ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2014

Dynamic Load Rebalancing by Monitoring the Elastic Map Reduce Service in Cloud

Suriya Mary [1], Vairachilai [2]

P.G. Student, Department of Computer Engineering, NPR Engineering College, NPR Nagar, Tamilnadu, India[1]

Assistant Professor, Department of Computer Engineering, NPR Engineering College, NPR Nagar, Tamilnadu, India[2]

Abstract: CLOUD Computing is a compelling technology. In clouds, Distributed File Systems (DFS) are sharing their resources to process the large amount of typical data. It uses imbalanced chunk relocation process using the centralized approach that causes adequate failure in real time operations. This results in load imbalance in the distributed file system. A fully distributed load rebalancing algorithm is presented to cope with the load imbalance problem. A computational cluster designed specifically for storing and analysing huge amount of unstructured data in the distributed computing environment. Finding the storage nodes capacity from the cluster formation can be able to resize the data nodes and task nodes depending upon the input files. Amazon CloudWatch provides monitoring for AWS cloud resources and its applications. Amazon EMR records metrics that can be used to monitor the cluster. The load balancing progress can be tracked by running the cluster. Amazon CloudWatch automatically read the metrics of the DataNodes and TaskNodes such as CPU Utilization, number of read and write operations performed on the disk. Rebalance the load of the DataNodes based upon the user privilege by using the monitored alerts dynamically. Hence the monitoring solution is reliable, scalable and flexible.

Keywords: Load balance, Distributed File Systems, Clouds.

I. INTRODUCTION

CLOUD Computing is a compelling technology. In clouds, clients can dynamically allocate their resources on-demand without sophisticated deployment and management of resources. Key enabling technologies for clouds include the MapReduce programming paradigm, distributed file systems virtualization and so forth. These techniques emphasize scalability, so clouds can be large in scale, and comprising entities can arbitrarily fail and join while maintaining system reliability. Distributed file systems are key building blocks for cloud computing applications based on the MapReduce programming paradigm. In such file systems, nodes simultaneously serve computing and storage functions; a file is partitioned into a number of chunks allocated in distinct nodes so that MapReduce tasks can be performed in parallel over the nodes.

For example, consider a wordcount application that counts the number of distinct words and the frequency of each unique word in a large file. In such an application, a cloud partitions the file into a large number of disjointed and fixed-size pieces (or file chunks) and assigns them to different cloud storage nodes (i.e., chunkservers). Each storage node (or node for short) then calculates the frequency of each unique word by scanning and parsing its local file chunks. In such a distributed file system, 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. Load balance among storage nodes is a critical function in clouds. In a load-balanced cloud, the resources can be well utilized and provisioned, maximizing the performance of MapReduce-based applications.

The scope of the project is to identify the load rebalancing problem in distributed file systems specialized for large-scale, dynamic and data-intensive clouds. Such a large-scale cloud has hundreds or thousands of nodes (and may reach tens of thousands in the future).The main 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. However, most existing solutions are designed without considering both movement cost and node heterogeneity. In contrast, our proposal not only takes advantage of

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2014

physical network locality in the reallocation of file chunks to reduce the movement cost but also exploits capable nodes to improve the overall system performance. Additionally, the algorithm reduces algorithmic overhead.

We can able to achieve the Load rebalancing for distributed file systems by using one of the Amazon web services. Amazon Web Services (AWS) is a collection of remote computing services (also called web services) that together make up a cloud computing platform, offered over the Internet by Amazon.com. The most central and well-known of these services are Amazon EC2 and Amazon S3. The service is advertised as providing a large computing capacity (potentially many servers) much faster and cheaper than building a physical server farm.AWS is located in 9 geographical regions.

Some of the Amazon Web Services that are related to this project is Amazon Simple Storage Service and Amazon Elastic MapReduce Service. Amazon S3 (Simple Storage Service) is an online file storage web service offered by Amazon Web Services. Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, secure, fast, inexpensive infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers. In this service we can store unlimited number of objects. Each object is stored in a bucket with unique key that are assigned.

Another one service called Amazon Elastic MapReduce Service that can be used to process our input file. 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 our input 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. By using these services, we can easily attain the load imbalance problem. The load balancing progress can be tracked by running the cluster. Amazon CloudWatch automatically monitors the load of the DataNodes based upon the user privilege by using the monitored alerts dynamically. Hence the monitoring solution is reliable, scalable and flexible.

The rest of this paper is organized as follows. In the next section, we presented some of the related work. In section III, we proposed our solution for Cluster Formation. Section IV presents implementation Steps. Finally, section V presents Conclusion.

II. RELATED WORK

The Apache Foundation's Hadoop Distributed File System (HDFS) and MapReduce engine comprise a distributed computing infrastructure inspired by Google MapReduce and the Google File System (GFS). The Hadoop framework allows processing of massive data sets with distributed computing techniques by leveraging large numbers of physical hosts. Hadoop's use is spreading far beyond its open source search engine roots. The Hadoop frame-work is also 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. HDFS is made up of geographically distributed Data Nodes. 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 and so on.

In a distributed file systems, nodes simultaneously serve computing and storage functions. A file is partitioned into a number of chunks allocated in distinct nodes so that MapReduce tasks can be performed in parallel over the nodes. However in a cloud computing environment, failure is the norm. The nodes may be upgraded, replaced, and added in the system. Files can also be dynamically created, deleted, and appended. This results in load imbalance in a distributed file system. So the file chunks are not distributed as uniformly as possible among the nodes. Distributed file systems 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

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2014

system. When the number of storage nodes, the number of files and the number of accesses to files increase linearly, the central nodes become a performance bottleneck. This results in load imbalance in a distributed file system. So the file chunks are not distributed as uniformly as possible among the nodes. They are unable to accommodate a large number of file accesses due to clients and MapReduce applications. Existing systems use the concept of virtual server. The solutions are designed without considering both movement cost and node heterogeneity.

III. CLUSTER FORMATION

The Figure 1 shows the overall proposed system design. The chunkservers are organized as a DHT network, that is, each storage nodes implements a DHT protocol such as Chord or Pastry. A file in the system is partitioned into a number of fixed-size chunks, and "each" chunk has a unique chunk handle (or chunk identifier) named with a globally known hash function such as SHA1. The hash function returns a unique identifier for a given file's pathname string and a chunk index.

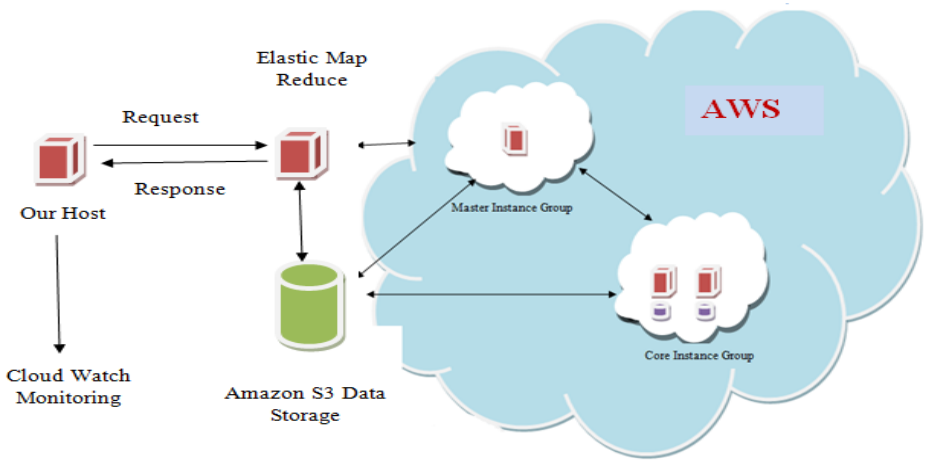

Fig-1 System Architecture

The chunkservers self-configure and self-heal in our proposal because of their arrivals, departures, and failures, simplifying the system provisioning and management. Specifically, 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. The proposal heavily depends on the node arrival and departure operations to migrate file chunks among nodes.

The implementation is done through a small-scale cluster environment consisting of a single, dedicated namenode and datanodes. The cluster environment formation is used to find the storage nodes capacity. The DataNode is responsible for storing the files. It manages the file blocks within the node. It sends information to the NameNode about the files and blocks stored in that node and responds to the NameNode for all file system operations. To scaling the number of data nodes depending upon the size of files and also scaling the task node based upon the job. Amazon EMR records metrics that can be used to monitor the cluster. The load balancing progress can be tracked by running the cluster. Amazon CloudWatch automatically monitors Elastic Load Balancers for metrics such as request count and latency. It also read the metrics of the DataNodes and TaskNodes such as CPU Utilization, number of read and write operations performed on the disk. Rebalance the load of the DataNodes based upon the user privilege by using the monitored alerts dynamically. Hence the monitoring solution is reliable, scalable and flexible.

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2014

IV. IMPLEMENTATION RESULTS

CLUSTER FORMATION

A Hadoop cluster is a special type of computational cluster designed specifically for storing and analyzing huge amount of unstructured data in a distributed computing environment. Typically one machine in the cluster is designated as the NameNode and another machine as the JobTracker; these are the masters. The rest of the machines in the cluster act as both DataNode and TaskTracker; these are the slaves. When a file is placed in HDFS it is broken down into blocks. These blocks are then replicated across the different nodes (DataNodes) in the cluster. Whenever a file is placed in the cluster a corresponding entry of it location is maintained by the NameNode.

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. The 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.

LOAD REBALANCING ALGORITHM

By using this algorithm, we can estimate each chunk server node, whether it is under loaded (light) or overloaded (heavy) without global knowledge.A node is light if the number of modules it hosts is smaller than the threshold as well as, a heavy node manages the number of modules greater than threshold. A large-scale distributed file system is in a load-balanced state if each module server hosts no more than modules. This process repeats until all the heavy nodes in the system become light nodes.

The storage nodes are structured as a network based on distributed hash tables (DHTs), e.g., discovering a file chunk can simply refer to rapid key lookup in DHTs, given that a unique handle (or identifier) 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 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.

DATA NODE & TASK NODE REBALANCING

The DataNode is responsible for storing the files. It manages the file blocks within the node. It sends information to the NameNode about the files and blocks, stored in that node and responds to the NameNode for all file system operations. DataNode can read or write a particular file. Upon initialization, each of the DataNodes informs the NameNode of the blocks it is currently storing. After this mapping is complete, the DataNodes continually poll the NameNode to provide information regarding local changes as well as receive instructions to create, move, or delete blocks from the local disk. DataNode may communicate with other DataNodes to replicate its data blocks for redundancy. To scaling the number of data nodes depending upon the data size of files.

A TaskTracker is a node in the cluster that accepts tasks Map, Reduce and Shuffle operations from a JobTracker. Every TaskTracker is configured with a set of slots, these indicate the number of tasks that it can accept. When the JobTracker tries to find somewhere to schedule a task within the MapReduce operations, it first looks for an empty slot on the same server that hosts the DataNode containing the data, and if not, it looks for an empty slot on a machine in the same rack. TaskTracker is a daemon that accepts tasks (Map, Reduce and Shuffle) from the JobTracker.

The TaskTracker keeps sending a heart beat message to the JobTracker to notify that it is alive. Along with the heartbeat it also sends the free slots available within it to process tasks. TaskTracker starts and monitors the Map & Reduce Tasks and sends progress/status information back to the JobTracker. To scaling the task node based upon the job.

CLOUDWATCH MONITORING

Amazon CloudWatch is an easy to use web service that provides visibility into our cloud assets. It is designed to provide comprehensive monitoring for all of the AWS services. Amazon CloudWatch provides monitoring for AWS cloud resources and the applications customers run on AWS. With Amazon CloudWatch, we gain system-wide visibility into resource utilization, application performance, and operational health. Amazon CloudWatch service provides a reliable, scalable, and flexible monitoring. By using this CloudWatch monitoring, we can able to retrieve our

ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 5, May 2014

monitoring data, view graphs, and set alarms to help us and take automated action based on the state of our cloud environment.

When running a cluster, we can able to track its progress. Amazon EMR records metrics that can be used to monitor the cluster. The load balancing progress can be tracked by running the cluster. Amazon CloudWatch automatically monitors Elastic Load Balancers for metrics such as request count and latency. Read the metrics of the DataNodes and TaskNodes such as CPU Utilization and Number of read and write operations performed on the disk. Rebalance the load of the DataNodes based upon the user privilege by using the monitored alerts dynamically.

V. CONCLUSION

An efficient load rebalancing algorithm to deal with the load imbalance problem in large-scale, dynamic and distributed file systems in clouds has been presented. The implementation is demonstrated through a small-scale cluster environment consisting of a single, dedicated namenode and datanodes. The proposal strives to balance the loads of data nodes and task nodes efficiently. Then only can able to distribute the file chunks as uniformly as possible. The proposed algorithm operates in a distributed manner in which nodes perform their load-balancing tasks independently without synchronization or global knowledge regarding the system. In a load-balanced cloud, the resources can be well utilized and provisioned, maximizing the performance of MapReduce-based applications. The load balancing progress can be tracked by running the cluster. Amazon CloudWatch automatically monitors the load of the DataNodes based upon the user privilege by using the monitored alerts dynamically. Hence the monitoring solution is reliable, scalable and flexible. In future we can able to analyse the cluster status by using this monitoring solution. If a metric goes outside parameters we can able to set alarms. These metrics are automatically collected and pushed to CloudWatch for every Amazon EMR cluster.

REFERENCES

[1] Hung-Chang Hsiao, Member, IEEE Computer Society, Hsueh-Yi Chung,Haiying Shen, Member, IEEE, and Yu-Chang Chao, proposed a "Load Rebalancing for Distributed File Systems in Clouds" IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 5, MAY 2013

[2] J.Dean and S. Ghemawat,proposed a "MapReduce: Simplified Data Processingon Large Clusters," in Proc. 6th Symp. Operating System Design and Implementation (OSDI'04), Dec. 2004, pp. 137–150.

[3] 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.

[4] Y. Zhu and Y. Hu, proposed a "Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems," IEEE Trans. Parallel and Distributed Systems, vol. 16, no. 4, pp. 349-361, Apr. 2005.

[5] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan,proposed a "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Trans. Networking, vol. 11, no. 1, pp. 17-21, Feb. 2003.

[6] Q.H. Vu, B.C. Ooi, M. Rinard, and K.-L. Tan, proposed a "Histogram-Based Global Load Balancing in Structured Peer-to-Peer Systems," IEEE Trans. Knowledge Data Eng., vol. 21, no. 4, pp. 595-608, Apr. 2009.

[7] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Structured P2P Systems," Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS '02), pp. 68-79, Feb. 2003.

[8] 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.