VOL. 10, NO. 18, OCTOBER 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

DATA TRANSMISSION PERFORMANCE ANALYSIS IN CLOUD AND GRID

Mohammed Abdulkarem[1] and Rohaya Latip[1, 2]
[1]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, (UPM), Serdang, Selangor, Malaysia
[2]Institute for Mathematical Research, Universiti Putra Malaysia, (UPM), Serdang, Selangor, Malaysia
E-Mail: memo.ye@gmail.com

ABSTRACT

Hadoop Distributed File System (HDFS) and MapReduce programming model are for storage and retrieval of the big data. The Terabytes size file can be easily stored on the HDFS and can be analyzed with MapReduce. HDFS is becoming more popular in recent years as a key building block of integrated grid storage solution in the field of scientific computing. However, due to the nature of HDFS that it cannot support asynchronous write, it is widely confirmed that for the case of sustained high throughput in WAN transfer, single stream per GridFTP transfer is the best solution. GridFTP, designed by using Globus, is one of the most popular protocol for performing data transfers in the Grid environment. In this paper, we take on the challenge of integrating Hadoop with grid, by proposing a new framework called Grid-over-Hadoop by retaining the features of Hadoop and using GridFTP for data transfer.

Keywords: hadoop, HDFS, GridFTP, MapReduce, Globus.

INTRODUCTION

Hadoop (Yu *et al.* 2014) is an open source usage of MapReduce introduced by the Apache Foundation, and upheld by the most important IT organizations, like, Facebook and Yahoo!. Besides, it is a recently accurate Java-based application that has a distributed file system as well as firmly incorporated applications, which are able to rapidly store a huge size of data. Hadoop carries out MapReduce framework with two classifications including a JobTracker and numerous Task-Trackers. The JobTracker orders TaskTrackers to process synchronically data via two principle tasks: map as well as reduce.

Hadoop allows distributed (Kala Karun *et al.* 2013) data intensive and parallel applications by disintegrating a large task into small tasks as well as huge data set into small segments where every tasks are processed into segments synchronically. Hadoop utilizes Hadoop distributed File System (HDFS) that is an open source origin of Google File System (GFS) to store data. The application of Map/Reduce fundamentally utilizes HDFS to store data. Moreover, HDFS is a vast distributed file system, which utilizes commercial hardware and gives high throughput as well as fault tolerance. Numerous organizations (Kulkarni *et al.* 2014) think that after few years half of the world's data is going to be stored in Hadoop. However, HDFS performs files storage in the form of series blocks as well as the block replicated for the sake of fault tolerance. Dividing key data, handling, formats, copying as well as blocking data will enhance Hadoop efficiency as a great deal of investigation is going ahead around there.

Grid (Radic, 2007) used to tackle issues that require more resources than a typical system can give. Grid involves distinctive systems and can be used for various relevant functions. Grids are made of various computer nodes and are utilized for calculating which need many computer resources.

GridFTP (Kettimuthu *et al.* 2012) is the real standard protocol for exchanging huge data files within the platform of Grid or HPC production. GridFTP is an extension of the standard FTP protocol to secure high-performance and trustworthy data transfer protocol upgraded for high-bandwidth WAN. The Globus GridFTP usage has turned into the overwhelming data transfer protocol for the Grid environment, with extensive facilities and organizations utilizing it to transform billion files annually.

GridFTP (Kettimuthu *et al.* 2012) is an effective arrangement that has been created over the previous decade to deal with such big data transport prerequisites. GridFTP amplifies that the standard File Transfer Protocol (FTP) through valuable characteristics, for example, expanded dependability through restart markers, Grid Security Infrastructure (GSI) security (Vivas *et al.* 2010), high-performance data transfer by utilizing parallel streams, striping and enhancing third-party exchange between GridFTP servers. GridFTP has indicated to perform numerous tasks of great higher throughput than performing other ways of data transfer like, secure copy (SCP).

Integrating Hadoop with GridFTP is by all accounts a superior choice for transferring data, on the grounds, it provides GridFTP data transfer with Hadoop features. This integration is because of HDFS ability that it cannot support asynchronous write. It is affirmed that, for the instance of supported high throughput in single stream GridFTP exchange in WAN, it is the best solution.

In this paper, three experiments have been carried out for analyzing the performance of WAN data transfer in terms of time on Hadoop, GridFTP, and the proposed new framework that integrates Hadoop with GridFTP called "Grid-over-Hadoop."

ARCHITECTURE

In this section, we give some brief background material and architecture on the topics covered in the paper.

Hadoop architecture

A Hadoop cluster (He *et al.* 2012) consists of two main components: Hadoop Distributed File System and MapReduce. A Hadoop cluster utilizes Hadoop Distributed File System (HDFS) (Shvachko *et al.* 2010) in order to deal with its data. HDFS supplies storage for MapReduce tasks of input and output data. It is devised as an exceptionally fault tolerant, high throughput, and high capacity distributed file system. Furthermore, it enables saving cluster data of terabytes or petabytes as well as it has variable hardware requirements, which commonly contain commercial hardware such as PCs. The main dissimilarities between HDFS and other distributed file systems are: HDFS's write-once, read-many and stream access models. It enables HDFS to distribute and process data effectively, to store safely huge quantities of data, and powerfully joining heterogeneous hardware and environments of operating system. However, it separates every file into small fixed-size blocks (ex, 64 MB) and stores different duplicates for every block on cluster node disks. Distributing data blocks increases fault tolerance as well as throughput. HDFS belongs to the master/slave architecture.

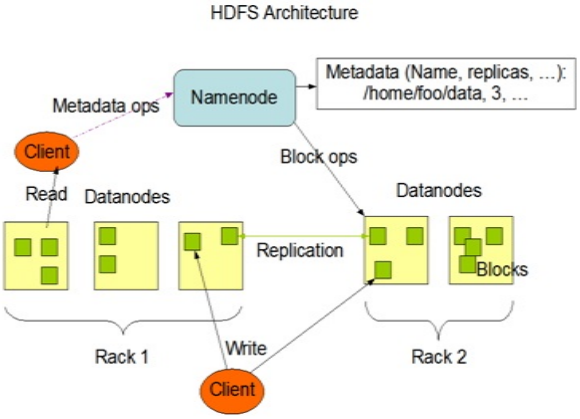

Figure-1. HDFS architecture.

The master node is known as the NameNode (He *et al.* 2012), which deals with the file system namespace and manages customer's access to data. There are various operating nods, named DataNodes, which store real data in the form of block units. The NameNode keeps up a mapping table, which maps data, blocks to DataNodes to perform write and read orders from the clients of HDFS. Moreover, it is accountable for file system namespace operations, for example, opening files, renaming, closing, and directories. HDFS permits a secondary NameNode to have a copy of the metadata saved on the NameNode when there is a NameNode error. The DataNodes stores the blocks of data in its local disk and performs commands such as substituting data, copying, creating, and deleting from the NameNode. Figure-1 (He *et al.* 2012) demonstrates the architecture of HDFS.

A DataNode occasionally reports its status through a pulse message and requests the NameNode for directions. Listening to the network by Each DataNode, consequently It lets other DataNode and clients to be able to perform the processes of read and write. In addition, the pulse may assist the NameNode to identify communications with its DataNode. The DataNode is indicated, as down by NameNode, in case the NameNode does not get a pulse from a DataNode in the configured period of time. Data blocks stored on this node are accounted as lost and the NameNode will spontaneously copy the blocks of the lost node.

Globus GridFTP architecture

GridFTP (Subramoni *et al.* 2010) is a reliable, secured, high-performance extension of the standard FTP data transfer protocol, upgraded for WAN data transfer with high-bandwidth. GridFTP, which implemented by Globus (Allcock *et al.* 2005) gives suitable software, streamlined to a wide scope for applications of data access, such as massive file exchanging and extracting data from complicated storing systems. The server of Globus GridFTP (Kettimuthu *et al.* 2012) is conveyed on more than 5,000 servers overall and is in charge of more than 10 million transfers summing almost half a petabyte of data daily. A huge number of establishments around the world depending on GridFTP (Allen *et al.* 2012), GridFTP has been utilized as a part of record setting DOE/HEP Large Hadron Collider Computing Grid data challenges (Subramoni *et al.* 2010). Moreover, it is a principal part for an extensive variety of distributed data management and computing systems. The following are some of the basic features of GridFTP:

- **Performance:** Enhanced performance by implementing of coordinated data transfers and parallel streams.
- **Robustness:** Restart markers enabling interrupted transfers to restart with minimal delay overhead.
- **Extensibility:** Easy-to-use OCRW interface to applications and users.
- **Security:** PKI/X.509 and SSH-based Grid security.

Globus GridFTP (Kettimuthu *et al.* 2012) includes three consistently unique components: client, server protocol interpreters (PIs), which deals with the control channel protocol, and the data transfer process (DTP) that deals with access to the actual data and its movement through the data channel protocol. However, the first two jobs are different on the grounds that a protocol exchange is asymmetric. Figure 2 illustrates the architecture of GridFTP.

These components can integrate in different ways to establish servers with different capabilities. For instance, connecting the components of server PI and DTP in one process makes a traditional FTP server, even

though a striped server could implement one PI server on the main node of a cluster and a DTP located on all other nodes.

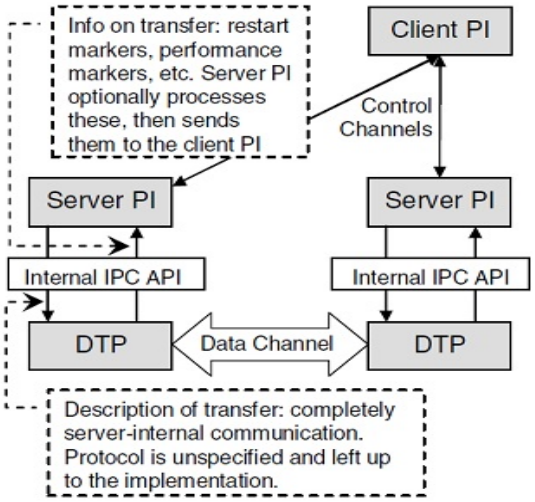

Figure-2. Globus GridFTP architecture.

Globus has Grid Security Infrastructure (GSI), which gives encryption and authentication to the data transfer, with user indicated levels of data integrity and privacy. The design of Globus GridFTP accommodates secure validation of control channel demands (required) and of data channel integrity and privacy (not obligatory). A session made when the client launches a TCP connection with the port on which the server is tuning in. Primarily, validation should be carried out as in RFC 2228 (Horowitz *et al.* 1997). However, the client shows an assigned proxy certificate (Tuecke *et al.* 2004), and the server must offer a certificate issued by a CA that is trusted by the client.

METHODOLOGY

In the integration of our framework Grid-over-Hadoop, the GridFTP should use GridFTP client to access the HDFS file system using Fuse. Most of the efforts focus on optimizing the total number of GridFTP that can transfer the data and the number of streams for each GridFTP for WAN data transfer at the same time. This is because HDFS's is not able to provide asynchronous write, it is largely affirmed that for the case of sustained high performance of data transfer, a single stream per GridFTP transfer is the most approved recommendation (Amin *et al.* 2011). Based on our hypothesis that the time of data transfer in GridFTP is faster than in Hadoop, we introduce a solution by proposing a framework called Grid-over-Hadoop. In this paper, three experiments have been carried out for examining the time of data transfer of GridFTP, Hadoop and the new proposed framework.

In order to implement the experiments, we consider the following software as required to be used with Hadoop to construct and test Hadoop, Globus GridFTP and Grid-over-Hadoop framework:

- **Globus GridFTP server**

Globus GridFTP (Liu, 2013) uses the SSL protocol along with X.509 credentials to provide transfer between Globus server and clients.

- **Fuse**

It is a Linux kernel module (Narayan *et al.* 2010) that permits file systems to write in userspace, and provides POSIX-like interface to HDFS. With Fuse mount, the entire file system of the HDFS will be local to the Globus GridFTP server, which provides data transfer from, or to the client.

- **Eclipse**

Eclipse with Hadoop-eclipse-plugin are established to supply a capability for the clients to transfer data from or to Hadoop HDFS file system.

Experiments

In this section, we describe our three experiments Hadoop, Globus GridFTP, and Grid-over-Hadoop by discussing the experimental setup, provide the results of our experiments, and give an in-depth analysis of the results. Besides, two parameters are considered in all experiments; the first parameter is data used to be transferred concerning file size in Gigabyte. The size range starts from one GB to eight GB for each file created as a text file; the second parameter is time of WAN data transfer from or to the client, which is measured using Wireshark and log files on each server.

Experiment 1: Globus GridFTP

The experimental setup includes two hosts connected through the D-link WAN routers. The hosts fit out with the Intel series of processors with i7 processor operating at 3.0 GHz with 4 GB RAM, 10/100MB network adapter, and windows 7 operating system. Oracle Virtual box 4.1.18 r78361 is used for Linux Ubuntu release 10.04 operating system with 2GB RAM. Globus 5.0.5 is used for GridFTP and Grid Security infrastructure (GSI). The installation and configuration of Globus GridFTP server and GSI are a multi-step process as the following:

- Installation (Foster, 2011) has four steps: (1) download Globus; (2) unzip the Globus tar file; (3) run “terminal”; and (4) execute “make” and “make install” in sequence. These steps are not complex but can take a lot of time, and need the target machine to have development and building tools to be installed.
- Security configuration includes four more steps. It has taken a long time to execute the tasks: (1) getting an X.509 host certificate from a well-known certificate authority such as Globus simpleCA; (2) install the X.509 host certificate; (3) configure the reliable certificates path with the certificates of simpleCAs that we want to trust; and (4) setup authorization, that is, to produce mappings between users’ Grid identities

(distinguished name in their certificate) to a local user account.

- Extra work is included to configure and assign certificates for each user, in case this has not been performed. This work requires getting an X.509 user certificate from certificate authority (Globus simpleCA), installing that certificate, configuring the trusted certificates directory with the certificates of simpleCAs that you want to trust, and sending the distinguished name in the new certificate to your server admin, so that it can be mapped to the local user account on the server.

Table-1 below contains the result of transferring the file using Globus GridFTP protocol from server to client or vice versa, concerning file size and transferring time in a second.

Table-1. Globus experiment: file size in Gigabyte with transmission time in second.

Size	1GB	2GB	3GB	4GB	5GB	6GB	7GB	8GB
Test1	94	189	292	386	478	574	679	794
Test2	95	192	297	378	518	596	672	748
Test3	96	189	294	398	519	573	676	758
Test4	96	190	291	386	520	572	679	786
Test5	98	189	293	380	520	579	676	746
Avg.	95.8	189.8	293.4	385.6	511	578.8	676.4	766.4

Experiment 2: Hadoop Multi-Node Cluster

The experimental setup has three hosts connected through the D-link WAN routers. The hosts are equipped with the Intel series of processors with i7 processor operating at 3.0 GHz with 4 GB RAM, 10/100MB network adapter, and windows 7 operating system. Oracle Virtual box 4.1.18 r78361 is utilized for Linux Ubuntu release 10.04 operating system with 2GB RAM for each one. Hadoop 1.0.3 is installed and configured as multi-node cluster, which includes NameNodes as master and two DataNodes as slaves. Hadoop Client should install eclipse with Hadoop 1.0.3 plugin to communicate master node and browse HDFS file system. Figure-3 demonstrates the structure of Hadoop experiment.

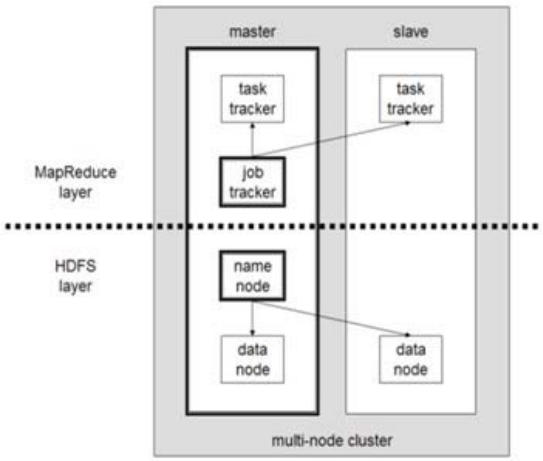

Figure-3. Hadoop multi-node cluster structure.

Hadoop architecture is characterized in (Attebury *et al.* 2009) and a summary of Hadoop experiment structure, main parts, and its file system HDFS, which was constructed in our experiments, is demonstrated in the following:

- HDFS has a master/slave architecture. An HDFS system consists a single NameNode and two DataNodes. HDFS exposes a file system namespace and permits user data to be stored in files. Locally, a file is divided into one or more blocks and such blocks are saved in a set of DataNodes.
- NameNode is a master server that manages the file system namespace and organizes access to files by clients. The NameNode implements file system namespace processes such as opening, renaming and closing files and directories. It identifies as well the mapping of blocks to DataNodes.
- DataNodes: Two DataNodes are used to perform storing attached to the nodes that they run on and to be in charge of presenting read and write orders from the file systems clients. The DataNodes as well carry out block creation, replication, and deletion upon tasks from the NameNode. The first DataNode configured to be in master server and the second one in the slave.

Table-2 below contains the result of transfer file using Hadoop multi-node cluster from server to client or vice versa using eclipse in terms of file size and time of transmission in a second:

Table-2. Hadoop experiment: file size in Gigabyte with transmission time in second.

Size	1GB	2GB	3GB	4GB	5GB	6GB	7GB	8GB
Test1	141	267	405	525	652	793	834	930
Test2	138	261	402	533	655	772	859	925
Test3	135	263	404	526	650	777	813	933
Test4	136	265	402	529	650	799	813	928
Test5	136	262	398	529	652	772	806	941
Avg.	**137.2**	**263.6**	**402.2**	**528.4**	**651.8**	**782.6**	**825**	**931.4**

Experiment 3: Grid-over-Hadoop framework

The architecture of Grid-over-Hadoop is comprised of three components. The first is the Globus GridFTP server and client for data transfer. The second major component is the Hadoop distributed file system (HDFS) that provides data storage to be mounted. The third component is Fuse to implement a HDFS file-system in a userspace program.

The experimental setup is the same as specification of Experiment 1 and 2. Besides, both Hadoop and Globus GridFTP located on the same server "master" and using Fuse to provide interface of HDFS for local data access from the GridFTP. Figure-4 illustrates the architecture of Grid-over-Hadoop framework.

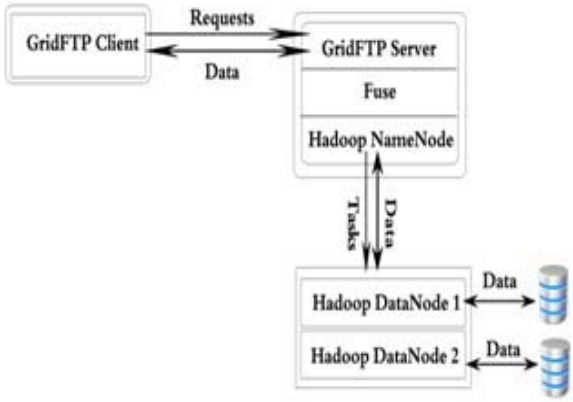

Figure-4. Framework Grid-Over-Hadoop structure.

Fuse (Narayan *et al.* 2010) is basically improved to enhance AVFS. In our experiment, we use Fuse to mount HDFS file-system to local storage and can be accessed by Globus GridFTP for data transfer. Figure-5 illustrates the directory of a file-system call as an instance.

The Fuse library and the Fuse kernel module (Narayan *et al.* 2010) communicate through a special file descriptor, which is maintained by opening /dev/fuse. This file can be opened many times, and obtained the file descriptor to move into the mount syscall, to match up the descriptor with the mounted file-system.

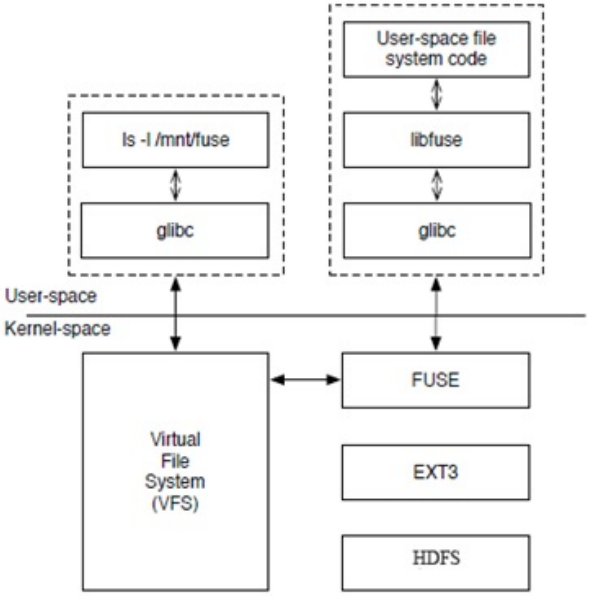

Figure-5. Fuse implementation method.

Result of transfer file using Grid-over-Hadoop from server to client or vice versa in terms of file size and time of transmission in a second is represented in Table-3. In additional, the overall result of the three experiments is illustrated in Figure-6.

Table-3. Grid-over-Hadoop experiment: file size in Gigabyte with transmission time in second.

Size	1GB	2GB	3GB	4GB	5GB	6GB	7GB	8GB
Test1	99	190	322	429	499	589	682	795
Test2	95	197	311	417	588	602	688	790
Test3	96	216	309	425	512	585	707	791
Test4	95	208	326	419	528	605	682	833
Test5	97	197	303	401	538	646	726	765
Avg.	**96.4**	**201.6**	**314.2**	**418.2**	**533**	**605.4**	**697**	**794.8**

Based on the result of the Experiments 1 and 2, HDFS did not perform as well as GridFTP even though HDFS forms a cluster. Regardless of the file size, GridFTP was around 28.56% better than Hadoop. Hadoop is located on top of the server's file system HDFS, therefore in order to access HDFS, a request needs to go through the server's storage first, and then get the file from HDFS. This extra process could be the reason why it takes longer to read the file. On the other hand, GridFTP client merely reads the file from hard disk. The overhead is too big for a HDFS client to access HDFS by Java RPC. It leads us to believe that the HDFS access layer is the bottleneck for faster data transfer of Hadoop-based cloud storage.

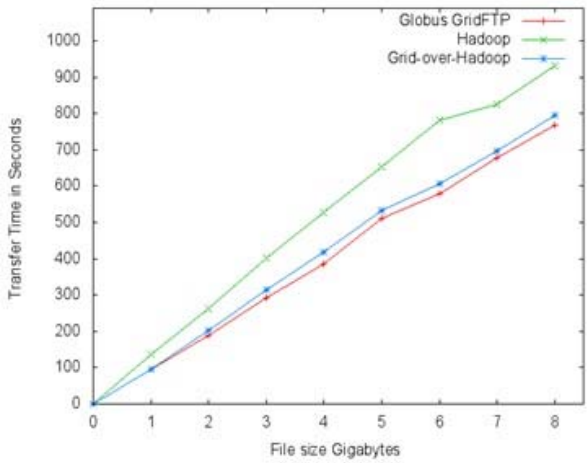

Figure-6. Result of the three experiments: file size in Gigabyte with transmission time in second.

Grid-over-Hadoop result was approximately 24%, better than Hadoop result, but it was not as fast as using GridFTP only. GridFTP was around 4.67% better than Grid-over-Hadoop that is due to Fuse, which acts as gateway to HDFS, therefore it creates overhead to read and write the file. However, the speed of GridFTP neutralizes the Fuse overhead.

This compromised solution Grid-over-Hadoop provided us a relative fast transfer rate, and in the meantime maintain all of the great features of Hadoop. Moreover, X.509 authentication will be mandatory if we use GridFTP, which solves the security weakness on HDFS.

CONCLUSIONS

In this paper, we presented a Grid-over-Hadoop framework and analyzed the time of data transfer among Hadoop, Globus GridFTP and the new proposed framework. Three experiments took place for testing and analyzing. Through the evaluation, we found that Hadoop did not perform as well as GridFTP, also Grid-over-Hadoop produces better results than Hadoop. Grid-over-Hadoop solution provided us a relative better data transfer time, and in the meantime maintained all the great features of Hadoop.

For future work, we plan to integrate Kerberos to Grid-over-Hadoop framework to solve the security weakness on Hadoop, therefore users first authenticate with Hadoop, and get the file from HDFS server to the client using GridFTP.

REFERENCES

[1] Allcock W., Bresnahan J., Kettimuthu R., Link M., Dumitrescu C., Raicu I. and Foster I. 2005. The Globus striped GridFTP framework and server. In: Proceedings of the 2005 ACM/IEEE conference on Supercomputing (p. 54). IEEE Computer Society.

[2] Allen B., Bresnahan J., Childers L., Foster I., Kandaswamy G., Kettimuthu R. and Tuecke S. 2012. Software as a service for data scientists. Communications of the ACM, Vol. 55, No. 2, pp.81-88.

[3] Amin A., Bockelman B., Letts J., Levshina T., Martin T., Pi H. and Wüerthwein F. 2011. High Throughput WAN Data Transfer with Hadoop-based Storage. In: Journal of Physics: Conference Series, December Vol. 331, No. 5, p. 052016. IOP Publishing

[4] Attebury G., Baranovski A., Bloom K., Bockelman B., Kcira D., Letts J. and Wuerthwein F. 2009. Hadoop distributed file system for the Grid. In: Nuclear Science Symposium Conference Record (NSS/MIC), pp. 1056-1061.

[5] Foster I. 2011. Globus Online: Accelerating and Democratizing Science through Cloud-Based Services. Internet Computing, IEEE, 15(1089-7801), pp.70-73.

[6] He C., Weitzel D., Swanson D. and Lu Y. 2012. Hog: Distributed hadoop mapreduce on the grid. In High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion, pp. 1276-1283.

[7] Horowitz M. and Lunt S. 1997. FTP security extensions (No. RFC 2228). RFC 2228.

[8] Kala Karun A. and Chitharanjan K. 2013. A review on hadoop—HDFS infrastructure extensions. In: Information & Communication Technologies (ICT), 2013 IEEE Conference on (pp. 132-137).

[9] Kettimuthu R., Lacinski L., Link M., Pickett K., Tuecke S. and Foster I. 2012. Instant gridftp. In: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, pp. 1104-1112.

[10] Kulkarni A. P. and Khandewal M. 2014. Survey on Hadoop and Introduction to YARN. International Journal of Emerging Technology and Advanced Engineering, Vol. 4, No. 05, pp. 82-87.

[11] Liu W.-L. 2013. Cloud Storage Performance and Security Analysis with Hadoop and GridFTP. s.l., sjsu.edu.

[12] Narayan S., Mehta R. K. and Chandy J. A. 2010. User space storage system stack modules with file level control. In: Proceedings of the 12th Annual Linux Symposium in Ottawa, pp. 189-196, July.

[13] Yu W., Wang Y. and Que X. 2014. Design and evaluation of network-levitated merge for hadoop

acceleration. IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 3, pp. 602-611.

[14] Shvachko K., Kuang H., Radia S. and Chansler R. 2010. The hadoop distributed file system. In: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on (pp. 1-10).

[15] Subramoni H., Lai P., Kettimuthu R. and Panda D. K. 2010. High performance data transfer in grid environment using gridftp over infiniband. In: Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on pp. 557-564, May.

[16] Tuecke S., Welch V., Engert D., Pearlman L. and Thompson M. 2004. Internet X. 509 public key infrastructure (PKI) proxy certificate profile (No. RFC 3820).

[17] Vivas J. L., Fernández-Gago C., Lopez J. and Benjumea A. 2010. A security framework for a workflow-based grid development platform. Computer Standards & Interfaces, Vol. 32, No. 5, pp. 230-245.