International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163
Volume 1 Issue 7 (August 2014) http://ijirae.com

Improved Utilization of Infrastructure of Clouds by using Upgraded Functionalities

Atul U. Patil
M.E CSE, ADCET
Shivaji University, Kolahpur

T.I. Bagban
Asso.Prof. DKTE, Ichalkarnji
Shivaji University, Kolhapur

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. 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 Hadoop configuration with improved CPU utilization and storage utilization is proposed using splitting algorithm 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 HDFS for file uploading and file downloading; and enhance the CPU utilization and storage utilization. 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, Storage utilization, Map-Reduce, Splitting algorithm, Encryption/decryption algorithm.

I. INTRODUCTION

Cloud computing considered as a rapidly emerging new paradigm for delivering computing as a utility. In cloud computing various cloud consumers demand variety of services as per their dynamically changing needs. So it is the job of cloud computing to avail all the demanded services to the cloud consumers. But due to the availability of finite resources it is very difficult for cloud providers to provide all the demanded services. From the cloud providers' perspective cloud resources must be allocated in a fair manner. So, it's a vital issue to meet cloud consumers' QoS requirements and satisfaction. In order to ensure on-demand availability a provider needs to overprovision: keep a large proportion of nodes idle so that they can be used to satisfy an on-demand request, which could come at any time. The need to keep all these nodes idle leads to low utilization. The only way to improve it is to keep fewer nodes idle. But this means potentially rejecting a higher proportion of requests to a point at which a provider no longer provides on-demand computing [2].

Several trends are opening up the era of Cloud Computing, which is an Internet based development and use of computer technology. The ever cheaper and more powerful processors, together with the "software as a service" (SaaS) computing architecture, are transforming data canters into pools of computing service on a huge scale. Meanwhile, the increasing network bandwidth and reliable yet flexible network connections make it even possible that clients can now subscribe high quality services from data and software that reside solely on remote data centers.

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 Hadoop 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 Map-Reducing algorithm, also increase the CPU utilization and storage utilization for uploading files and downloading files. 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 over the past decades. Some of the work done has been discussed, this paper researched cloud computing architecture and its safety, proposed a computing architecture, SaaS model was used to deployed the related software on the cloud platform, so that the resource utilization and computing of scientific tasks quality

will be improved [20].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].

III. THE PROPOSED SYSTEM

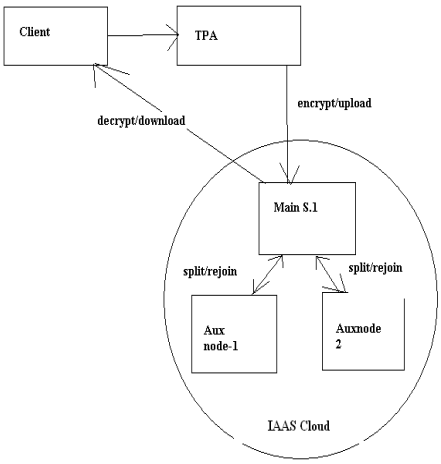

Fig.1 System Architecture

Cloud computing has become a viable, mainstream solution for data processing, storage and distribution, but moving large amounts of data in and out of the cloud presented an insurmountable challenge [4]. Cloud computing is an extremely successful paradigm of service oriented computing and has revolutionized the way computing infrastructure is abstracted and used. Three most popular cloud paradigms include:

1. Infrastructure as a Service (IaaS)

2. Platform as a Service (PaaS)

3. Software as a Service (SaaS)

The concept can also be extended to database as a Service or Storage as a Service. Scalable database management system (DBMS) both for update intensive application workloads, as well as decision support systems are critical part of the cloud infrastructure. Initial designs include distributed databases for update intensive workloads and parallel database systems for analytical workloads. Changes in data access patterns of application and the need to scale out to thousands of commodity machines led to birth of a new class of systems referred to as Key-Value stores[5].

In the domain of data analysis, we propose the Map Reduce paradigm and its open-source implementation Hadoop, in terms of usability and performance. The algorithm has six modules:

1. Hadoop Multinode Configuration(Cloud Server Setup)
2. Client registration and Login facility
3. Cloud Service Provider
4. File Splitting Map-Reduce Algorithm
5. Encryption/Decryption of Data for security
6. Administration of client files(Third Party Auditor)

3.1 Hadoop Configuration (Cloud Server Setup)

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 implements Map Reduce, using the Hadoop Distributed File System (HDFS). The HDFS allows users to have a single addressable namespace, spread across many hundreds or thousands of

servers, creating a single large file system. Hadoop has been demonstrated on clusters with 2000 nodes. The current design target is 10,000 node clusters.

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. The current Apache Hadoop ecosystem consists of the Hadoop kernel, MapReduce, the Hadoop distributed file system (HDFS).

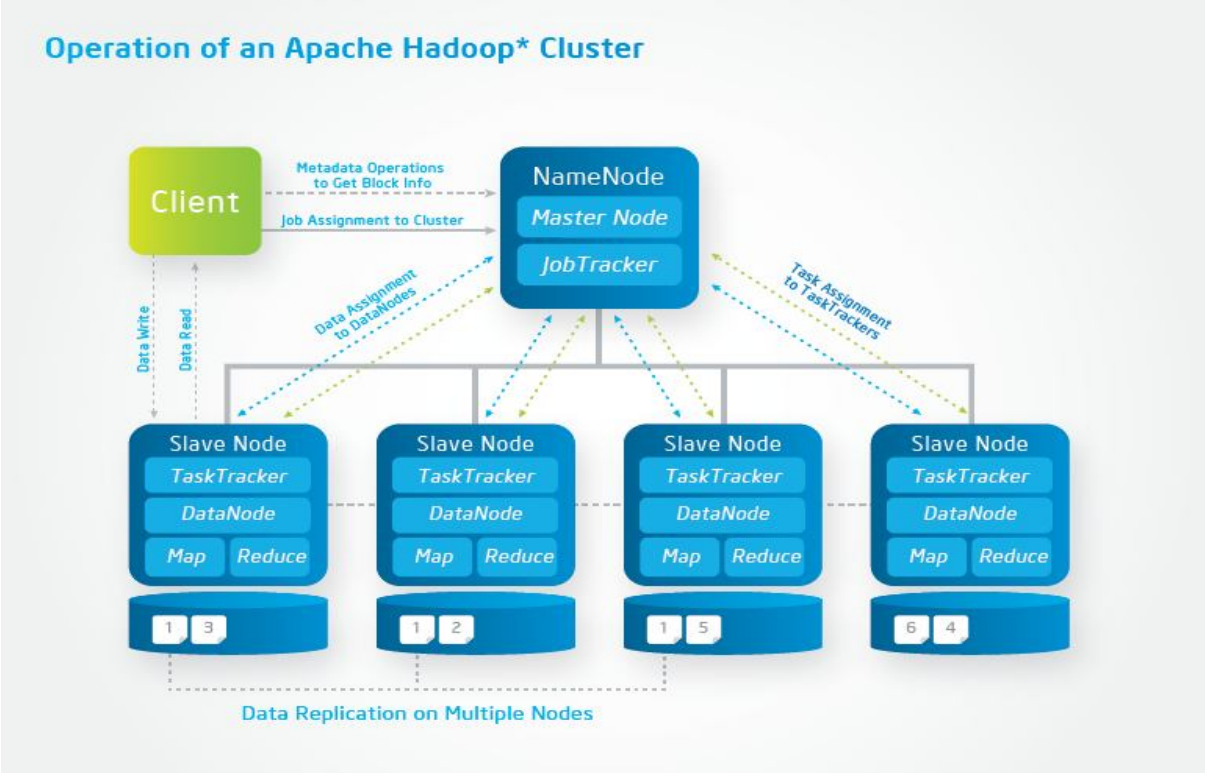

Fig.2 architecture of hadoopdb

JobTracker is the daemon service for submitting and tracking MapReduce jobs in Hadoop. There is only One Job Tracker process run on any hadoop cluster. Job Tracker runs on its own JVM process. In a typical production cluster its run on a separate machine. Each slave node is configured with job tracker node location. The JobTracker is single point of failure for the Hadoop MapReduce service. If it goes down, all running jobs are halted. JobTracker in Hadoop performs, Client applications submit jobs to the Job tracker. The JobTracker talks to the NameNode to determine the location of the data The JobTracker locates TaskTracker nodes with available slots at or near the data The JobTracker submits the work to the chosen TaskTracker nodes. The TaskTracker nodes are monitored. If they do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different TaskTracker. A TaskTracker will notify the JobTracker when a task fails. The JobTracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may even blacklist the TaskTracker as unreliable. When the work is completed, the JobTracker updates its status [9].

A TaskTracker is a slave node daemon in the cluster that accepts tasks (Map, Reduce and Shuffle operations) from a JobTracker. There is only One Task Tracker process run on any hadoop slave node. Task Tracker runs on its own JVM process. Every TaskTracker is configured with a set of slots, these indicate the number of tasks that it can accept. The TaskTracker starts a separate JVM processes to do the actual work (called as Task Instance) this is to ensure that process failure does not take down the task tracker. The TaskTracker monitors these task instances, capturing the output and exit codes. When the Task instances finish, successfully or not, the task tracker notifies the JobTracker. The TaskTrackers also send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the JobTracker that it is still alive. These message also inform the JobTracker of the number of available slots, so the JobTracker can stay up to date with where in the cluster work can be delegated [9].

Namenode stores the entire system namespace. Information like last modified time, created time, file size, owner, permissions etc. are stored in Namenode. The fsimage on the name node is in a binary format. Use the "Offline Image Viewer" to dump the

fsimage in a human-readable format. When the number of files are huge, a single Namenode will not be able to keep all the metadata. In fact that is one of the limitations of HDFS [9].

The current Apache Hadoop ecosystem consists of the Hadoop kernel, MapReduce, the Hadoop distributed file system (HDFS).

The Hadoop Distributed File System (HDFS)

HDFS is a fault tolerant and self-healing distributed file system designed to turn a cluster of industry standard servers into a massively scalable pool of storage. Developed specifically for large-scale data processing workloads where scalability, flexibility and throughput are critical, HDFS accepts data in any format regardless of schema, optimizes for high bandwidth streaming, and scales to proven deployments of 100PB and beyond [8].

Key HDFS 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
- Tunable 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].

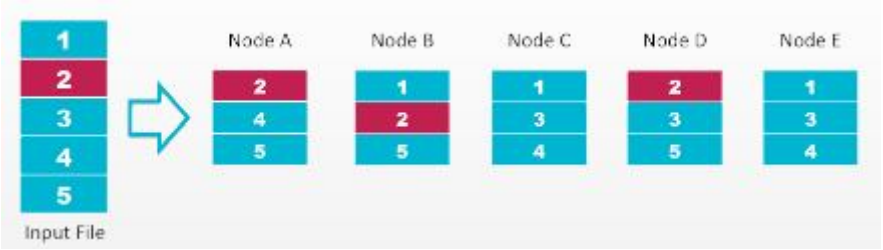

Fig.3 hdfs data distribution [8]

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

3.2 Client registration and Login facility

It provide Interface to Login. Client can upload the file and download file from cloud and get the detailed summery of his account.

3.3 Cloud Service Provider(Administrator)

Administration of User and Data. Authority to Add/Remove user.

3.4 File Splitting Map-Reduce Algorithm

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 File Splitting 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 File Splitting 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 hadoop multinode 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.

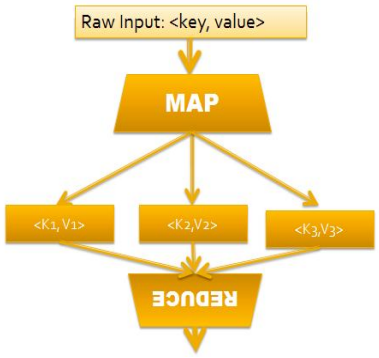

Fig.4 programming framework

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 operations allows 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.

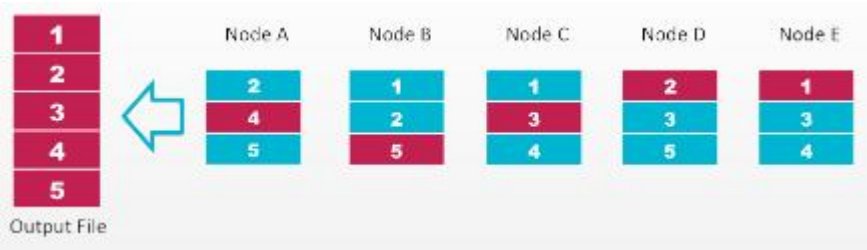

Fig.5 Map-Reduce compute distribution [8]

3.4.1 Programming Model
File Splitting 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].

The Reduce function, also written by the user, accepts the intermediate key and a set of values for that key. It merges together to these values to form a possibly smaller set of values. Typically just zero or one output value is produced per Reduce

invocation. The intermediate values are supplied to the user's reduce function via an iterator. This allow us to handle lists of values that are too large to fit in memory [7].

The map and reduce functions supplied by the user have associated types:

Map (k1,v1) → list (k2,v2)
Reduce (k2, list(v2)) → list (v2)

It means the input keys and values are drawn from a different domain than the output keys and values. Furthermore, the intermediate keys and values are from the same domain as the output keys and values [7]. This process is automatic Parallelization. Depending on the size of RAW INPUT DATA ➔ instantiate multiple MAP tasks. Similarly, depending upon the number of intermediate <key, value> partitions ➔ instantiate multiple REDUCE tasks. Map-Reduce data-parallel programming model hides complexity of distribution and fault tolerance.

3.5 Encryption/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.

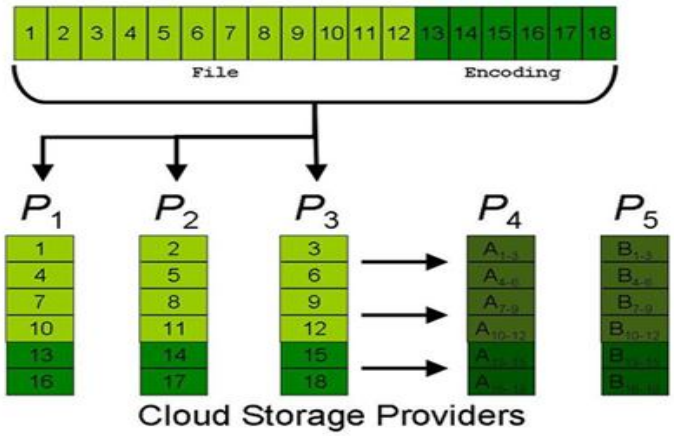

Fig.5 encryption/decryption

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.

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 and then three to five secondary servers so then we have get these results bases on three parameters taken into consideration like

1) Time
2) CPU Utilization
3) Storage Utilization

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

5.1 Results for time utilization

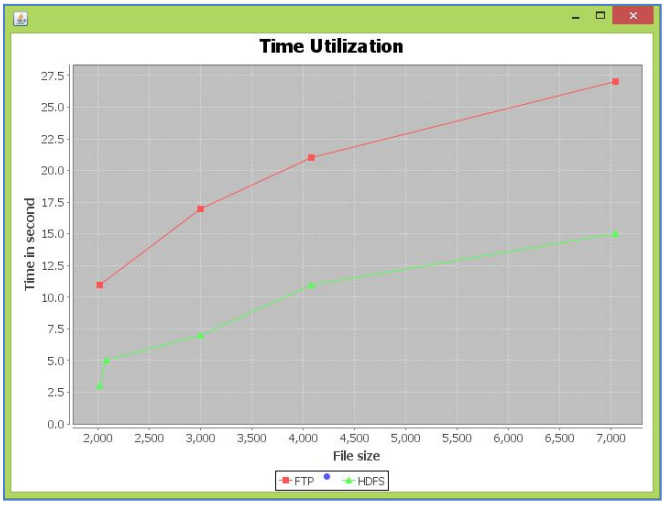

Fig.8 time utilization graph for uploading files

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

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

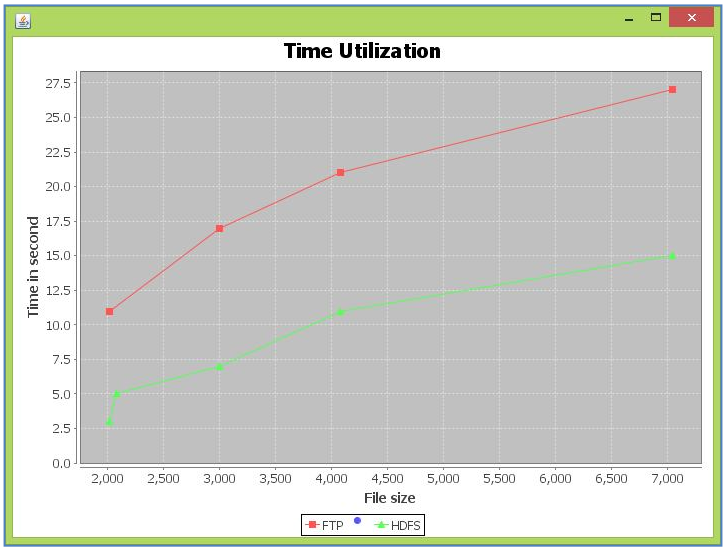

Fig.9 time utilization graph for download files

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

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

5.2 Results for CPU utilization

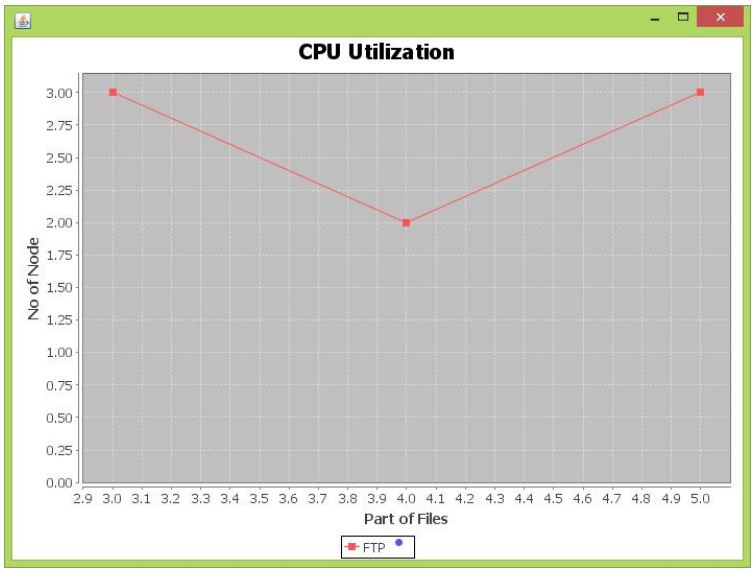

Fig.10 cpu utilization graph for FTP files

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

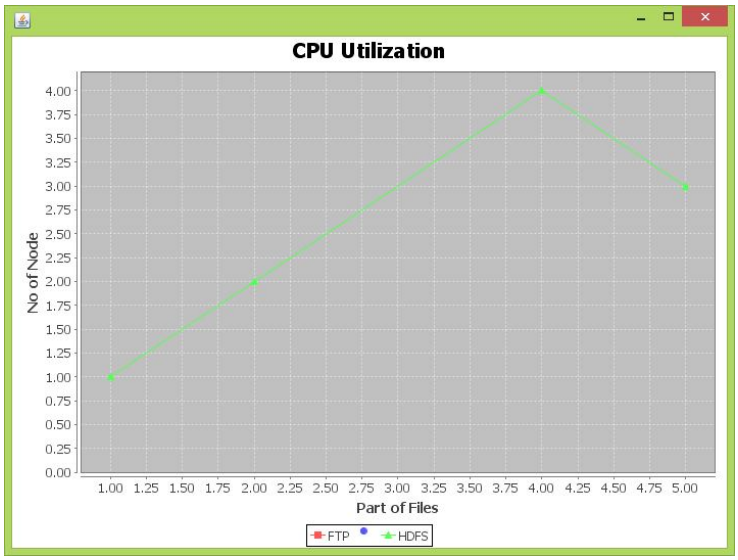

Fig.11 Describes CPU utilization graph on Hadoop HDFS on number of Cluster nodes.

V. CONCLUSION

We have proposed improved cloud infrastructure that combines On-Demand allocation of resources with improved utilization, opportunistic provisioning of cycles from idle cloud nodes to other processes. A cloud infrastructure using Hadoop configuration with improved CPU utilization and storage utilization is proposed using File splitting 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 HDFS for file uploading and file downloading; and enhance the CPU utilization and storage utilization.

Till now in many proposed works, there is Hadoop configuration for cloud infrastructure. But still the cloud nodes remains idle. Hence no such work on CPU utilization for FTP files versus HDFS and storage utilization for FTP files versus HDFS, we did. We evaluate the backfill solution using an on-demand user workload on cloud structure using hadoop. We contribute to an increase of the CPU utilization and time utilization between FTP and HDFS. 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 .

REFERENCES

[1] Paul Marshall "*Improving Utilization of Infrastructure Clouds"*, CO USA, IEEE/ACM Cloud Computing May 2011.
[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 of retrievability for large files", pp. 584–597. ACM Press, New York (2007).
[4] Aspera an IBM company(2014,07,14). Big Data Cloud[English].Available: http://cloud.asperasoft.com/big-data-cloud/.
[5] Divyakant Agrawal et al., " Big Data and Cloud Computing: Current State and Future Opportunities" , EDBT, pp 22-24, March 2011.
[6] The Apache Software Foundation(2014,07,14). Hadoop[English]. Available: http://hadoop.apache.org/.
[7] 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]Clouders(2014,07,14).Cloudera[English].Available: http://www.cloudera.com/content/cloudera/en/products-and-services/cdh/hdfs-and-mapreduce.html
[9] Stack overflow (2014,07,14)."Hadoop Architecture Internals: use of job and task trackers"[English].Available:http://stackoverflow.com/questions/11263187/hadoop architecture-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] Paul Marshall et al., "Improving Utilization of Infrastructure Clouds", IEEE/ACM International Symposium, pp. 205-2014, 2011.
[22] Paul Marshall "*Improving Utilization of Infrastructure Clouds"*, CO USA, IEEE/ACM Cloud Computing May 2011.

