inc. 2550 Garcia Sve. Mountain View,CA. 94049(415) 960-7293.

[5] J.J Kistler and M. Satyanarayanan, Disconnected Operation in the Coda File System, Proceedings of the Thirteenth ACM Symposium on Operating Systems Principles, October 13-16, 1991, pages 213-225

[6] John H Howard, An overview of the Andrew File System, Information Technology Center, Carnegie Mellon University

[7] Tom White, "Hadoop: The Definitive Guide", Third Edition, O'reily/Yahoo! Press

[8] Eric Sammer, Hadoop Operations, September 2012: First Edition., O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol,CA 95472.

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

[10] P.H.Carms, W.B Ligon III, R.B.Ross, and R. Thakur, "PVFS: A Parallel File System for Linux Clusters," in Proc. Of 4th Annual Linux Showcase and Conference, 2000, pp 317-327

[11] J.Dean, S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," In Pric. Of the 6th Se Pig Experience," Symposium on Operating System Design and Implementation, San Francisco CA, Dec. 2004.

[12] A. Gates, O. Natkovich, S. Chopra, P. Kamath, S. Narayanam, C. Olston, B.Reed, S. Srinivasan, U. Srivastava. "Building a High-Level Dataflow System on top of MapReduce: The Pig Experience," In Pric. Of Very Large Data Bases, viol 2 no.2,2008 pp 1414-1425.

[13] S.Ghemawat, H.Gobioff,S.Leung, "The Google file System," In Proc. Of ACM Symposium on Operating Systems Principles, Lake George, NY, Oct 2003, pp 29-43

[14] F.P. Junqueira, B.C. Reed. "The life and times of a zookeeper" In Proc of the 28th ACM Symposium on Principles of Distributed Computing, Calgary, AB, Canada, August 10-12,2009.

[15] Lustre File System. http://www.lustre.org

[16] M.K. McKusick, S.Quinlan. "GFS: Evolution on Fast-Forward," ACM Queue, vol. 7, no.7, New York, NY. August 2009

[17] O. O'Malley, A.C.Murthy. Hadoop Sorts a Petabyte in 16.25 Hours and a Terabyte in 62 seconds. May 2009. http://developer.yahoo.net/blogs/hadoop/2009/05/hadoop_sorts_a_petabyte_in16_2.html

[18] R. Pike, D. Presotto, k.Thompson, H.Tricky, P.Winterbottom, "Use of Name Spaces in Plan9," Operating Systems Review,27(2), April 1993,pages 72-76

[19] S. Radia, "Naming Policies in the spring system," In Proc. Of 1st IEEE workshop on Services in Distributed and Networked Environments, June 1994, pp 164-171.

[20] S. Radia, J. Pachl, "The Peer-Process View of Naming and Remote Execution," IEEE parallel and Distributed Technology, vol.1, no.3, August 1993, pp 71-80.

[21] K.v. Shvachko,"HDFS Scalability: The limits to growth,"; login:. April 2010, pp6-16.

[22] W. Tantisiriroj, S. Patil, G. Gibson. "Data-intensive file systems for Internet services: A rose by any other name…." Technical Report CMUPDL-08-114, Parallel Data Laboratory, Varnegie Mellon University, Pittsburgh, A, October 2008.

[23] A.T Husoo, J.S. Sarma, N. Jain, Z.Shao, P.Chakka,S.Anthony, H.Liu,P.Wyckofff, R.Murthy, "Hive-A Warehousing solution Over a MapReduce Framework," In Proc. Of Very Large Data Bases, vol.2 no.2, August 2009, pp 1626-1629.

Author Profile

Linthala Srinithya received Bachelor of Engineering in Computer Science and Engineering from TRRCE, JNTUH. She is pursuing Master of Technology in Computer Science. Her research interests are Big Data and Analytics, Operating Systems, Data mining, Networking, Web Technologies, Image Processing, Computer Graphics.

G. Venkata Rami Reddy has completed his Master of Technology in Computer Science from School Of IT, JNTU, Kukatpally Hyderabad.. He is the Associate Professor and course coordinator of Software Engineering for School of IT, JNTUH. His subjects of interests are Image Processing, Computer Networks, Analysis of Algorithms, Data mining, Operating Systems and Web technologies.