Contemporary Engineering Sciences, Vol. 8, 2015, no. 10, 447 - 451
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2015.5263

Load Re-Balancing for Distributed File System with Replication Strategies in Cloud

R. Surendran

Information System and Technology Department, Sur University College
P.O. Box 440, Postal Code 411, Sur, Sultanate of Oman.

Abstract

File storage load can be balanced in the storage nodes avail in the cloud system by using totally distributed load rebalancing algorithm. Large level distributed systems such as cloud applications come with rising challenges on how to transfer and where to store compute data. Cloud computing is a distributed computing over a network. Node concurrently serves as a computing and storage task. In cloud computing environment, files can also be dynamically created, deleted and append. The file chunks are not distributed as equally among the nodes. Lead to load inequity in a distributed file system. The existing distributed file system depends on single node to manage almost all operations such as chunk reallocation of every data block in the file system. As a result it can be bottleneck resource and a single point of failure. A new technique Random Linear Network Coding (RLNC) is employed in the proposed system. RLNC is performed at the opening when the file is stored in the cloud. Using this strategy, a file will be split into different parts and send distinct parts to each chunk server. RSA algorithm applied to calculate the response time file size and deadlock detection. RSA algorithm used to detect the anomalies of the system. Dynamic scheduling algorithm present in the proposed system to overcome to load inequity problem. This algorithm is compared against a centralized approach in a production system. The results indicate that our proposal is considerably outperforms the prior distributed in terms of load inequity factor, movement cost, and algorithmic overhead.

Keywords: Cloud computing, Load balancing, Distributed hash table

1 Introduction

In cloud computing environment distributed file systems are key building blocks based on map reduce programming paradigm. The performances of cloud computing depend on several factor, one of the factor is load rebalancing. The main objective of the load rebalancing is to reallocate the file chunks such that the chunks can be distributed to the system as uniformly as possible to reduce the movement cost as much as possible. In this work employing Random linear network coding technique which ensures that at any time, the requested data available at one or multiple peers collectively. Using this strategy, a file will be split into different parts and send distinct parts to each chunk server. Then a copy of each part will be send to remaining peer randomly. RLNC is performed at the beginning when the file is stored in the cloud. Load rebalancing algorithm designed for large scale distributed file system consisting of a set of chunk server in cloud. In cloud computing Load rebalancing mechanism spread the excess dynamic workload evenly across the entire server in cloud.

2 Literature Survey

In this work [3], message oriented model to balance the load in cloud computing using middleware technologies. It also discussed the static and dynamic algorithm for load balancing in cloud computing. In this work [4] discussed three load rebalancing algorithms used in the cloud which are round robin algorithm, equally spread current execution load algorithm and Throttled load balancing algorithm and it compared to request time and cost of the three algorithms. In this work [5], Enhanced equally distributed load balancing algorithm. In this algorithm reduces the response time of the request. In the existing system like State-of-the-art distributed file system in cloud relay on central node to manage the Meta information of the file system and to balance the loads of the storage nodes based on the Meta data. The centralized approach simplifies the design and implementation of the distributed file system. In the existing system, if the number of storage nodes, the number of files and the number of access to the file increase linearly, the central node or master node becomes a performance bottleneck, as they are unable to accommodate a large number of file accesses due to the client and map reduce application. In the existing system load rebalance among the storage node is critical.

3 Proposed Work

In this system employing the Random linear network coding (RLNC) technique which ensures that at any time, the requested data are available at one or multiple chunk server collectively. Using this strategy file can be split into multiple parts and sent to multiple chunk servers. Then a copy of each part will be send to the remaining peer randomly. The offloading process of load rebalancing task performed on storage nodes. The storage node can balance the node spontaneously.

The storage node structured as a network based distributed hash tables (DHTs). A unique handler is assigned to each file chunk which is loaded into DHT which enable nodes to self organize and repair while constantly offering lookup functionality in the node dynamism, simplifying the system provision and management.

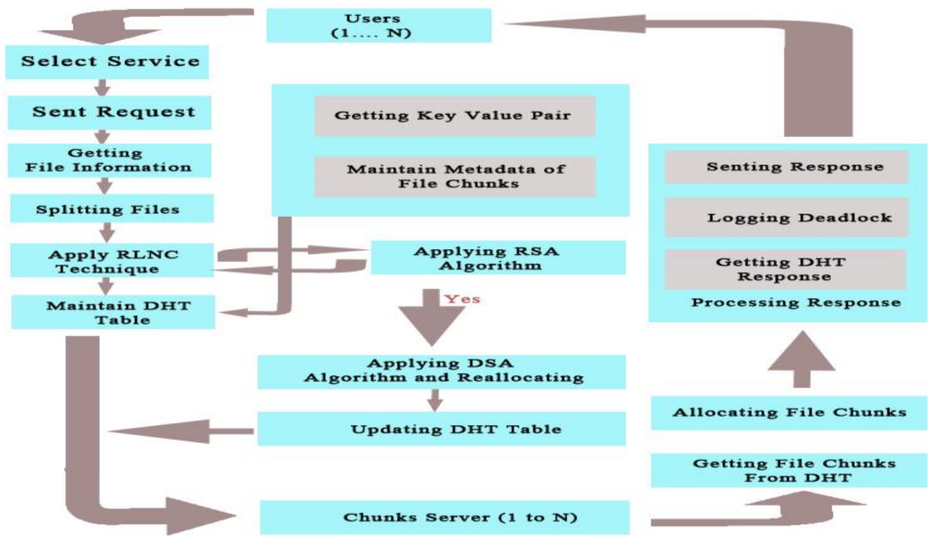

Figure 1. Architecture Diagram

The chunk server in our proposal is organized as a DHT network that is each chunk server implements a DHT protocol. The ideal number of chunks for each storage node is defined, according to the capacity of chunk server. Successor and predecessor of each node is defined. In the cloud network design, the initial node must be the successor of the final node.

In cloud client can dynamically allocate the resources on demand without sophisticated deployment and management of resources. Lookup process using DHT table is carried to reallocate the chunk file. In practice one can view the distributed storage system based on RLNC strategy, as letting individual peers to perform a single task in random manner. However after some time, the data might no longer be recoverable. RLNC based strategy preserve data in the network exponentially longer; it is the least efficient of the three in terms of replenishment bandwidth. This is because amount of download data is more than a peer’s storage capacity.
DHT table is generated to maintain the storage node information. It is maintained at the server side of the system. The chunk file information’s can be updated by the individual storage server in dynamic manner. DHT is used to perform a lookup service similar to hash table. Key pairs are always stored in DHT in order to get the corresponding values for the concerned key pairs. All the keys are collecti-

vely distributed over all the nodes which help to handle large amount of data upon success or failure. To discover file chunk, DHT lookup operation is performed.
A file in the system is partitioned into number of fixed-size chunks, and each chunk has a unique identifier named with globally known hash function. The hash function returns a unique identifier for a given file's path name string and chunk index. Partitioned file chunks are distributed among the distinct storage node. The metadata of each file chunk is maintained in the DHT table. To discover a file chunk, the DHT lookup operation is performed. The metadata of file gives information about the location of file creation, time and data of creation and the purpose of data.
Using the RSA algorithm to calculate response time, file size and deadlock detection. When the file is uploaded by two users from two different situations the server may become slow when compare accordingly with the amount of data they are storing. When two data uses the same node resources and if the data is large the deadlock situation will occur. In order to avoid deadlock, maintain a constant speed of upload or download.
After applying the RSA algorithm file chunks were moved from one node to another node modifying the nodes in cloud network. While the chunk server experiencing the overload, it will look up the DHT table for requesting reallocation process. The node which has least load will be chosen by the overloaded chunk server. The least loaded chunk server moves all loads to its successor chunk server. The distributed hash table was updated to after the applying of RSA algorithm. Scheduling algorithm is applied on chunk server while it experiencing the overload. Storage load of each chunk server can be balanced by reallocating the chunk files. In this process, every storage node is limited for storing chunk files according to the capacity of the node. While the chunk server crosses it storage limit for storing the file chunks, the load rebalance process will be carried out.

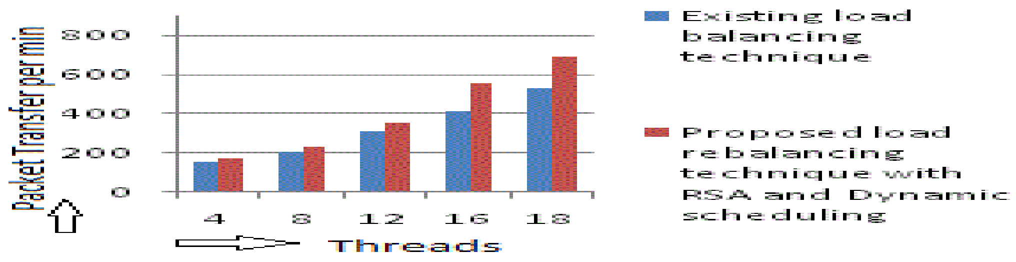

Figure 2. Performance Analysis

In this proposed system the copy of chunk file is stored in the different chunk server. When the node is failed to process the chunk files, a new node is replaced and the file chunks are reallocated by using copy of chunk file is stored in another chunk servers. The reallocation is carried out by using DHT table which holds the metadata of file chunks stored in the cloud network. A new node can also added in the network using updating information in the DHT table and then name is assigned to the node according to the predecessor node. In the Figure 2 graph we compared the time take to carry out one single request. In the existing system the time taken to complete the response is higher when compared with the proposed system.

4. Conclusion

The proposed algorithm exhibits fast coverage rate and reduce the movement cost. In this system also reduces the network traffic caused by the nodes in cloud. RLNC is employed to split file and to manage the replicas which are distributed distinctly over the chunk server in the network. The proposed algorithm is operates in distributed manner in which node performs their load-rebalancing tasks with global knowledge regarding the system.

Acknowledgements. I thank Sur University College for providing us with various resources and an unconditional support for carrying out this work.

References

[1] Hung-Chang Hsiao, Hsueh-Yi Chang, Haiying-shen, and Yu-chang Cho, Load rebalancing for distributed file system in cloud, *IEEE transaction on parallel and distributed file systems*, 25 (2013).
http://dx.doi.org/10.1109/tpds.2012.196

[2] Zenon Chaczko, Snahrzad Aslanzadesh and Christopher, Venkatesh Mahadevan, Availability and load balancing in cloud computing, *International conference on computer and software modeling*,14 (2011).

[3] Hemant S. Mahalle, Parag R. Kaveri, Vinay Chavan, Load balancing on cloud data centres, *International Journal of advanced research in computer science and software engineering*, 3 (2013).

[4] Doaa M. Abdelkader, Fatma Omara, Dynamic task scheduling algorithm with load balancing for heterogeneous computing system, *Egyptian informatics journal*, 13(2012),135-145. http://dx.doi.org/10.1016/j.eij.2012.04.001

[5] K. McKusick and S. Quinlan, GFS: Evolution on Fast-Forward, *Comm.ACM*, 53 (2010), 42-49. http://dx.doi.org/10.1145/1666420.1666439

[6] Tamilvizhi T., Parvatha Varthini B., Vinothini P., An Innovative Mechanism for Proactive Fault Tolerance in Cloud Computing, *International Conference on Mathematical Sciences*, *Elsevier* (2014), 618-622.

Received: March 7, 2015; Published: April 6, 2015