IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. III (Mar-Apr. 2014), PP 81-86
www.iosrjournals.org

Survey on Load Rebalancing For Distributed File Systems in Clouds

Ms. Nithya Kuriakose[1], Ms. Shinu Acca Mani[2]
[1, 2](Department of Computer Science and Engineering Nehru College of Engineering and Research Center, Pampady, Thrissur, Kerala)

Abstract: *Cloud Computing is an emerging technology, it is based on demand service in which shared resources, information, software and other devices are provided according to the clients to the requirements at specific time with the availability of internet. Load balancing is one of the challenging issue in cloud computing. An efficient load balancing makes cloud computing more efficient and improves user satisfaction. It includes fault tolerance, high availability, scalability, flexibility, reduced overhead for users, reduced cost of ownership, on demand services etc. Distributed file systems are key building blocks for cloud computing applications based on the Map Reduce programming paradigm. In such file systems, nodes at the same time serve computing and storage functions. Files can be created, deleted, and appended dynamically. This results in load imbalance in a distributed file system; that is, the file chunks are not distributed uniformly as possible among the nodes.*

Keywords: *Cloud computing, Load rebalancing, Distributed File system*

I. Introduction

Cloud computing is a model to provide convenient, on-demand access to a shared pool configurable computing resources. In cloud computing, IT-related capabilities are provided as services, accessible without requiring detailed knowledge of the underlying technologies, and with minimal management effort. A cloud system is also user friendly, in the respect that it requires less expertise to use. It is sold on demand, typically by the minute or the hour. A cloud can be private or public. A public cloud sells services to anyone on the Internet. A private cloud is a proprietary network or a data center that supplies hosted services to a limited number of people. Private or public, the goal of cloud computing is to provide easy, scalable access to computing resources and IT services. Cloud computing provide everything as a service to their users, like as: storage of data as a service, application software as a service, computing platform as a service and computing infrastructure as a service etc. Up-and-coming distributed file systems in production systems strongly depend on a central node for chunk reallocation. This dependence is clearly insufficient in an extensive, failure-prone environment because the central load balancer is put under considerable workload that is linearly scaled with the system size, and may thus become the performance bottleneck and the single point of failure. The aim to reduce network traffic or movement cost caused by rebalancing the loads of nodes as much as possible to maximize the network bandwidth available to normal applications. Moreover, as failure is the norm, nodes are newly added to sustain the overall system performance resulting in the heterogeneity of nodes. Exploiting capable nodes to improve the system performance is thus demanded.

The popular file system for networked computers is the Network File System. It is a way to share files between machines on a network as if the files were located on the client's local hard drive. Frangipani is a scalable distributed file system that manages a collection of disks on multiple machines as a single shared pool of storage. The machines are required to be under a common administrator and be able to communicate securely. The first one is that it depends on a single name node to manage almost all operations of every data block in the file system. As a result it can be a bottleneck resource and a single point of failure. [1]

II. Related Work

Distributed Hash Tables are key building block for variety of distributed applications. It uses the hashing approach; both the keys and peers are hashed onto a 1D ring. Keys are then assigned to the nearest peer in the clockwise direction. Servers connected to their neighbors in the ring and searching for a key reduces to traversing ring this result a considerable load imbalance. One of the solutions is the use of virtual peers that is for each peers, assigning number of virtual peers. In this case large size request may not be processed because of the tightly bounded expected value. Substitute solution is that of power of two choice paradigms. In this paradigm use standard hashing scenarios using bins to reduce or balance the load. Less shared routing information stored at each peer. [2] The use of range partitioning can make partitioning a dynamic relation across a large number of disks/nodes. Range portioning is frequently popular in large scale parallel as well as peer-to-peer databases. Load balancing is necessary in such scenarios to eliminate skew. This introduces asymptotically optimal online load-balancing algorithms that guarantee a constant imbalance ratio. The data movement cost per tuple insert or delete is constant, and was shown to be close to 1 in experiments. Advantages are Decentralized System, Automatically performs all operations, Avoid Data Skew. One of the disadvantages is that it take too much of time to complete the task. [3] Antony Rowstron et al presents the design and evaluation of Pastry, a scalable, distributed object location and routing scheme for wide-area peer-to-peer applications. Pastry performs application level routing and object location in a potentially very large overlay network of nodes connected via the Internet. In application level routing, different applications will have different requirements according to that routing is performed. For example video conferencing requires high requirements, if any one use this path the requirement will decreases and hence leads to complete no sharing of path. In the case of low requirement application such as email and text messages gives a busy path. According to the requirement application the routing is performed. It can be used to support a wide range of peer-to-peer applications like global data storage, global data sharing, and naming. Advantages are Decentralized System, Automatically performs all operations and one of the disadvantages is that Every time lookup operation is needed. [4]

David R. Karger et al have given several provably efficient load balancing protocols for distributed data storage in P2P systems. Algorithms are simple, and easy to implement, so an obvious next research step should be a practical evaluation of these schemes. In addition, several concrete open problems follow from our work. First, it might be possible to further improve the consistent hashing scheme. It uses the hashing approach; both the keys and peers are hashed onto a one dimensional ring. Keys are then assigned to the nearest peer in the clockwise direction. Servers connected to their neighbors in the ring and searching for a key reduces to traversing ring this result a considerable load imbalance. One of the solutions is the use of virtual peers that is for each peers, assigning number of virtual peers. In this case large size request may not be processed because of the tightly bounded expected value. Second, our range search data structure does not easily generalize to more than one order. For example when storing music files, one might want to index them by both artist and song title, allowing lookups according to two orderings. It provides efficient load balancing but hard to achieve. [5] Jeffrey Dean et al introduce the MapReduce programming model has been successfully used at Google for many different purposes. Attribute this success to several reasons. First, the model is easy to use, even for programmers without experience with parallel and distributed systems, since it hides the details of parallelization, fault-tolerance, locality optimization, and load balancing. Second, a large variety of problems are easily expressible as MapReduce computations. MapReduce is the programming model and associated implementation for processing and generating large data sets. Users specify a map function that processes a key-value pair to generate a set of intermediate key-value pairs and reduce function that merges all intermediate value associated with the same intermediate key [6]. It has been emerging as a popular paradigm for data intensive computing in clustered environment such as enterprise data centers and cloud which solves parallel problems using large number of computers collectively called as cluster. Advantages are highly scalable, Can compute large data set, but it is Expensive, More time to compute the reducing functions.

The different qualitative metrics or parameters that are considered important for load balancing in cloudcomputing [10] are Throughput, Associated overhead, Fault tolerant, Migration time, Response time, Resource utilization, Scalability, and Performance. The major concerns of cloud computing that is Load balancing. The goal of load balancing is to increase client satisfaction and maximize resource utilization and substantially increase the performance of the cloud system thereby reducing the energy consumed and the carbon emission rate. Also the purpose of load balancing is to make every processor or machine perform the same amount of work throughout which helps in increasing the throughput, minimizing the response time and reducing the number of job rejection. Comparisons of papers are shown in the table 1 bellow.

Section	**Method**	**Advantages**	**Disadvantages**
Pastry: Scalable, distributed object location and routing for large-scale P2P systems	Distributed File System	Decentralized System Automatically performs all operations	Every time lookup operation is needed
Online Balancing of Range-Partitioned Data with Applications to P2P	Range Partitioning	Decentralized System Automatically performs all operations Avoid Data Skew	Take too much of time to complete the task
Simple Efficient Load Balancing Algorithms for P2P Systems	Hashing scheme and Range Search DS	Provide efficient load balancing	Difficult to achieve
Simple Load Balancing for DHT	Power of two choice	Can take more number of keys	Less shared routing information
MapReduce: Simplified Data Processing on Large Clusters	MapReduce Programming model	Highly scalable Can compute large data set	Expensive More time to compute the reducing function

Table 1: Comparison table

III. Load Rebalancing

Load balancing is the process of distributing the load among various nodes of a distributed system to improve both resource utilization and job response time while also avoiding a situation where some of the nodes are heavily loaded while other nodes are idle or doing very little work. Load balancing ensures that all the processor in the system or every node in the network does approximately the equal amount of work at any instant of time. This technique can be sender initiated, receiver initiated or symmetric type. The objective is to develop an effective load balancing algorithm using divisible load scheduling theorem to maximize or minimize different performance parameters for the clouds of different sizes. It is a process of reassigning the total load to the individual nodes of the collective system to make resource utilization effective and to improve the response time of the job, simultaneously removing a condition in which some of the nodes are over loaded while some others are under loaded. A load balancing algorithm which is dynamic in nature does not consider the previous state or behavior of the system, that is, it depends on the present behavior of the system. The important things to consider while developing such algorithm are : estimation of load, comparison of load, stability of different system, performance of system, interaction between the nodes, nature of work to be transferred, selecting of nodes and many other ones.

Load rebalancing eliminates the dependence on central nodes. The storage nodes are structured as a network based on distributed hash tables (DHT). DHTs enable nodes to self-organize and repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management. The algorithm is compared against a centralized approach in a production system and a competing distributed solution presented in the literature. The simulation results indicate that although each node performs our load rebalancing algorithm independently without acquiring global knowledge. Specifically, in this study, suggest offloading the load rebalancing task to storage nodes by having the storage nodes balance their loads spontaneously. This eliminates the dependence on central nodes. The storage nodes are structured as a network based on distributed hash tables discovering a file chunk can simply refer to rapid key lookup in DHTs, given that a unique handle is assigned to each file chunk. DHTs enable nodes to self-organize and repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management. It includes chunk creation, DHT formulation, Load balancing algorithm, Replica Management. [7][8][9][11][12]

A. Chunk creation: A file is partitioned into a number of chunks allocated in distinct nodes so that Map Reduce Tasks can be performed in parallel over the nodes. The load of a node is typically proportional to the number of file chunks the node possesses. Because the files in a cloud can be arbitrarily created, deleted, and appended, and nodes can be upgraded, replaced and added in the file system, the file chunks are not distributed as uniformly as possible among the nodes. Objective is to allocate the chunks of files as uniformly as possible among the nodes such that no node manages an excessive number of chunks.

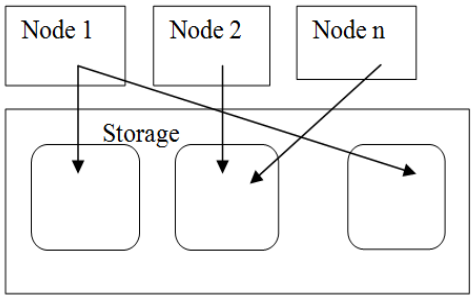

Fig 1: Chunk Creation Module

B. DHT formulation: The storage nodes are structured as a network based on DHTs, e.g., discovering a file chunk can simply refer to rapid key lookup in DHTs, given that a unique handle is assigned to each file chunk. DHTs enable nodes to self-organize and Repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management. The chunk servers in our proposal are organized as a DHT network. Typical DHTs guarantee that if a node leaves, then its locally hosted chunks are reliably migrated to its successor; if a node joins, then it allocates the chunks whose IDs immediately precede the joining node from its successor to manage.

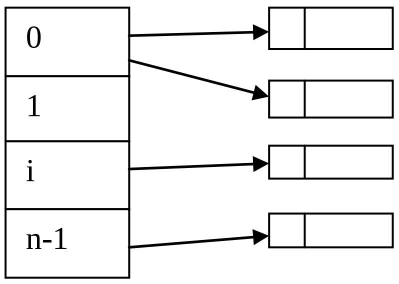

Fig 2: DHT Formulation Module

C. Load balancing algorithm: Each chunk server node I first estimate whether it is under loaded or overloaded without global knowledge. A node is light if the number of chunks it hosts is smaller than the threshold. Load statuses of a sample of randomly selected nodes. Specifically, each node contacts a number of randomly selected nodes in the system and builds a vector denoted by V. A vector consists of entries, and each entry contains the ID, network address and load status of a randomly selected node.

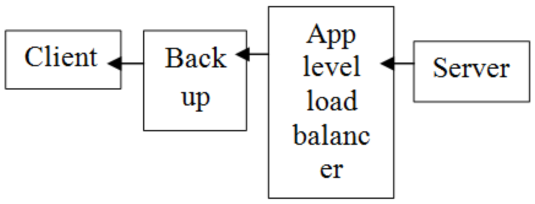

Fig 3: Load Balancing Module

D. Replica Management: In distributed file systems, a constant number of replicas for each file chunk are maintained in distinct nodes to improve file availability with respect to node failures and departures. Our current load balancing algorithm does not treat replicas distinctly. It is unlikely that two or more replicas are placed in an identical node because of the random nature of our load rebalancing algorithm.

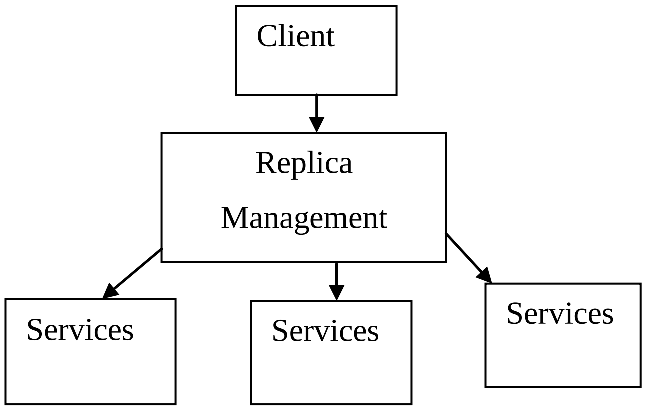

Fig 4: Replica Management Module

IV. Load Balancing Algorithm

1. Initialize server and its sub-servers
2. Establish connection between sub-server and servers using the IP or Port number.
3. Upload File to server that should be shared.
4. Split the file into multiple chunks
5. Calculate the each sub server memory
6. Divide the total chunks value by total number of sub-servers
7. Upload each chunk into sub servers based on its memory capacity
8. If Capacity is less then transfer the excess chunks into next sub-servers
9. Each chunk will be appended with a index value.

10. When the client request for a file, that will be received from different sub-servers based on the index value.
11. Client collects all the chunks then the file will be decrypted, then that will be viewed by client.
    Fig 5 shows the System Architecture, clients can upload and download the files from the main server through the sub server. The system is centralized so that only through the sub server the operations are performed.

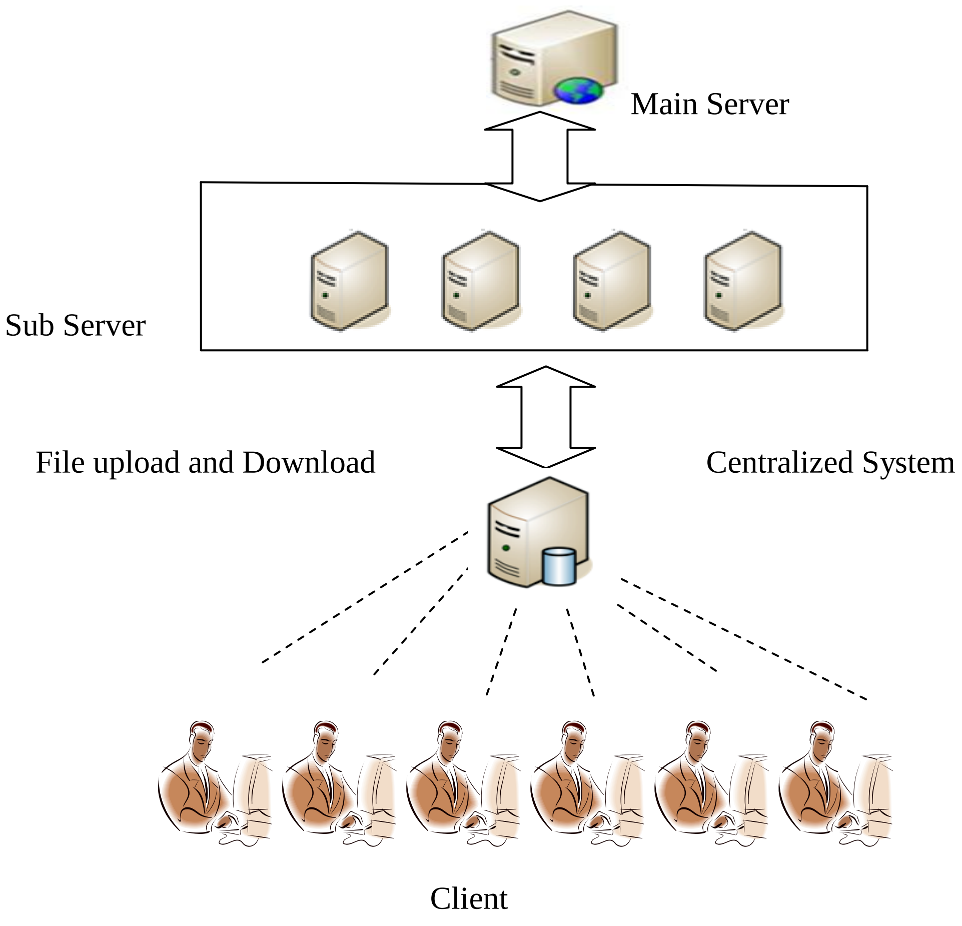

Fig 5: System Architecture

V. Conclusion

The proposal 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. In the absence of representative real workloads (i.e., the distributions of file chunks in a large scale storage system) in the public domain, we have investigated the performance of the proposal and compared it against competing algorithms through synthesized probabilistic distributions of file chunks. Emerging distributed file systems in production systems strongly depend on a central node for chunk reallocation. This dependence is clearly inadequate 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 may thus become the performance bottleneck and the single point of failure. The algorithm is compared against a centralized approach in a production system and a competing distributed solution presented in the literature. The simulation results indicate that the proposal is comparable with the existing centralized approach and considerably outperforms the prior distributed algorithm in terms of load imbalance factor, movement cost, and algorithmic overhead, a fully distributed load rebalancing algorithm is presented to cope with the load imbalance problem. In future increase efficiency and effectiveness of design are further validated by analytical models and a real implementation with a small-scale cluster environment. Highly desirable to improve the network efficiency by reducing each user's download time.

References

[1] Hung-Chang Hsiao, Hsueh-Yi Chung,Haiying Shen, and Yu-Chang Chao,"Load Rebalancing for Distributed File Systems in Clouds," IEEE Transactions On Parallel And Distributed Systems, Vol. 24, No. 5, May 2013.

[2] 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.

[3] P. Ganesan, M. Bawa, and H. Garcia-Molina, “Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems,” Proc. 13th Int’l Conf. Very Large Data Bases (VLDB ’04), pp. 444-455, Sept. 2004.

[4] A. Rowstron and P. Druschel, “Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems,” Proc. IFIP/ACM Int’l Conf. Distributed Systems Platforms Heidelberg, pp. 161-172, Nov. 2001.

[5] D. Karger and M. Ruhl, “Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems,” Proc. 16th ACM Symp. Parallel Algorithms and Architectures (SPAA ’04), pp. 36-43, June 2004.

[6] 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.

[7] T.Sakthisri and S.Pallavi, “Balancing Blocks For Distributed File System In Clouds By Using Load Rebalancing Algorithm,” Proc. International Conference on Information Systems and Computing (ICISC-2013), pp. 220, Jan. 2013.

[8] Revathy R and A.Illayarajaa, “Efficient Load Re Balancing Algorithm for Distributed File Systems,” Proc. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-2, Issue-6, May 2013.

[9] Yatendra Sahu and R.K. Pateriya, “Cloud Computing Overview With Load Balancing Techniques,” Proc. International Journal Of Computer Applications (0975 – 8887) Volume 65– No.24, March 2013.

[10] Aarti Khetan, Vivek Bhushan and Subhash Chand Gupta, “A Novel Survey On Load Balancing In Cloud Computing,” Proc. International Journal Of Engineering Research & Technology (IJERT) ISSN: 2278-0181, Vol. 2 Issue 2, February 2013.

[11] S.Indira and P.Jyothi, “Load Rebalancing In Large-Scale Distributed File System,” Proc. International Journal Of Reviews On Recent Electronics And Computer Science (IJRRECS), Volume-1, Issue-6, Issn 2321-5461, October 2013.

[12] Ch. Mounika, L. RamaDevi and P.Nikhila, “Simple Load Rebalancing For Distributed Hash Tables In Cloud,” Proc. IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN: 2278-0661, p- ISSN: 2278-8727 Volume 13, Issue 2, pp 60-65, Jul. - Aug. 2013.