File Chunks Balancing For DFSs by Load Rebalancing Algorithm

[1]Syamili Mohandas, [2]Nakulraj K.R
M.TechStudentComputer Science & Engineering, VAST,Thrissur,India
[1]syamili.das@gmail.com, [2]nakulraj45@gmail.com

***Abstract*—Distributed File Systems are file systems that allow access to files from multiple hosts via a computer network, making it possible for multiple users on multiple machines to share files and storage resources. These are the key building blocks for cloud computing applications based on the Map Reduce programming paradigm. In a cloud computing environment, failure is the norm, and chunk servers may be upgraded, replaced, and added in the system. In addition, files can also be dynamically created, deleted and appended. And that lead to load imbalance in a distributed file system. It means that file chunks are not distributed equitably between the nodes. Emerging distributed file systems in production systems strongly depend on a central node for chunk reallocation. However, this centralized approach can provoke a bottleneck for those central nodes as they become unable to manage a large number of file accesses. Consequently, dealing with the load imbalance problem with the central nodes complicate more the situation as it increases their heavy loads. A fully distributed load rebalancing algorithm is presented here to cope with the load imbalance problem. Additionally, aspires 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. For the security purpose the partitioned file chunks are stored in an encrypted format and also the main server can view the details of who download which file at which time.**

***IndexTerms*—Distributed File System (DFS),Map Reduce programming paradigm, Cloud.**

I. INTRODUCTION

Cloud computing is the delivery of computing services over the internet.The cloudcomputing model allows access to information and computer resources from anywherethat a network connection is available.It provides a shared pool of resources,including data storage space, networks, computer processing power, andspecialized corporate and user applications. Key enabling technologies for cloudsinclude the MapReduce programming paradigm , distributed file systems, virtualization,and so forth. These techniques emphasize scalability, so clouds can belarge in scale, and comprising entities can arbitrarily fail and join while maintainingsystem reliability.

In the cloud storage era, the distributed file system concept extended to cloud storage.With cloud storage, it is possible to unify multiple file servers from multiplesites into one single namespace.It makes easier to distribute documents to multipleclients and provide a centralized storage system so that client machines are notusing their resources to store files.In a distributed file system, the load of a nodeis typically proportional to the number of file chunks the node possesses. Becausethe files in a cloud can be arbitrarily created, deleted, and appended, and nodescan be upgraded, replaced and added in the file system, the file chunks are not distributedas uniformly as possible among the nodes.Load balancing is essential forefficient operations in distributed environments. It means distributing the amountof work to do between different servers in order to get more work done in thesame amount of time and clients get served faster. In a load balanced cloud, theresources can be well used while maximizing the performance of MapReducebasedapplications.

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. Additionally, 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. And also central server makes able to see the details of file download by the clients.Moreover, as failure is the norm, nodes are newly addedto sustain the overall system performance, resulting in the heterogeneity of nodes.Exploiting capable nodes to improve the system performance is, thus, demanded.

Network File System, Frangipani etc are examples for existing DFSs.Network FileSystem is a popular file system for the networked computers by which sharingfiles between machines on a network as if the files were located on the clientslocal hard drive. Frangipani is another one that manages a collection of disks onmultiple machines as a single shared pool of storage. In these system requires acommon administrator and depends on a single name node to manage almost alloperations of every data block in the file system. As a result, it can be a bottleneckresource and a single point of failure. The proposed system mainly avoidsthis dependence on central nodes by partitioning the files in the central node intodifferent chunks of fixed size and loads these into the available chunk servers asuniformly as possible. Also make sure that no any node manages excess numberof file chunks.

II. ARCHITECHURE

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. Proposed system presenting a fully distributed load rebalancing algorithm to cope with the load imbalance problem. Additionally, aims 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.

The storage nodes in the proposed system are structured as a network based on distributed hash tables. Distributed hash tables are a building block used to locate key-based objects over millions of hosts on the internet. Each node maintaining a DHT.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. To make the loads of each storage node as uniform as possible, each file which are kept in the central server is splitted into several parts/chunks of fixed- size. The load of each chunk server is proportional to the number of chunks hosted by the server. Whenever a particular chunk server fails which result in the lost of it's corresponding file chunks. At that time ,the central server get the notification about the port number of the corresponding failed chunk server ,then perform the rebalancing of the loads of remaining chunk servers and again upload the lost file uniformly to the remaining available chunk servers.

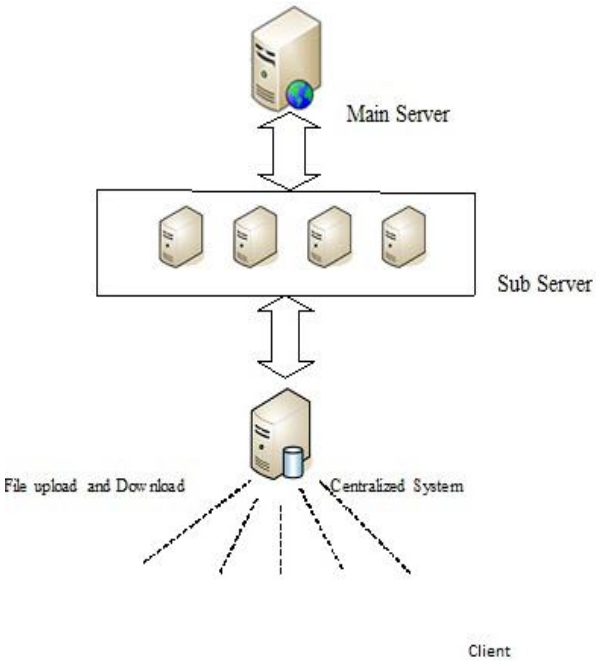

Fig1: Architecture

The main advantages of the proposed system are: The load of each virtual server is stable over the timescale when load balancing is performed; Reduce the network traffic, The load rebalancing algorithm exhibits a fast convergence rate.
The load rebalancing problem in distributed file systems specialized for largescale,dynamic and data-intensive clouds. Such a large-scale cloud has hundreds or thousands of nodes .Load balancing of these storage nodes are performed by performing the functions which are mentioned in the following modules. Thereare four completely different modules for the successful implementationof theproposed system.

A).CHUNK CREATION

A file is partitioned into a number of chunks allocated in distinct nodes so thatMap Reduce tasks can be performed in parallel over the nodes. The load of anode 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, andnodes can be upgraded, replaced and added in the file system, the file chunksare not distributed as uniformly as possible among the nodes.The objective is toallocate the chunks of files as uniformly as possible among the nodes such thatno node manages an excessive number of chunks.Each file will be divided by thetotal number of available chunk servers.

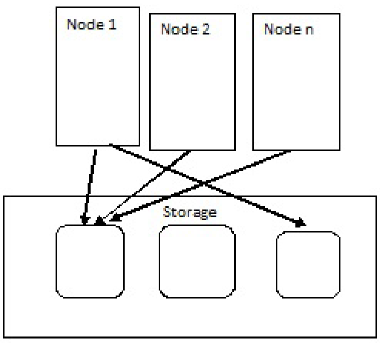

Fig 2: Chunk Allocation

B).DHT FORMULATION

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 inDHTs, given that a unique handle (or identifier) is assigned to each file chunk.DHTs enable nodes to self-organize and repair while constantly offering lookupfunctionality in node dynamism, simplifying the system provision and management.The chunk servers in this proposal are organized as a DHT network.

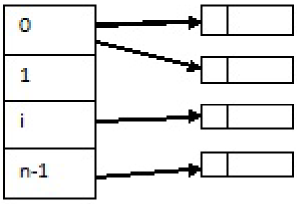

Fig 3: DHT Formulation

C).LOAD BALANCING

In the proposed algorithm, each chunk server node I first estimate whether it isunder loaded (light) or overloaded (heavy) without global knowledge. A node islight if the number of chunks it hosts is smaller than the threshold. Load statuses ofa sample of randomly selected nodes. Specifically, each node contacts a numberof randomly selected nodes in the system and builds a vector denoted by V. Avector consists of entries, and each entry contains the ID, network address andload status of a randomly selected node. Before loading a file chunk into a chunkserver it compares it's memory capacity to avoid load imbalance. If it exceeds then load the file chunk to the next appropriate chunk server.

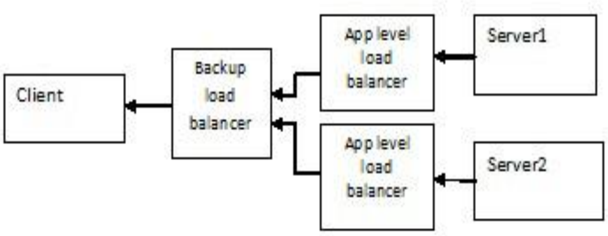

Fig 4:Load Balancing

D). REPLICA MANAGEMENT

In distributed file systems, a constant number of replicas for each file chunk aremaintained in distinct nodes to improve file availability with respect to node failuresand departures. Current load balancing algorithm does not treat replicas distinctly.
It is unlikely that two or more replicas are placed in an identical nodebecause of the random nature of the proposed load rebalancing algorithm. Morespecifically, each under loaded node samples a number of nodes, each selectedwith a probability of 1/n, to share their loads (where n is the total number of storagenodes).

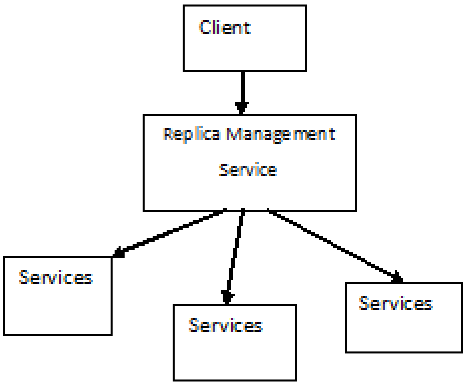

Fig 5: Replica Management

III. CONCLUSION

A load-balancing algorithm to make the load of storage nodes in the large-scale, dynamic, and distributed file systems in clouds as uniform as possible. Load balancingis performed by partitioning the files in the central sever into differentchunks of fixed size and loading these into the chunk servers. Also ensure that noany node manages excess number of file chunks. However, in a cloud computingenvironment, files can also be dynamically created, deleted, and appended. Theproposed system eliminates the dependence on central nodes. The storage nodesare structured as a network based on distributed hash tables. DHTs enable nodes to self-organize and repair while constantly offering lookup functionality in nodedynamism, simplifying the system provision and management. For the securitypurpose the partitioned file chunks are stored in an encrypted format and also themain server can view the details of who download which file at which time.

REFERENCES

[1] A.Hung-Chang Hsiao, Hsueh-Yi Chung, HaiyingShen,and Yu-Chang Chao:".,Load Rebalancing for Distributed File Systems in Clouds",Proceedings of IEEE TRANSACTIONS ON PARALLEL ANDDISTRIBUTED SYSTEMS VOL. 24, NO. 5, MAY 2013 .

[2] B.J. Dean and S. Ghemawat, MapReduce:" Simplified Data Processing onLarge Clusters",Proc. Sixth Symp. Operating System Design and Implementation(OSDI 04), pp. 137-150, Dec. 2004.

[3] C. G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A.Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: AmazonsHighly Available Key-Value Store", Proc. 21st ACM Symp. OperatingSystems Principles (SOSP 07), pp. 205-220, Oct. 2007.

[4] D.A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica," Load Balancingin Structured P2P Systems",Proc. Second Intl Workshop Peer-to-PeerSystems (IPTPS 02), pp. 68-79, Feb. 2003.

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

[6] F.J.W. Byers, J. Considine, and M. Mitzenmacher, "Simple Load Balancingfor Distributed Hash Tables",Proc. First Intl Workshop Peer-to-Peer Systems(IPTPS 03), pp. 80-87, Feb. 2003.

[7] G. G.S. Manku, "Balanced Binary Trees for ID Management and Load Balancein Distributed Hash Tables", Proc. 23rd ACM Symp. Principles DistributedComputing (PODC 04), pp. 197-205, July 2004.

[8] H. Y. Zhu and Y. Hu, Efficient, "Proximity-Aware Load Balancing for DHTBasedP2P Systems",IEEE Trans. Parallel and Distributed Systems, vol. 16,no. 4, pp. 349-361, Apr. 2005

[9] H.-C. Hsiao, H. Liao, S.-S.Chen, and K.-C. Huang, "Load Balance with ImperfectInformation in Structured Peer-to-Peer Systems",IEEE Trans. ParallelDistributed Systems, vol. 22, no. 4, pp. 634-649, Apr. 2011.

[10] J. S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica, "LoadBalancing in Dynamic Structured P2P Systems,Performance Evaluation",vol. 63, no. 6, pp. 217-240, Mar. 2006.

[11] K. H. Feelifl, M. Kitsuregawa, and B. C. Ooi. A fast convergence technique foronline heat-balancing of btree indexed database over shared-nothing parallelsystems.In Proc. DEXA, 2000.

[12] L.J. Dean and S. Ghemawat, MapReduce: Simplified processing on giant Clusters,in Proc. 6th Symp. software system style and Implementation (OSDI04),Dec. 2004, pp. 137150.

[13] M. A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, Load equalizationin Structured P2P Systems, inProc. ordinal Intl Workshop Peerto-Peer Systems (IPTPS02), Feb. 2003, pp. 6879.

[14] N.Karger and M. Ruhl, Simple economical Load equalization Algorithms forPeer-to-Peer Systems, in Proc. sixteenth ACM Symp. Parallel Algorithmsand Architectures (SPAA04), June 2004, pp. 3643.