International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

Survey Paper on Load Rebalancing for Distributed File Systems in Clouds

Juhi Shah[1], D. N. Rewadkar[2]

[1]RMD Sinhgad School of Engineering, Warje, Pune - 411058, India
[2]RMD Sinhgad School of Engineering, Head of the Computer Department

Abstract: ***Distributed file systems (DFS) are key building blocks for cloud computing applications based on the MapReduce programming paradigm. In Distributed file systems (DFS), nodes simultaneously serve computing and storage functions; a file is partitioned into a number of chunks and allocated in distinct nodes so MapReduce tasks can be performed in parallel over the nodes. And in a cloud failure is the norms/files, and nodes/files may be upgraded, replaced, added in the system. Files can also be dynamically created, deleted, and updated/appended. However, this results in load imbalance in a distributed file system (DFS); that is; the file chunks are not distributed as uniformly as possible among the nodes/files. Emerging distributed file systems in production systems strongly depend on the central node for chunk reallocation/migration. And this dependence is clearly inadequate in a large-scale, failure-prone environment because of the central load balancer is put under considerable workload that is linearly scaled with the system size. This may thus become the performance bottleneck and the single point of failure in DFS. In this paper, a fully distributed load rebalancing algorithm is used to present to cope with the load imbalance problem. Our algorithm is compared against a centralized approach in a production system and a competing distributed solution presented in the literature (related work). The simulation results indicate that our proposal system is comparable with the existing centralized approach to and considerably outperforms the prior distributed algorithm in terms of load imbalance factor, movement cost, and algorithmic overhead.***

Keywords: Load balance, Distributed File Systems, Cloud, Distributed Hash Table, MapReduce

1. Introduction

In Cloud computing, the number of computers that are connected using communication network. The notation of cloud indicates that internet is mandatory to perform the various cloud operations i.e. to create append/update, delete, and replace. It is used in IT-companies to share information and resources with the all users on network. There are various characteristics of cloud i.e. Scalable, on demand service, Versatile, User centric, Powerful, Platform independent etc. In cloud three technologies are included the Virtualization, MapReduce programming, and distributed file systems for the data storage purpose.

Distributed file system (DFS) is classical model of file system that is used in the form of chunks for cloud computing. Cloud application is based on the MapReduce programming used in distributed file system (DFS). MapReduce is the master-slave architecture in Hadoop. Master act like Namenode and Slave act like Datanode. Master takes large problem, divides it into the sub problem and assigns it to worker node i.e. to multiple slaves to solve problem individually. InDistributed file system, a large file is divided into number of chunks and allocates each chunk to separate node to perform MapReduce function parallel over each node. For example in word count application it identifies the occurrences of each distinct word in large file. In this application a large file is divided into fixed-size chunks (parts) and assigns each chunk (parts) to different cloud storage node. Then each storage node calculates the occurrences of each distinct word by scanning or parsing its own chunk. Then give its result to master to calculate the final result. In distributed file system, the load of each node is directly proportional to number of file chunks/parts that node consists. As the increase in storage and network, load balancing is the main issue in the large scale distributed systems. Load should be balance over multiple nodes to improve system performance, resource utilization, response time, cost and stability. Load balancing is divided into two categories: static and dynamic. In static load balancing algorithm, it does not consider the previous behavior of a node while distribute the load. But in case of dynamic load balancing algorithm, it checks the previous behavior of node while distribute the load. In cloud, if number of storage nodes, number of files and assesses to that file increases then the central node (master in MapReduce) becomes bottleneck. The load rebalancing task is used to eliminate the load on central node. In load balancing algorithm, storage nodes are structured over network based on the distributed hash table (DHT); each file chunk having rapid key lookup in DHTs, in that unique identifier is assign to each file chunk [1]. DHTs enable nodes to self-recognize and repair while it constantly offers lookup functionality in node. Here aim is to reduce the movement cost which is caused by load rebalancing of nodes to maximize the network bandwidth. Each Chunkserver first find whether it is light node or heavy node without global knowledge of node. The numbers of file chunks are migrated from heavy node to light node to balance their load. This process repeats until all heavy nodes becomes the light nodes. To overcome this load balancing problem each node perform load rebalancing algorithm independently without global knowledge about load of all nodes.

2. Some of the Frameworks

2.1 The Apache Foundation's Hadoop Distributed File System (HDFS) and MapReduce

Engine comprises a distributed computing infrastructure inspired by Google MapReduce and the Google File System (GFS). The Hadoop framework allows processing of massive (bulky) data sets with distributed computing techniques by

Licensed Under Creative Commons Attribution CC BY

leveraging large numbers of physical hosts. However, hadoop's use is spreading far beyond its open source search engine roots. Also the Hadoop frame-work is being offered by "Platform as a Service" cloud computing providers. Hadoop is made up of two primary components. These components are the Hadoop Distributed File System (HDFS) and the MapReduce engine and HDFS is made up of geographically distributed Data Nodes and Access to these Data Nodes is coordinated by a service called the Name Node. Data Nodes communicate over the network in order to rebalance data and en-sure data is replicated throughout the cluster. The MapReduce engine is made up of two main components. Users submit jobs to a Job Tracker which then distributes the task to Task Trackers as physically close to the required data as possible While these are the primary components of a Hadoop cluster there are often other services running in a Hadoop cluster such as a workflow manager.

2.2 Amazon Elastic MapReduce

Amazon Elastic MapReduce (Amazon EMR) is a web service that makes it easy to quickly and cost-effectively process vast amounts of data. Amazon EMR uses Hadoop, an open source framework, to distribute your data and processing across a resizable cluster of Amazon EC2 instances. Amazon EMR is used in a variety of applications, including log analysis, web indexing, data warehousing, machine learning, financial analysis, scientific simulation, and bioinformatics. A customer launches millions of Amazon EMR clusters every year.

3. Literature Survey

Distributed Hash Tables are key building block for variety of distributed applications. It uses the hashing approach and both the keys and peers are hashed onto a 1D ring. After that 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 solution is the use of virtual peers is that for each peers and assigning number of virtual peers. In this case a large size request may not be processed because of the tightly bounded expected value and Substitute solution is that power of two choice paradigms. In this paradigm use standard hashing scenarios using bins to reduce or balanced 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, distributed object location, a scalable, 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 which connected via the Internet. In application level routing, different applications will have different requirements according to the routing is performed. For example video conferencing requires high requirements, if any one of 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 us 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, easy to implement, so its 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 a 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 and Keys are then assigned to the nearest peer in the clockwise direction. Server is connected to their neighbors in the ring and searching for a key reduces to traversing ring and this result is considerable load imbalance. One of the solutions is the use of virtual peers and this is for each peers and assigning number of virtual peers. In this case large size request may not be processed because of the tightly bounded expected value and Second, our range search data structure does not easily generalize into 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 cloud computing [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

Licensed Under Creative Commons Attribution CC BY

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.

Table 1: Comparison table

Section	*Method*	*Advantage*	*Disadvantage*
Pastry: Scalable, distributed object location and routing for large-scale P2P systems	Distributed File System	Decentralized System Automatically performs all the operations	Every time lookup operation is needed
Simple Efficient Load Balancing Algorithms for Peer to Peer Systems	Hashing scheme and Range Search DS	Provide efficient load balancing	Difficult to achieve
Online Balancing of Range-Partitioned Data with Applications to Peer to Peer	Range Partitioning	Decentralized System Automatically performs all operations Avoid Data Skew	Take too much of time to complete the task
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

MapReduce: Simplified Data Processing On Large Clusters [6]

MapReduce is the programming model used in implementation for processing and generating large scaledatasets. It is used at Google for many different purposes. Here map and reduce functions are used. Map function generate set of intermediate key pairs and reduce function merges all intermediate key values associated with same intermediate key. The map and reduce function allows to perform parallelize operation easily and re-execute the mechanism for fault tolerance. At the run-time, system takes care of detail information of partitioning the input data, schedule the program execution across number of available machines, handling failures and managing intercommunication between machines.

In distributed file system nodes simultaneously perform computing and storage operations. The large file in partitioned into number of chunks and allocate it to distinct nodes to perform MapReduce task parallel over nodes.Typically, MapReduce task processes on many terabytes of data on thousands of machines. This model is easy to use; it hides the details of parallelization, optimization, fault-tolerance and load balancing. MapReduce is used for Google's production Web search service, machine learning, data mining, etc. Using this programming model, redundant executionused to reduce the impact of slow machines, handle machine failure as well as data loss.

Load Balancing Algorithm for DHT based structured Peer to Peer System [8]

Peer to peer system have an emerging application in distributed environment. As compared to client-serverarchitecture, peer to peer system improved resource utilization by making use of unused resources over network. Peer to peer system uses Distributed Hash Table (DHTs) as an allocation mechanism. It perform join, leave and update operations. Here load balancing algorithm uses the concept of virtual server to temporary storage of data. Using theheterogeneous indexing, peers balanced their loads proportional to their capacities. In this, decentralized load balance algorithm construct network to manipulate global information and organized in tree shape fashion. Each peer canindependently compute probability distribution capacities of participating peers and reallocate their load in parallel.

4. Conclusion

The proposal strives to balance the loads of nodes/norms and it reduce the demanded movement cost is much as possible while taking advantage of physical network locality and node heterogeneity in cloud. In the absence of representative real workloads such as the distributions of file chunks in a large scale storage system in the public domain that we have investigated the performance of the proposal and compared it against competing algorithms through the synthesized probabilistic distributions of file chunks/parts. Emerging a distributed file systems (DFS) in production systems strongly depend on the central node for chunk the reallocation.However, this dependence is clearly inadequate in a large-scale and failure-prone environment because the central load balancer is put under considerable workload that is linearly scaled with the system size and this may become the performance bottleneck and the single point of failure for node. The algorithm is compared against a centralized approach in a production system and a competing distributed solution is 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.

Licensed Under Creative Commons Attribution CC BY

References

[1] Hung-Chang Hsiao, Member, IEEE Computer Society, Hsueh-Yi Chung, HaiyingShen, Member, IEEE, and Yu-Chang Chao, proposed a "Load Rebalancing for Distributed File Systems in Clouds" IEEE transaction onparallel 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] AartiKhetan, VivekBhushan 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.

[8] ChahitaTanak, Rajesh Bharati "Load Balancing Algorithm for DHT Based Structured Peer to PeerSystem", International Journal of Emerging Technology and Advanced Engineering (ISSN 2250-2459, ISO9001:2008 Certified Journal, Volume 3, Issue 1, January 2013)

Licensed Under Creative Commons Attribution CC BY