IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 1, Ver. IV (Jan – Feb. 2015), PP 41-48
www.iosrjournals.org

Comparative Analysis, Security Aspects & Optimization of Workload in Gfs Based Map Reduce Framework in Cloud System

Rahul U. Patil[1], Atul.U. Patil[2]
[1]*(M.TECH CSE, JNTU/ JNTU University, Hyderabad)*
[2]*(M.E CSE, ADCET)/ Shivaji University, Kolahpur)*

Abstract: *This paper discusses a propose cloud infrastructure that combines On-Demand allocation of resources with improved utilization, opportunistic provisioning of cycles from idle cloud nodes to other processes It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.Because for cloud computing to avail all the demanded services to the cloud consumers is very difficult. It is a major issue to meet cloud consumer's requirements. Hence On-Demand cloud infrastructure using map reduce configuration with improved CPU utilization and storage utilization is proposed using Google File System by using Map-Reduce. Hence all cloud nodes which remains idle are all in use and also improvement in security challenges and achieves load balancing and fast processing of large data in less amount of time. Here we compare the FTP and GFS for file uploading and file downloading; and enhance the CPU utilization and storage utilization and fault tolerance,. Cloud computing moves the application software and databases to the large data centres, where the management of the data and services may not be fully trustworthy. Therefore this security problem is solve by encrypting the data using encryption/decryption algorithm and Map-Reducing algorithm which solve the problem of utilization of all idle cloud nodes for larger data.*

Keywords: *CPU utilization, GFS Master, Chunk Servers, Map-Reduce, Google File System, Encryption/decryption algorithm.*

I. Introduction

Google has designed and implemented a scalable distributed file system for their large distributed data intensive applications. They named it Google File System, GFS. Google File System is designed by Sanjay Ghemawat, Howard Gobioff and Shun-Tak Leung of Google in 2002-03. GFS provides fault tolerance, while running on inexpensive commodity hardware and also serving large number of clients with high aggregate performance. Even though the GFS shares many similar goals with previous distributed file systems, the design has been driven by Google's unique workload and environment. Google had to rethink the file system to serve their "very large scale" applications, using inexpensive commodity hardware. [1].

Google give results faster and more accurate than other search engines. Definitely the accuracy is dependent on how the algorithm is designed. Their initial search technology is Page Rank Algorithm designed by Garry Brin and Larry Page in 1998. And currently they are merging the technology of using both software and hardware in smarter way. Now the field of Google is beyond the searching. It supports uploading video in their server, Google Video; it gives email account of few gigabytes to each user, Gmail; it has great map applications like Google Map and Google Earth; Google Product application, Google News application, and the count goes on. Like, search application, all these applications are heavily data intensive and Google provides the service very efficiently.

In the recent years, Infrastructure-as-a-Service (IaaS) cloud computing has emerged as an attractive alternative to the acquisition and management of physical resources. A key advantage of Infrastructure-as-a-Service (IaaS) clouds is providing users on-demand access to resources. However, to provide on-demand access, cloud providers must either significantly overprovision their infrastructure (and pay a high price for operating resources with low utilization) or reject a large proportion of user requests (in which case the access is no longer on-demand). At the same time, not all users require truly on-demand access to resources [3].

Many applications and workflows are designed for recoverable systems where interruptions in service are expected. Here a method is propose, a cloud infrastructure with GFS configuration that combines on-demand allocation of resources with opportunistic provisioning of cycles from idle cloud nodes to other processes. The objective is to handles larger data in less amount of time and keep utilization of all idle cloud nodes through

splitting of larger files into smaller one using GFS read/write algorithm, also increase the CPU utilization and storage utilization for uploading files and downloading files It provides fault tolerance while running on inexpensive commodity hardware . To keep data and services trustworthy, security is also maintain using RSA algorithm which is widely used for secure data transmission.

II. Related Work

There is much research work in the field of cloud computing and distributed computing over the past decades. Some of the work done has been discussed, this paper researched distributed file system and its safety, proposed a new cloud computing architecture, SaaS model was used to deployed the related software on the GFS map reduce platform, so that the resource utilization and computing of scientific tasks quality will be improved[19].

a cloud infrastructure that combines on-demand allocation of resources with opportunistic provisioning of cycles from idle cloud nodes to other processes by deploying backfill virtual machines (VMs)[21].A model for securing Map/Reduce computation in the cloud. The model uses a language based security approach to enforce information flow policies that vary dynamically due to a restricted revocable delegation of access rights between principals. The decentralized label model (DLM) is used to express these policies[20].

A new security architecture, Split Clouds, which protects the information stored in a cloud, while the architecture lets each organization hold direct security controls to their information, instead of leaving them to cloud providers. The core of the architecture consists of real-time lineage summaries, in-line security gateway and shadow auditor. By the combination of the three solutions, the architecture will prevent malicious activities performed even by the security administrators in the cloud providers [21].

III. Google File System Architecture

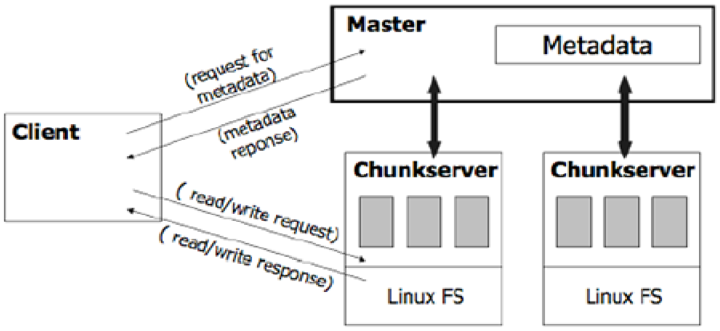


Fig. 1 System Architecture.

A GFS cluster consists of a single master and multiple chunkservers and is accessed by multiple clients. The basic analogy of GFS is master maintains the metadata; client contacts the master and retrieves the metadata about chunks that are stored in chunkservers; next time, client directly contacts the chunkservers. Figure 1 describes these steps more clearly. Each of these is typically a commodity Linux machine running a user-level server process. Files are divided into fixed-size chunks. Each chunk is identified by an immutable and globally unique 64 bit chunk handle assigned by the master at the time of chunk creation. Chunkservers store chunks on local disks as Linux files and read or write [5].

Chunk data specified by a chunk handle and byte range. For reliability, each chunk is replicated on multiple chunk servers. By default, three replicas are stored, though users can designate different replication levels for different regions of the file namespace. The master maintains all file system metadata. This includes the namespace, access control information, the mapping from files to chunks, and the current locations of chunks. It also controls system-wide activities such as chunk lease management, garbage collection of orphaned chunks, and chunk migration between chunk servers.

The master periodically communicates with each chunk server in Heartbeat messages to give it instructions and collect its state. GFS client code linked into each application implements the file system API and communicates with the master and chunk servers to read or write data on behalf of the application. Clients interact with the master for metadata operations, but all data-bearing communication goes directly to the chunkservers. Neither the client nor the chunkservers caches file data. Client caches offer little benefit because most applications stream through huge files or have working sets too large to be cached. Not having them simplifies the client and the overall system by eliminating cache coherence issues. (Clients do cache metadata, however.) Chunkservers need not cache file data because chunks are stored as local files and so Linux's buffer cache already keeps frequently accessed data in memory. Before going into basic distributed file system operations like read, write, we will discuss the concept of chunks, metadata, master, and will also describe how master and chunkservers communicates.

In the domain of data analysis, we propose GFS and Map Reduce paradigm and its open-source implementation in Hadoop , in terms of usability and performance :

1. GFS Multinode Configuration(GFS Master)
2. Client registration and Login facility(GFS client)

3. Cloud Service Provider
4. GFS Read/Write Algorithm using map reduce.
5. Encryption/Decryption of Data for security
6. Administration of client files(Third Party Auditor)

3.1 GFS Multinode Configuration (GFS Master): Master is a single process running on a separate machine that stores all metadata, e.g. file namespace, file to chunk mappings, chunk location information, access control information, chunk version numbers, etc. Clients contact master to get the metadata to contact the chunkservers. Master and chunkservers communicate regularly to obtain the state, if the chunkservers is down, if there is any disk corruption, if any replicas got corrupted, which chunk replicas store chunkservers, etc. Master also sends instruction to the chunkservers for deleting existing chunks, creating new chunks.The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures [6]. Hadoop was inspired by MapReduce, framework in which an application is broken down into numerous small parts. Any of these parts (also called fragments or blocks) can be run on any node in the cluster.

3.2 Chunk: Chunk in GFS is very important design decision. It is similar to the concept of block in file systems, but much larger than the typical block size. Compared to the few KBs of general block size of file systems, the size of chunk is 64 MB. This design was to help in the unique environment of Google. As explained in the introduction, in Google's world, nothing is small. They work with TBs of data and multiple-GB files are very common.

Their average file size is around 100MB, so 64MB works very well for them; in fact it was needed for them. It has few benefits, e.g. it doesn't need to contact master many times, it can gather lots of data in one contact, and hence it reduces client's need to contact with the master, which reduces loads from the master; it reduces size of metadata in master, (bigger the size of chunks, less number of chunks available. e.g. with 2 MB chunk size for 100 MB data, we have 50 chunks; again with 10 MB chunk size for same 100 MB, we have 10 chunks), so we have less chunks and less metadata for chunks in the master; on large chunks the client can perform many operations; and finally because of lazy space allocation, there are no internal fragmentation, which otherwise could be a big downside against the large chunk size.

3.3 Metadata: The master stores three major types of metadata: the file and chunk namespaces, the mapping from files to chunks, and the location of each chunk's replicas. Among these three, the first two types (namespaces and file-to-chunk mapping) are kept persistent by keeping the log of mutations to an operation log stored on the master's local disk. This operation log is also replicated on remote machines. In case the master crashes anytime, it can update the master state simply, reliably, and without risking inconsistency with the help of these operation logs. The master doesn't store chunk location information persistently, instead it asks each chunkservers about its chunks when it starts up or when a chunkserver joins the cluster.

Key GFS Features:

- Scale-Out Architecture - Add servers to increase capacity
- High Availability - Serve mission-critical workflows and applications
- Fault Tolerance - Automatically and seamlessly recover from failures
- Flexible Access – Multiple and open frameworks for serialization and file system mounts
- Load Balancing - Place data intelligently for maximum efficiency and utilization
- Chunk Replication- Multiple copies of each file provide data protection and computational performance
- Security - POSIX-based file permissions for users and groups with optional LDAP integration[8].

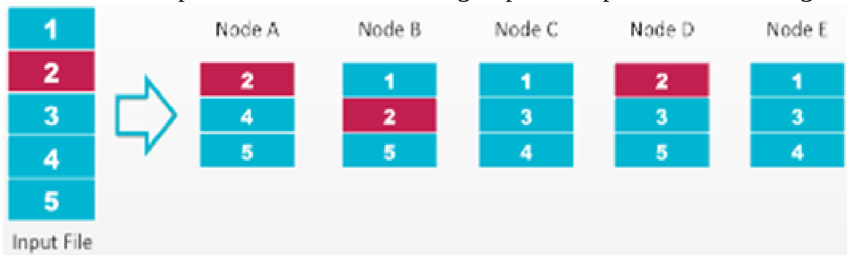


Fig.2 GFS data distribution[8]

Data in GFS is replicated across multiple nodes for compute performance and data protection.

3.2 Client registration and Login facility(GFS client)

It provide Interface to Login. Client can upload the file and download file from cloud and get the detailed summery of his account. In this way security is provided to the client by providing client user name and password and stores it in database at the main server which ensures the security. Any data uploaded and downloaded, log record has each activity which can be used for further audit trails. With this facility, it ensures enough security to client and data stored at the cloud servers only can be modified by the client.

3.3 Cloud Service Provider(Administrator)

It is administration of user and data.Cloud service provider has an authority to add and remove clients. It ensures enough security on client's data stored at the cloud servers. Also the log records of each registered and authorize client on cloud only can access the services. This specific client log record is helps in improve security.

3.4 GFS Read/Write Algorithm using map reduce

Map-Reduce is a programming model and an associated implementation for processing and generating large datasets that is amenable to a broad variety of real-world tasks. Users specify the computation in terms of a map and a reduce function also Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system[7]. MapReduce is a massively scalable, parallel processing framework that works in tandem with HDFS. With MapReduce and Hadoop, compute is executed at the location of the data, rather than moving data to the compute location; data storage and computation coexist on the same physical nodes in the cluster. MapReduce processes exceedingly large amounts of data without being affected by traditional bottlenecks like network bandwidth by taking advantage of this data proximity[8].

Our implementation of GFS Read/Write Map-Reduce Algorithm runs on a large cluster of commodity machines and is highly scalable. Map-Reduce is Popularized by open-source Hadoop project. Our GFS Read/Write Map-Reduce Algorithm works on processing of large files by dividing them on number of chunks and assigning the tasks to the cluster nodes in GFS multimode configuration. In these ways our proposed File Splitting Map-Reduce algorithm improves the Utilization of the Cluster nodes in terms of Time, CPU, and storage.

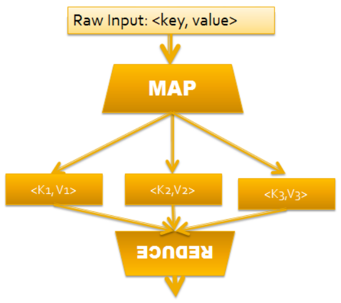


Fig.3 programming framework.

Applying a map operation to each logical 'record' in our input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key, in order to combine the derived data appropriately. Our use of a programming model with user specified map and reduce operationsallows us to parallelize large computations easily [7]. It enables parallelization and distribution of large scale computations, combined with an implementation of this interface that achieves high performance on large clusters of commodity PCs.

3.4.1 Programming Model

GFS Read/Write Map-Reduce Algorithm -

In this scenario clients is going to upload or download file from the main server where the file splitting map-reduce algorithm going to execute. On main server the mapper function will provide the list of available cluster I/P addresses to which tasks are get assigned so that the task of files splitting get assigned to each live clusters. File splitting map-reduce algorithm splits file according to size and the available cluster nodes.

The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of Map-Reduce library expresses the computation as two functions: Map and Reduce[7].

Map, Written by user, takes an input pair and produces a set of intermediate key/value pairs. The Map-Reduce library groups together all intermediate values associated with the same intermediate key and passes them to the Reduce function[7].

Read Algorithm-

We have explained the concept of chunk, metadata, master, and also briefly explained theCommunication process between client, master, and chunk servers in different sections. Now we will explain few basic operations of a distributed file systems, like Read, Write and also Record Append that is another basic operation for Google. In this section, we will see how the read operation works

Following is the algorithm for the Read operation, with Figures explaining the part of the algorithm.

1. Application originates the read request
2. GFS client translates the request form (filename, byte range) -> (filename, chunk index), and sends it to master
3. Master responds with chunk handle and replica locations (i.e. chunk servers where the replicas are stored).
4. Client picks a location and sends the (chunk handle, byte range) request to the location.
5. Chunk server sends requested data to the client
6. Client forwards the data to the application.

Write Algorithm-

1. Application originates the request
2. GFS client translates request from (filename, data) -> (filename, chunk index), and sends it to master
3. Master responds with chunk handle and (primary + secondary) replica locations.
4. Client pushes write data to all locations. Data is stored in chunkservers' internal buffers.
5. Client sends write command to primary
6. Primary determines serial order for data instances stored in its buffer and writes the instances in that order to the chunk
7. Primary sends the serial order to the secondary's and tells them to perform the write.
8. Secondary's respond to the primary
9. Primary responds back to the client

3.5Encryption/decryption for data security by using RSA Algorithm

In this, file get encrypted/decrypted by using the RSA encryption/decryption algorithm.RSA encryption/decryption algorithm uses public key & private key for the encryption and decryption of data.Client upload the file along with some secrete/public key so private key is generated &file get encrypted. At the reverse process by using the public key/private key pair file get decrypted and downloaded.Like client upload the file with the public key and the file name which is used to generate the unique private key is used for encrypting the file.

In this way uploaded file get encrypted and store at main servers and then this file get splitted by using the File splitting map reduce algorithm which provides unique security feature for cloud data. In a reverse process of downloading the data from cloud servers, file name and public key used to generate secrete and combines the all parts of file and then data get decrypted and downloaded which ensures the tremendous amount of security to cloud data.

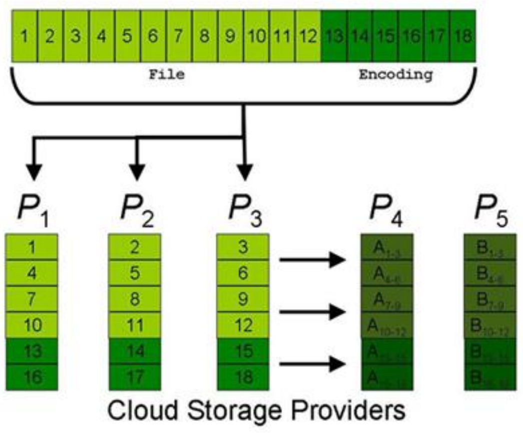
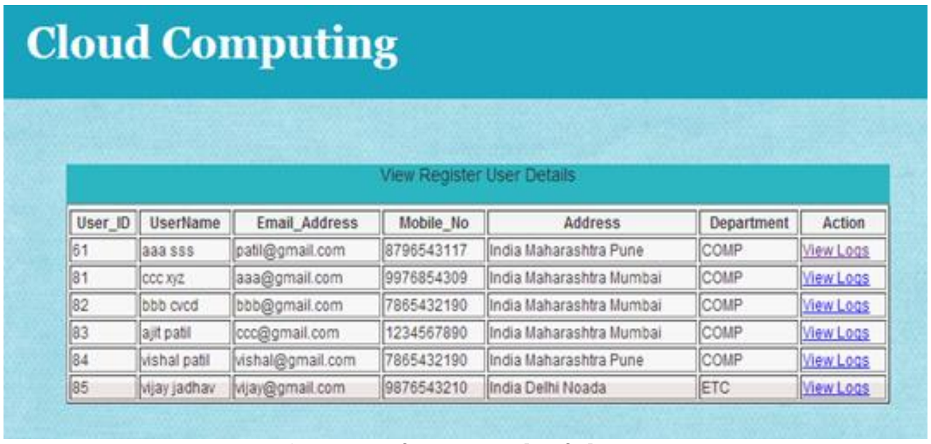


Fig.4 encryption/decryption.

3.6 Administration of client files(Third Party Auditor)

This module provides facility for auditing all client files, As Various activities are done by Client. Files Log records and got created and Stored on Main Server. For each registered client Log record is get created which records the various activities like which operations (upload/download) performed by client.Also Log records keep track of time and date at which various activities carried out by client. For the safety and security of the Client data and also for the auditing purposes the Log records helps.Also for the Administrator Log record facility is provided which records the Log information of all the registered clients. So that Administrator can control over the all the data stored on Cloud servers.Administrator can see Client wise Log records which helps us to detect the fraud data access if any fake user try to access the data stored on Cloud servers.

Registered Client Log records:

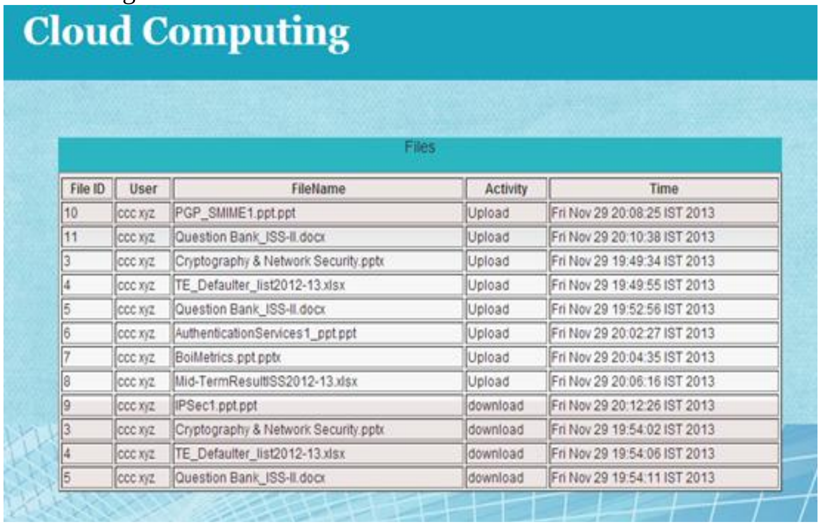


View Register User Details

User_ID	UserName	Email_Address	Mobile_No	Address	Department	Action
61	aaa sss	patil@gmail.com	8796543117	India Maharashtra Pune	COMP	View Logs
81	ccc xyz	aaa@gmail.com	9976854309	India Maharashtra Mumbai	COMP	View Logs
82	bbb cvcd	bbb@gmail.com	7865432190	India Maharashtra Mumbai	COMP	View Logs
83	ajit patil	ccc@gmail.com	1234567890	India Maharashtra Mumbai	COMP	View Logs
84	vishal patil	vishal@gmail.com	7865432190	India Maharashtra Pune	COMP	View Logs
85	vijay jadhav	vijay@gmail.com	9876543210	India Delhi Noada	ETC	View Logs

Fig.5 List of Log records of clients.

Registered Specific Client Log records:



Files

File ID	User	FileName	Activity	Time
10	ccc xyz	PGP_SMIME1.ppt.ppt	Upload	Fri Nov 29 20:08:25 IST 2013
11	ccc xyz	Question Bank_ISS-II.docx	Upload	Fri Nov 29 20:10:38 IST 2013
3	ccc xyz	Cryptography & Network Security.pptx	Upload	Fri Nov 29 19:49:34 IST 2013
4	ccc xyz	TE_Defaulter_list2012-13.xlsx	Upload	Fri Nov 29 19:49:55 IST 2013
5	ccc xyz	Question Bank_ISS-II.docx	Upload	Fri Nov 29 19:52:56 IST 2013
6	ccc xyz	AuthenticationServices1_ppt.ppt	Upload	Fri Nov 29 20:02:27 IST 2013
7	ccc xyz	BoiMetrics.ppt.pptx	Upload	Fri Nov 29 20:04:35 IST 2013
8	ccc xyz	Mid-TermResultISS2012-13.xlsx	Upload	Fri Nov 29 20:06:16 IST 2013
9	ccc xyz	IPSec1.ppt.ppt	download	Fri Nov 29 20:12:26 IST 2013
3	ccc xyz	Cryptography & Network Security.pptx	download	Fri Nov 29 19:54:02 IST 2013
4	ccc xyz	TE_Defaulter_list2012-13.xlsx	download	Fri Nov 29 19:54:06 IST 2013
5	ccc xyz	Question Bank_ISS-II.docx	download	Fri Nov 29 19:54:11 IST 2013

Fig.6 Log record specific client.

IV. Results

Our results of the project will be explained well with the help of project work done on number of clients and one main server (master) on GFS architecture based on map reduce framework and then three to five secondary servers(chunk servers).So then we have get these results after comparison with FTP file processing approach on three parameters taken into consideration like

1) Time Utilization.
2) CPU Utilization.
3) Storage Utilization.

Our evaluation examines the improved utilization of Cluster nodes i.e. Secondary servers by uploading and downloading files on GFS Architecture versus FTP from three perspectives. First is improved time utilization and second is improved CPU utilization also the storage utilization also get improved tremendously.

4.1 Results for time utilization

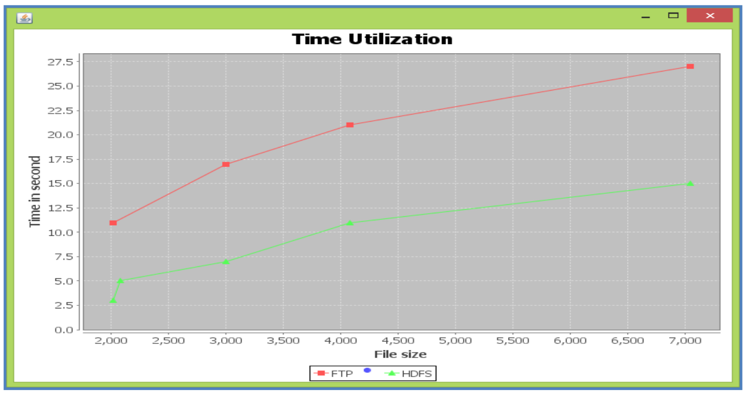


Fig.7 time utilization graph for uploading files.

Fig. 8 shows time utilization for FTP and GFS for uploading files. These are:

Uploading File Size(in Mb)	Time (in sec) for FTP	Time (in sec) for GFS
2	10	2.5
3	17.5	7.5
4.2	20	10
7	27	12.5

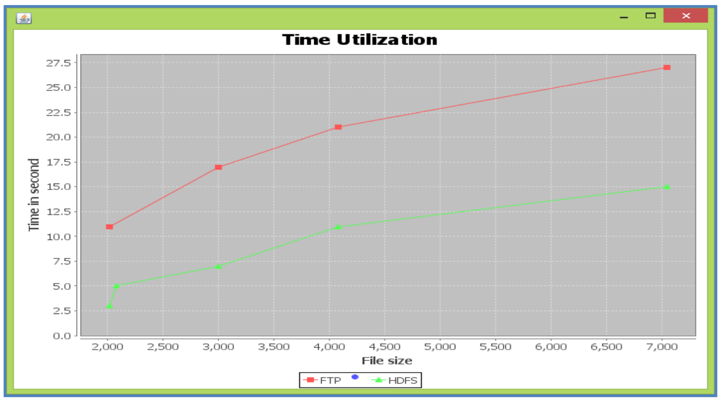


Fig.8 time utilization graph for download files.

Fig. 9 shows time utilization for FTP and GFS for downloading files. These are:

Downloading File Size(in Mb)	Time (in sec) for FTP	Time (in sec) for GFS
2	10	2.5
3	17.5	7.5
4.2	20	10
7	27	12.5

4.2 Results for CPU utilization

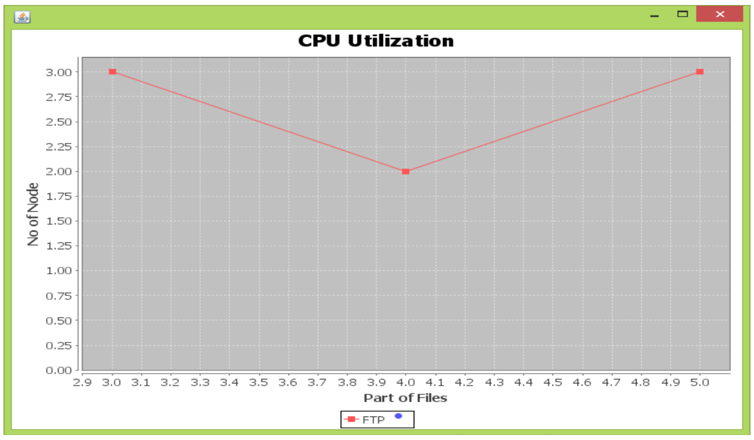


Fig.9 cpu utilization graph for FTP files.

Fig.10 describes the CPU utilization for FTP files on number of clusternodes.

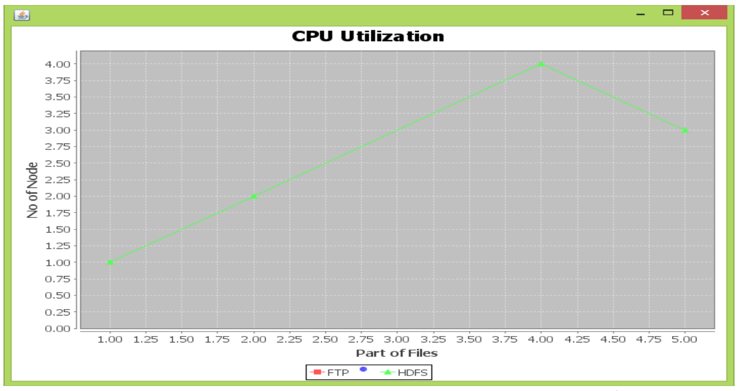


Fig.10 Describes CPU utilization graph on GFS on number of Cluster nodes.

V. Conclusion

We have proposed improved GFS distributed infrastructure that combines On-Demand allocation of resources with improved utilization, opportunistic provisioning of cycles from idle cloud nodes to other processes. A GFS infrastructure using map reduce configuration with improved CPU utilization and storage utilization is proposed using GFS read write Map-Reduce Algorithm. Hence all cloud nodes which remains idle are all get utilized and also improvement in security challenges and achieves load balancing and fast processing of large data in less amount of time. We compare the FTP and GFS for file uploading and file downloading; and enhance the CPU utilization and storage utilization. In this paper, we also proposed some of the techniques that are implemented to protect data and propose architecture to protect data in cloud. This architecture was developed to store data in cloud in encrypted data format using RSA technique which is based on encryption and decryption of data. Till now in many proposed works, there is GFS configuration for cloud infrastructure. But still the cloud nodes remains idle and fault tolerance ,security of data related problems. Hence no such work on CPU utilization for FTP files versus GFS and storage utilization for FTP files versus GFS, we did.
We contribute to an increase of the CPU utilization and time utilization between FTP and GFS. In our work also all cloud nodes are get fully utilized , no any cloud remain idle, also processing of file get at faster rate so that tasks get processed at less amount of time which is also a big advantage hence improve utilization. We also implement RSA algorithm to secure the data, hence improve security.

References

[1]. Sanjay Ghemawat, Howard Gobioff and Shun-Tak Leung, "The Google File System" ACM SIGOPS Operating Systems Review, Volume 37, Issue 5, December 2003..
[2]. Shah, M.A., et.al.,"Privacy-preserving audit and extraction of digital contents", Cryptology ePrint Archive, Report 2008/186 (2008).
[3]. Juels, A., Kaliski Jr., et al.,"proofs ofretrievability for large files",pp. 584–597. ACM Press, New York (2007).
[4]. Sean Quinlan, Kirk McKusick "GFS-Evolution and Fast-Forward" Communications"
[5]. of the ACM, Vol 53, March 2010.
[6]. Divyakant Agrawal et al., " Big Data and Cloud Computing: Current State and Future Opportunities" , EDBT, pp 22-24, March 2011.
[7]. The Apache Software Foundation(2014,07,14). Hadoop[English]. Available:http://hadoop.apache.org/. Jeffrey Dean et al., "MapReduce: simplified data processing on large clusters", communications of the acm, Vol S1, No. 1, pp.107-113, 2008 January.
[8]. Naushad Uzzman, "Survey on Google File System" Conference on SIGOPS at University of Rochester, December 2007.
[9]. Stackoverflow (2014,07,14)." Hadoop Architecture Internals: use of jobandtasktrackers"[English]. Available: http:// stackoverflow.com/questions/11263187/hadooparchitecture-internals-use-of-job-and-task trackers
[10]. J. Dean et al.,"MapReduce: Simplified Data Processing on Large Clusters",In OSDI, 2004
[11]. J. Dean et al., "MapReduce: Simplified Data Processing on Large Clusters", In CACM, Jan 2008.
[12]. J. Dean et al.,"MapReduce: a flexible data processing tool", In CACM, Jan 2010.
[13]. M. Stonebraker et al., "MapReduce and parallel DBMSs: friends or foes?", In CACM. Jan 2010.
[14]. A.Pavlo et al., "A comparison of approaches to large-scale data analysis", In SIGMOD 2009.
[15]. A. Abouzeid et al., "HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads", In VLDB 2009.
[16]. F. N. Afrati et al.,"Optimizing joins in a map-reduce environment",In EDBT 2010.
[17]. P. Agrawal et al., "Asynchronous view maintenance for VLSD databases", In SIGMOD 2009.
[18]. S. Das et al., "Ricardo: Integrating R and Hadoop", In SIGMOD 2010.
[19]. J. Cohen et al.,"MAD Skills: New Analysis Practices for Big Data", In VLDB, 2009.
[20]. Gaizhen Yang et al., "The Application of SaaS-Based Cloud Computing in the University Research and Teaching Platform", ISIE, pp. 210-213, 2011.
[21]. D. Hassan et. al., "A Language Based Security Approach for Securing Map-Reduce Computations in the Cloud", IEEE, pp. 307-308, 2013.
[22]. Chandramohan A. Thekkath, Timothy Mann, and Edward K. Lee. Frangipani: A scalable distributed file system. In Proceedings of the 16th ACM Symposium on Operating System Principles, pages 224–237, October 1997.