ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Monthly Peer Reviewed Open Access International e-Journal

Load Performing in Servers by Using Main Server

Bommakanti Nithesh
M.Tech Student,
Department of Computer Science Engineering,
Institute of Aeronautical Engineering.

Dr. N.Chandrashekar Reddy
Professor & HoD,
Department of Computer Science Engineering,
Institute of Aeronautical Engineering.

Abstract:

Cloud computing is typically defined as a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. In cloud computing, the word cloud (also phrased as "the cloud") is used as a metaphor for "the Internet," so the phrase cloud computing means "a type of Internet-based computing," where different services — such as servers, storage and applications — are delivered to an organization's computers and devices through the Internet. Distributed file systems do not share block level access to the same storage but use a network protocol.

These are commonly known as network file systems, even though they are not the only file systems that use the network to send data.[citation needed] Distributed file systems can restrict access to the file system depending on access lists or capabilities on both the servers and the clients, depending on how the protocol is designed. In this paper, a fully distributed load rebalancing algorithm is studied and presented to cope with the load imbalance problem.

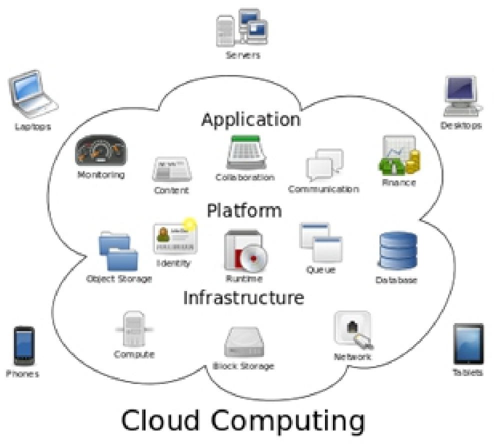
This method is compared against a centralized approach in a production system and a competing distributed solution presented in the literature. The simulation results indicate that our proposal is comparable with the existing centralized approach and considerably outperforms the prior distributed algorithm in terms of load imbalance factor, movement cost, and algorithmic overhead.

Keywords:

Reliability, security, Load balancing, Main controller, Balancers, Servers, energy conservation, DHT, Centralized System, Load Imbalancing.

Introduction:

In a cloud computing system, there's a significant workload shift. Local computers no longer have to do all the heavy lifting when it comes to running applications. The network of computers that make up the cloud handles them instead. Hardware and software demands on the user's side decrease. The only thing the user's computer needs to be able to run is the cloud computing system's interface software, which can be as simple as a Web browser, and the cloud's network takes care of the rest.



The National Institute of Standards and Technology's definition of cloud computing identifies "five essential characteristics":On-demand self-service: A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service provider.Broad network access: Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, tablets, laptops, and workstations). Resource pooling: The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. Rapid elasticity: Capabilities can be elastically provisioned and released, in some cases automatically, to scale rapidly outward and inward commensurate with demand.

To the consumer, the capabilities available for provisioning often appear unlimited and can be appropriated in any quantity at any time.Measured service: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts).

Resource usage can be monitored, controlled, and reported, providing transparency for both the provider and consumer of the utilized service.

Distributed file system:

The difference between a distributed file system and a distributed data store is that a distributed file system allows files to be accessed using the same interfaces and semantics as local files - e.g. mounting/unmounting, listing directories, read/write at byte boundaries, system's native permission model. Distributed data stores, by contrast, require using a different API or library and have different semantics (most often those of a database).

Design goals of DFS:

Distributed file systems may aim for "transparency" in a number of aspects. That is, they aim to be "invisible" to client programs, which "see" a system which is similar to a local file system. Behind the scenes, the distributed file system handles locating files, transporting data, and potentially providing other features listed below.

• Access transparency is that clients are unaware that files are distributed and can access them in the same way as local files are accessed.

• Location transparency; a consistent name space exists encompassing local as well as remote files. The name of a file does not give its location.

• Concurrency transparency; all clients have the same view of the state of the file system. This means that if one process is modifying a file, any other processes on the same system or remote systems that are accessing the files will see the modifications in a coherent manner.

• Failure transparency; the client and client programs should operate correctly after a server failure.

• Heterogeneity; file service should be provided across different hardware and operating system platforms.

• Scalability; the file system should work well in small environments (1 machine, a dozen machines) and also scale gracefully to huge ones (hundreds through tens of thousands of systems).

• Replication transparency; to support scalability, we may wish to replicate files across multiple servers. Clients should be unaware of this.

• Migration transparency; files should be able to move around without the client's knowledge.

Related Work:

This attempt to load-balance can fail in two ways. First, the typical "random" partition of the address space among nodes is not completely balanced. Some nodes end up with a larger portion of the addresses and thus receive a larger portion of the randomly distributed items. Second, some applications may preclude the randomization of data items' addresses. For example, to support range searching in a database application the items may need to be placed in a specific order, or even at specific addresses, on the ring. In such cases, we may find the items unevenly distributed in address space, meaning that balancing the address space among nodes is not adequate to balance the distribution of items among nodes. We give protocols to solve both of the load balancing challenges just described.

Performance in a P2P System:

Our online load balancing algorithms are motivated by a new application domain for range partitioning peertopeer systems. P2P systems store a relation over a large and dynamic set of nodes, and support queries over this relation.

Many current systems, known as Distributed Hash Tables (DHTs) use hash partitioning to ensure storage balance, and support point queries over the relation. There has been considerable recent interest in developing P2P systems that can support efficient range queries. For example, a P2P multi-player game might query for all objects located in an area in a virtual 2-D space. In a P2P web cache, a node may request (prefetch) all pages with a specific URL prefix. It is well-known that hash partitioning is inefficient for answering such ad hoc range queries, motivating a search for new networks that allow range partitioning while still maintaining the storage balance offered by normal DHTs.

Handling Dynamism in the Network:

The network is a splits the range of Nh to take over half the load of Nh, using the NBRADJUST operation. After this split, there may be NBRBALANCE violations between two pairs of neighbors and In response, ADJUSTLOAD is executed, first at node Nh and then at node N. It is easy to show (as in Lemma 3) that the resulting sequence of NBRADJUST operations repair all NBRBALANCE violations.

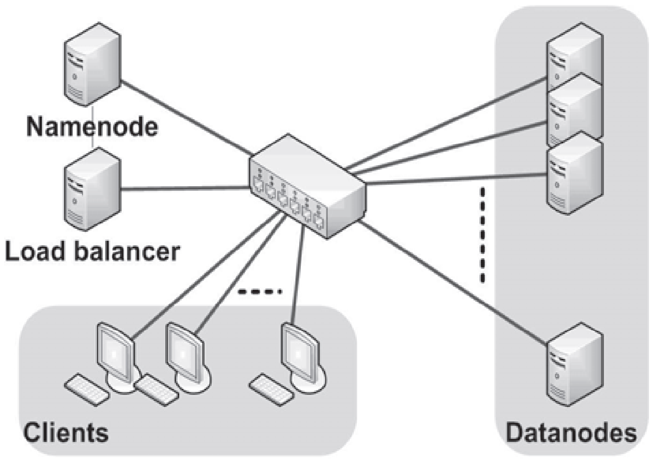
Node Departure:

While in the network, each node manages data for a particular range. When the node departs, the data is stored becomes unavailable to the rest of the peers. P2P networks reconcile this data loss in two ways: (a) Do nothing and let the "owners" of the data deal with its availability. The owners will frequently poll the data to detect its loss and re-insert the data into the network. Maintain replicas of each range across multiple nodes.

A Skip Net DHT organizes peers and data objects according to their lexicographic addresses in the form of a variant of a probabilistic skip list. It supports logarithmic time range-based lookups and guarantees path locality. Mercury is more general than Skip Net since it supports range-based lookups on multiple-attributes. Our use of random sampling to estimate query selectivity constitutes a novel contribution towards implementing scalable multi-dimensional range queries. Load balancing is another important way in which Mercury from Skip Net.

While Skip Net incorporates a constrained load-balancing mechanism, it is only useful when part of a data name is hashed, in which case the part is inaccessible for performing a range query. This implies that Skip Net supports load-balancing or range queries not both.

Architecture:



EXISTING SYSTEM:

However, recent experience concludes that 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, as they are unable to accommodate a large number of file accesses due to clients and Map Reduce applications. Thus, depending on the central nodes to tackle the load imbalance problem exacerbate their heavy loads. Even with the latest development in distributed file systems, the central nodes may still be overloaded.

PROPOSED SYSTEM:

In this paper, we are interested in studying the load rebalancing problem in distributed file systems specialized for large-scale, dynamic and data-intensive clouds. (The terms "rebalance" and "balance" are interchangeable in this paper.) Such a large-scale cloud has hundreds or thousands of nodes (and may reach tens of thousands in the future). Our 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.

Advantages of Proposed System:

Using this we can use in large scale, failure-prone environment because the central load balancer is put under considerable workload that is linearly scaled with the system size. Another advantage of the proposed system is the security consistency provided by it. Various nodes with heavy loads have been proposed as alternatives to central node module so that so many drawbacks of the existing system can be avoided. The proposed system will help in keeping the system consistent so that we can avoid data loss.

Modules :

1.Data Owner Registration.

2.Data User Registration.

3.TTP (TRUSTED THIRD PARTY) LOGIN.

4.CSP(CLOUD SERVICE PROVIDER) LOGIN.

5.Download File

Data Owner Registration:

In this module if a owner of data(File) have to store data on a cloud server,he/she should register their details first.These details are maintained in a Database. Then he has to upload the file in a file database. The file which are stored in a database are in an encrypted form. Authorized users can only decode it.

Data User Registration:

In this module if a user wants to access the data which is stored in a cloud server,he/she should register their details first.These details are maintained in a Database.

TTP (TRUSTED THIRD PARTY) LOGIN:

In this module TTP has monitors the data owners file by verifying the data owner's file and stored the file in a database .Also ttp checks the CSP(CLOUD SERVICE PROVIDER),and find out whether the csp is authorized one or not.

CSP(CLOUD SERVICE PROVIDER) LOGIN:

In this module CSP has to login first.Then only he can store the file in his cloud server.Ttp can only check the csp whether the csp is authorized csp or not.If its fake,ttp wont allow the file to store in cloud server.

Download File:

If the user is an authorized user,he/she can download the file by using meta data of the file which have uploaded and divided.

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] Hsiao, Hung-Chang , Chung, Hsueh-Yi ; Shen, Haiying ; Chao, Yu-Chang, National Cheng Kung University, Tainan , Parallel and Distributed Systems, IEEE Transactions on (Volume:24 , Issue: 5).

[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-toPeer 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-toPeer Systems (IPTPS '03), pp. 80-87, Feb. 2003.

[9] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's Highly Available Key-value Store," in Proc. 21st ACM Symp.

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