Webpage: www.ijaret.org
Volume 2, Issue IX Sep 2014
ISSN 2320-6802

INTERNATIONAL JOURNAL FOR ADVANCE RESEARCH IN ENGINEERING AND TECHNOLOGY
WINGS TO YOUR THOUGHTS…..

Load Rebalancing for File System in Public Cloud

Roopa R.L[1], Jyothi Patil [2]

[1]PDA College of Engineering,
Gulbarga, Karnataka, India
rlrooparl@gmail.com

[2]PDA College of Engineering,
Gulbarga, Karnataka, India
jyothip_pda2003@yahoo.com

Abstract: *Distributed file systems are the pillars of cloud computing applications. They are based on the MapReduce framework which follows the divide and conquer approach of problem solving. In a cloud computing environment a big and complex file is processed by dividing it into chunks and distributed among the nodes over the network for processing. A central system divides and distributes the chunks. The distributed chunks use commodity hardware and software for processing in parallel. Normally commodity hardware are low cost resources and its quality is compromised over cost. Commodity hardware are prone to failures and needs replacement. This dependence is not accepted in a large-scale, failure-prone environment because the central load balancer is put under considerable workload that is linearly scaled with the system size, and becomes the performance bottleneck and the prime point of failure. Also, files can be dynamically created, deleted and appended in a distributed file system which may need re-dividing and re-distribution over the nodes. Due to this the distribution of chunks is not uniform which may degrade the performance of the application. In the paper, a fully distributed load rebalancing algorithm is presented to cope with the load imbalances.*

Keywords: *load balancing, public cloud, load rebalancing, overload and balancers*.

1. INTRODUCTION

Cloud computing is the technology of the 21st century and it is commonly being used in our day to day life. Almost everyone uses cloud services. NIST defines cloud computing as it is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1].

1.1 Characteristics of cloud computing [8]

There are five key characteristics, namely:

1.1.1 On-demand self-service

A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.

1.1.2 Broad network access

Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations).

1.1.3 Resource pooling

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

1.1.4 Rapid elasticity

Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.

1.1.5 Measured service

Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

1.2 Service models of cloud computing

1.2.1 Software as a Service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

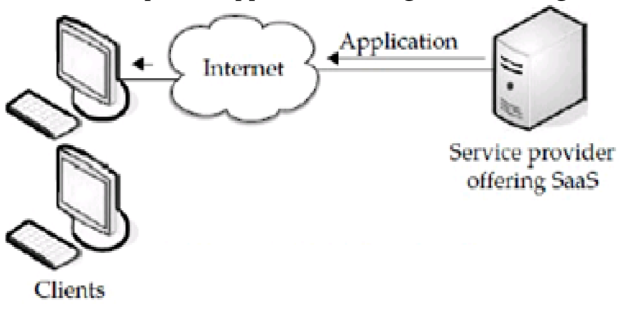

Figure 1: SaaS [2]

1.2.2 Platform as a Service (PaaS)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

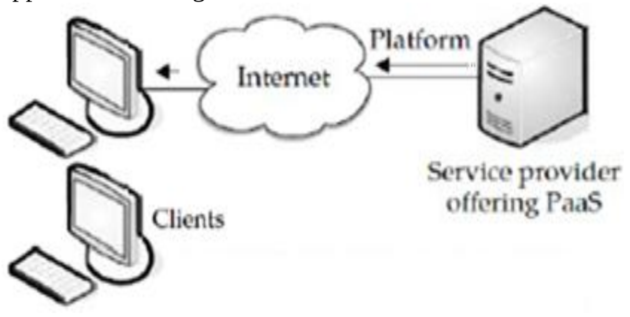

Figure 2: PaaS [2]

1.2.3 Infrastructure as a Service (IaaS)

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

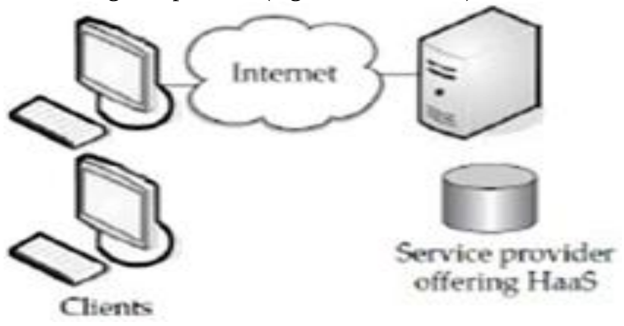

Figure 3: IaaS [2]

1.3 Deployment models of cloud computing

1.3.1 Private cloud

The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises.

1.3.2 Community cloud

The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

1.3.3 Public cloud

The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider [3].

1.3.4 Hybrid cloud

The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).

1.4 Key enabling technologies

Some of the key enabling technologies are:

1.4.1 Fast wide-area networks.

1.4.2 Powerful, inexpensive server computers.

1.4.3 High-performance virtualization for commodity hardware.

Commodity hardware are the lost cost resources whose quality is compromised over cost. Due to this they fail very often. Commodity hardware are easy to replace without ceasing the work in progress. These commodity resources are also called as nodes. These nodes are connected to one another in the form of internet. Internet is the backbone of cloud computing services. In a public cloud different types of workloads are run and due to the increased deployment of applications along with the extensive increase in the user base overloading has become a common and iterative problem.

1.5 Load balancing

Load balancing is essential for efficient operations in distributed environments. It means distributing the amount of work to do between different servers in order to get more work done in the same amount of time and serve clients faster. In this case, consider a large-scale distributed file system. The system contains *N* chunk-servers in a cloud (N can be 1000, 10000, or more), where a certain number of files are stored. Each file is split into several parts or chunks of fixed size (for example 64 megabytes). The load of each chunk server is proportional to the number of chunks hosted by the server. In a load-balanced cloud, the resources can be well used while maximizing the performance of MapReduce based applications.

a. Load re-balancing

In a cloud computing environment, failure is the norm, and chunk servers may be upgraded, replaced, and added in the system. Files can also be dynamically created, deleted, and appended. That leads to load imbalance in a distributed file system, meaning that the file chunks are not distributed equitably between the nodes.

Distributed file systems in clouds such as GFS and HDFS rely on central servers (master for GFS and NameNode for HDFS) to manage the metadata and the load balancing. The master rebalances replicas periodically: data must be moved from a Data Node/chunk server to another one if its free space is below a certain threshold. However, this centralized approach can provoke a bottleneck for those servers as they become unable to manage a large number of file accesses. Consequently,

dealing with the load imbalance problem with the central nodes complicates more the situation as it increases their heavy loads.
In order to manage large number of chunk servers to work in collaboration, and solve the problem of load balancing in distributed file systems, several approaches have been proposed such as reallocating file chunks such that the chunks can be distributed to the system as uniformly as possible while reducing the movement cost as much as possible.

1.7 Comparison between load balancing and rebalancing

Data load balancing takes place at the time of arrival of new data, Service broker based on the information about existing loads on the datacenters (from Metadata) calculates the chunks and the replicas that needs to be saved in each datacenter. However at the time of taking decision, several other decision of include and delete of the files takes place which literally dis-balances the load available at the nodes. For this HDFS file system is considered as reply to such a system. HDFS is used for large scale data and using Map reduce the data is compressed. A rebalancing algorithm is applied periodically. It is performed on the existing storage cluster to reorganize the files such that they remain organized.

2. OBJECTIVES

Load rebalancing problem in distributed file systems are specialized for large-scale, dynamic and data-intensive clouds is addressed. Such a large-scale cloud has hundreds or thousands of nodes. Following are the objectives:

- To simulate a cloud with distributed file system.
- Files are divided as chunks and stored in servers such that the chunk servers has equal load.
- Commodity hardware are prone to failures and hence they are replaced very often.
- Several files are added or deleted or modified and hardware are replaced so after some time cloud is in imbalanced state.
- To avoid such scenario rebalancing is initiated by chunk servers without disturbing broker to bring the cloud under balanced state.
- Rebalancing depends upon moving chunks from one chunk server and adding it to another.
- Normally files are divided into chunks of fixed maximum size and are stored into different chunk servers and the objective is to reduce number of chunks per file by applying mapreduce technique.
- To demonstrate the physical storage of such chunk servers we use hadoop.
- Expected outcome using minimum number of iterations and chunk movement cloud should be under balanced state.

3. RELATED WORK

In MapReduce: Simplified Data Processing on Large Clusters[4] proposed by Jeffrey Dean and Sanjay Ghemawat presents an approach which is based on processing and generating large data sets by a mapreduce programming model. Mapreduce is a data processing model. Mapreduce is a data processing component of hadoop, 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.
In The Google File System [5] proposed by Sanjay Ghemawat, Howard Gobioff and Shun-Tak Leung presents a file system that has successfully met storage needs. It is widely deployed within google as the storage platform for the generation and processing of data used by the services as well as research and development efforts that require large data sets. The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over thousand machines, and it is concurrently accessed by hundreds of clients. The author has presented file system interface extensions designed to support distributed applications, discuss many aspects of our design, and report measurements from both micro-bench marks and real world use.
In Hadoop Distributed File System [6] Proposed by Thomas Kiencke institute of Telematics,University of Lubeck,Germany presents a set of tools that supports running of applications on bigdata. These days internet has become an important part in our life as a result companies have to deal with millions of data sets. These datasets have to be stored safely and should be fault tolerant. At the same time hardware should be very cheap. To be faced with this contrast, HDFS[6] was developed and fulfills those requirements. The highly available and fault tolerant framework is designed to run on commodity hardware where failure is the normal rather than exception. It provides the data replication plus integrity and several interfaces to interact with. HDFS is written in java and licensed under the open-source Apache V2 license. This means that no particular company is controlling the distribution of hadoop and it is maintained by Apache. Hadoop challenges are created at three levels i.e.,Velocity, Volume and Variety. Hadoop takes a very different approach then the enterprise approach.
In Simple Load Balancing For Distributed Hash Tables [7] Proposed by J.W Byers, J. Considine, and M. Mitzenmacher describes that distributed hash tables have recently become a useful building block for a variety of distributed applications. However, current schemes based upon consistent hashing require both considerable implementation complexity and substantial storage overhead to achieve desired load balancing goals. These goals can be achieved more simply and more cost-effectively. Firstly, the direct application of the power of two choices" paradigm is suggested, whereby an item is stored at the less loaded of the two (or more) random alternatives then associating a small constant number of hash values with a key can naturally be extended to support other load balancing methods, including load-stealing or load-shedding schemes, as well as providing natural fault-tolerance mechanisms.

4. NEW PROPOSED SCHEME

In the current system the distributed file systems (e.g., Google GFS and Hadoop HDFS) in clouds rely on central nodes to manage the metadata information of the file systems and to balance the loads of storage nodes based on that metadata. The centralized approach simplifies the design and implementation of a distributed file system. Here files are first divided into chunks and then are saved directly through MapReduce to Hadoop. All the merging, distributing and other file services are performed directly on the HDFS file system. The major problem with this technique is that once the files and data are saved, never again its sanctity is checked. Different operations in the network (UPDATE, INSERT, and DELETE) operations changes the data balance hence queries takes more time. On the other hand now a load balancing algorithm calculates the load on every datacenter's HDFS file system and distributes the data such that the chunks are in balance. When network is idle, a background process performs the rebalancing operations. Rebalancing operations are performed many times to keep the data centers and the data in sync and balanced. Due to this, the queries are processed faster than the recent system. This is shown in figure 4.

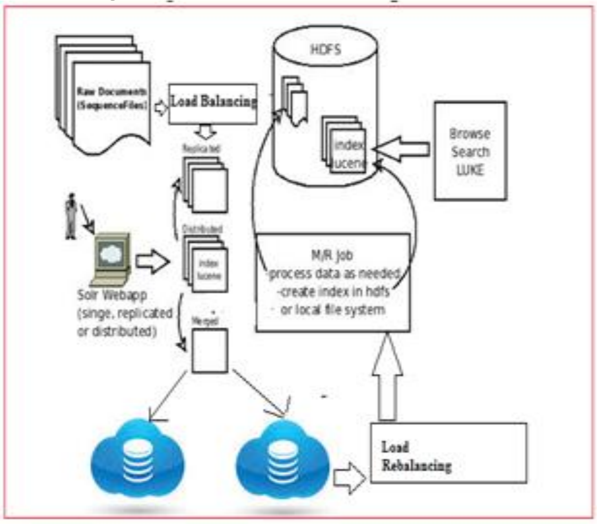

Figure 4: Data storage system with load rebalancing

The algorithm designed for the above system model is given below.

4.1 Algorithm

Step 1: Node i of each chunk-server estimates whether it is over-loaded (heavy) or under-loaded (light) without having the global knowledge.

Step 2: A node is said to be light if the number of chunks it hosts is smaller than the threshold of $(1-\Delta_L)A$ (where $0 \leq \Delta_L < 1$).

Step 3: A node is said to be heavy node if it manages the number of chunk greater than $(1+\Delta_V)A$, Where $0 \leq \Delta_V < 1$.Here Δ_L and Δ_V are system parameters.

Step 4: If the node i departs and rejoins as a successor of another node j, then node I is represented as node j+1, node j's original successor as j+2,the successor of j's original successor as node j+3 and so on.

Step 5: If node I is light, it takes at most A nodes from the heavy chunks.

Step 6: First we present a load balancing algorithm in which each node has a global knowledge.

Step 7: We then extend the algorithm where the global knowledge is not present and migration of light and heavy nodes are been done iteratively until no need remains heavier or lighter.

5. IMPLEMENTATION AND EXPERIMENTAL RESULT

Keeping the datacenters constant and varying the number of users interpreting the change in Transfer cost, Balance time and Rebalance time. Graph showing the results of proposed system where the time taken to balance and rebalance the load is lesser as compared to the present system and hence it can be said that there is a minimization in the data transfer rate.

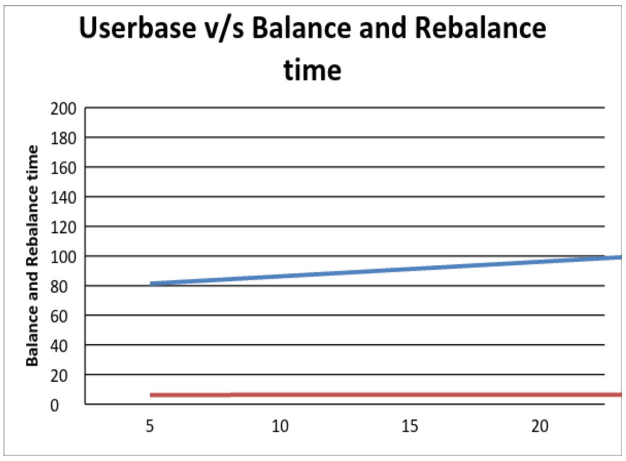

Figure 5: Graph comparing balance and re balancing time

The table below shows various parameters used and their respective values.

Table1: Simulation parameters

Parameters	**Values**
User Base	5-25
Simulation time	1000 minutes
Data center	5-25
Load Balancing Technique	Round robin
Scheduling Technique	Closest data center
Vm/datacenter	5-10

6. CONCLUSION

This algorithm strives to balance the loads of nodes and reduce the demanded movement cost as much as possible, while taking advantage of physical network locality and node heterogeneity. Particularly, the load-balancing algorithm exhibits a fast convergence rate. The efficiency and effectiveness of the design are further validated by analytical models.

REFERENCES

[1] P. Mell and T. Grance, The NIST definition of cloud computing, http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, 2012.
[2] www.images.google.com.
[3] http: //en.wikipedia.org /wiki/Public_cloud
[4] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proc. Sixth Symp. Operating System Design and Implementation (OSDI'04), pp. 137-150, Dec. 2004.
[5] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," Proc. 19th ACM Symp. Operating Systems Principles (SOSP'03), pp. 29-43, Oct. 2003.
[6] Hadoop Distributed File System, http://hadoop.apache.org/ hdfs/, 2012.
[7] J.W. Byers, J. Considine, and M. Mitzenmacher, "Simple Load Balancing for Distributed Hash Tables," Proc. First Int'l Workshop Peer-to-Peer Systems (IPTPS '03), pp. 80-87, Feb. 2003.
[8] Abhijeet G Purohit, Md. Abdul Waheed, Asma Parveen, "Load Balancing In Public Cloud By Division Of Cloud Based On The Geographical Location," Proc. NCRIET'14: National Conference on Recent Innovations in Engineering and Technology-2014.