Distributed File of Load Rebalancing Systems in Clouds

Mr. G. Jvothi Krishna[1] and Mr. A.V.S. Pavan Kumar[2]

1. *Final Year M.Tech, Department of CSE, Baba Institute of Technology and Sciences, Visakhapatnam, AP, India.*

2. *Assistant Professor Department of CSE, Baba Institute of Technology and Sciences, Visakhapatnam, AP, India*

Abstract: File system that allows many clients to have access to the same data/file providing important operations (create, delete, modify, read, write). Each file may be partitioned into several parts called chunks. Each chunk is stored in remote machines. Typically, data is stored in files in a hierarchical tree where the nodes represent the directories. Hence, it facilitates the parallel execution of applications. There are several ways to share files in a distributed architecture. Each solution must be suitable for a certain type of application relying on how complex is the application or how simple it is. Meanwhile, the security of the system must be ensured. Confidentiality, availability and integrity are the main keys for a secure system. Nowadays, users can share resources from any computer/device, anywhere and everywhere through internet thanks to cloud computing which is typically characterized by the scalable and elastic resources -such as physical servers, applications and any services that are virtualized and allocated dynamically. Thus, synchronization is required to make sure that all devices are update. Distributed file systems enable also many big, medium and small enterprises to store and access their remote data exactly as they do locally, facilitating the use of variable resources. As cloud computing provides a large-scale computing thanks to its ability of providing to the user the needful CPU and storage resources with a complete transparency, it makes it very suitable to different types of applications that require a large-scale distributed processing. That kind of Data-intensive computing needs a high performance file system that can share data between VMs (Virtual machine).

Key Words: Vectors, Cloud computing, Load balance, distributed file systems, clouds

Introduction:

In a cloud computing environment, failure is the norm, and chunk servers may be upgraded, replaced, and added in the system. Files can also be dynamically created, deleted, and appended. That leads to load imbalance in a distributed file system, meaning that the file chunks are not distributed equitably between the nodes. Distributed file systems in clouds such as GFS and HDFS rely on central servers (master for GFS and Name Node for HDFS) to manage the metadata and the load balancing. The master rebalances replicas periodically: data must be moved form a Data Node/ chunk server to another one if its free space is below a certain threshold. However, this centralized approach can provoke a bottleneck for those servers as they become unable to manage a large number of file accesses. Consequently, dealing with the load imbalance problem with the central nodes complicates more the situation as it increases their heavy loads. The load rebalance problem is NP-hard In order to manage large number of chunk servers to work in collaboration, and solve the problem of load balancing in distributed file systems, several approaches have been proposed such as reallocating file chunks such that the chunks can be distributed to the system as uniformly as possible while reducing the movement cost as much as possible. Among the biggest internet companies, Google has created its own distributed file system named Google File System to meet the rapidly growing requests of Google's data processing needs and it is used for all cloud

services. GFS is a scalable distributed file system for data-intensive applications. It provides a fault-tolerant way to store data and offer a high performance to a large number of clients.GFS uses Map Reduce that allows users to create programs and run them on multiple machines without thinking about the parallelization and load-balancing issues . GFS architecture is based on a single master, multiple chunk servers and multiple clients. The master server running on a dedicated node is responsible for coordinating storage resources and managing files's metadata (such as the equivalent of inodes in classical file systems). Each file is split to multiple chunks of 64 Megabyte. Each chunk is stored in a chunk server. A chunk is identified by a chunk handle, which is a globally unique 64-bit number that is assigned by the master when the chunk is first created. As said previously, the master maintain all of the files's metadata including their names, directories and the mapping of files to the list of chunks that contain each file's data. The metadata is kept in the master main memory, along with the mapping of files to chunks. Updates of these data are logged to the disk onto an operation log. This operation log is also replicated onto remote machines. When the log become too large, a checkpoint is made and the main-memory data is stored in a B-tree structure to facilitate the mapped back into main memory

we present a new methodology for managing read-write file sets across multiple file servers of a Distributed File System, thus balancing the load of file access requests across servers. The proposed methodology is based on a rule-based data mining technique and graph theory algorithms. The rule-based technique generates rules from access request data to identify present file access patterns in the system. We then use the rules, graph analysis and statistical information (usage and size of the filesets) to relocate the filesets between different file servers. The algorithm for fileset relocation is based on the graph coloring problem. We tested our algorithms on data collected for five months on DFS file servers in a production environment. Experiments with the data show that our methodology can make intelligent decisions about file system transfers in order to balance the access request load across DFS servers.

Claws in Load Rebalancing:

In the classical load balancing or multiprocessor scheduling problem, we are given a sequence of jobs of varying sizes and are asked to assign each job to one of the m empty processors. A typical objective is to minimize the make span, which is the load on the heaviest loaded processor. Since in most real world scenarios the load is a dynamic measure, the initial assignment may not remain optimal over time. Motivated by such considerations in a variety of systems, we formulate the problem of load rebalancing--given a possibly suboptimal assignment of jobs to processors, relocate a set of the jobs so as to decrease the make span. Specifically, the goal is to achieve the best possible make span under the constraint that no more than k jobs are relocated. We also consider the weighted version of this problem where there is an arbitrary cost associated with each job's relocation. The problem is NP-hard and hence, we focus on approximation algorithms. We construct an algorithm which achieves a 1.5-approximation, with near linear running time. We also show that the problem has a PTAS, thereby resolving the complexity issue. Finally, we investigate the approximability of several extensions of the load rebalancing model.

Mode of Action:

Acceleration is split into three parts. The first is a high speed web cache, providing the possibility to

cache all static content of applications. The second part is compression which can be applied to all content. Both parts are designed to speed up application delivery. Third part is Server Load Balancing. Applications can now spread of several application servers reducing the risk on downtimes and accelerate application response times.

DenyAll's solutions are helping to avoid following issues:

- Webpage downtime
- Long loading times
- To much web traffic

The benefits of Load Balancing solutions are the following:

- Fulfill strong Service Level Agreements
- Speed up Web applications
- Fault tolerant

Technologies used by Deny All products are unique on the market:

- Profile based configuration
- Standard templates available
- Easy and reliable configuration

Technical Overview:

Network Load Balancing, a clustering technology included in the Microsoft Windows 2000 Advanced Server and Datacenter Server operating systems, enhances the scalability and availability of mission-critical, TCP/IP-based services, such as Web, Terminal Services, virtual private networking, and streaming media servers. This component runs within cluster hosts as part of the Windows 2000 operating system and requires no dedicated hardware support. To scale performance, Network Load Balancing distributes IP traffic across multiple cluster hosts. It also ensures high availability by detecting host failures and automatically redistributing traffic to the surviving hosts. Network Load Balancing provides remote controllability and supports rolling upgrades from the Windows NT 4.0 operating system.

The unique and fully distributed architecture of Network Load Balancing enables it to deliver very high performance and failover protection, especially in comparison with dispatcher-based load balancers. This white paper describes the key features of this technology and explores its internal architecture and performance characteristics in detail.

Internet server programs supporting mission-critical applications such as financial transactions, database access, corporate intranets, and other key functions must run 24 hours a day, seven days a week. And networks need the ability to scale performance to handle large volumes of client requests without creating unwanted delays. For these reasons, clustering is of wide interest to the enterprise. Clustering enables a group of independent servers to be managed as a single system for higher availability, easier manageability, and greater scalability.

The Microsoft® Windows® 2000 Advanced Server and Datacenter Server operating systems include two clustering technologies designed for this purpose: Cluster service, which is intended primarily to provide failover support for critical line-of-business applications such as databases, messaging systems, and file/print services; and Network Load Balancing, which serves to balance

incoming IP traffic among multi-node clusters. We will treat this latter technology in detail here.

Network Load Balancing provides scalability and high availability to enterprise-wide TCP/IP services, such as Web, Terminal Services, proxy, Virtual Private Networking (VPN), and streaming media services. Network Load Balancing brings special value to enterprises deploying TCP/IP services, such as e-commerce applications, that link clients with transaction applications and back-end databases.

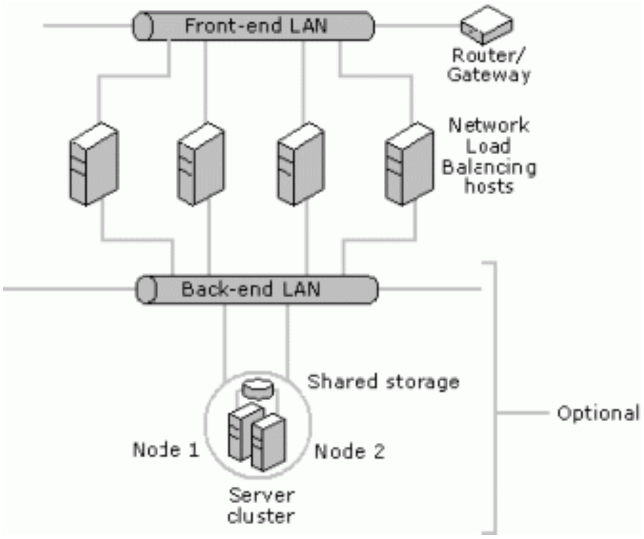
Network Load Balancing servers (also called hosts) in a cluster communicate among themselves to provide key benefits, including:

• Scalability. Network Load Balancing scales the performance of a server-based program, such as a Web server, by distributing its client requests across multiple servers within the cluster. As traffic increases, additional servers can be added to the cluster, with up to 32 servers possible in any one cluster.

• High availability. Network Load Balancing provides high availability by automatically detecting the failure of a server and repartitioning client traffic among the remaining servers within ten seconds, while providing users with continuous service.

Network Load Balancing distributes IP traffic to multiple copies (or instances) of a TCP/IP service, such as a Web server, each running on a host within the cluster. Network Load Balancing transparently partitions the client requests among the hosts and lets the clients access the cluster using one or more "virtual" IP addresses. From the client's point of view, the cluster appears to be a single server that answers these client requests. As enterprise traffic increases, network administrators can simply plug another server into the cluster.

For example, the clustered hosts in Figure 1 below work together to service network traffic from the Internet. Each server runs a copy of an IP-based service, such as Internet Information Services 5.0 (IIS), and Network Load Balancing distributes the networking workload among them. This speeds up normal processing so that Internet clients see faster turnaround on their requests. For added system availability, the back-end application (a database, for example) may operate on a two-node cluster running Cluster service.



A four-host cluster works as a single virtual server to handle network traffic. Each host runs its own copy of the server with Network Load Balancing distributing the work among the four hosts.

Advantages of Network Load Balancing:

Network Load Balancing is superior to other software solutions such as round robin DNS (RRDNS), which distributes workload among multiple servers but does not provide a mechanism for server availability. If a server within the host fails, RRDNS, unlike Network Load Balancing, will continue to send it work until a network

administrator detects the failure and removes the server from the DNS address list. This results in service disruption for clients. Network Load Balancing also has advantages over other load balancing solutions—both hardware- and software-based—that introduce single points of failure or performance bottlenecks by using a centralized dispatcher. Because Network Load Balancing has no proprietary hardware requirements, any industry-standard compatible computer can be used. This provides significant cost savings when compared to proprietary hardware load balancing solutions.

The unique and fully distributed software architecture of Network Load Balancing enables it to deliver the industry's best load balancing performance and availability. The specific advantages of this architecture are described below in the "Network Load Balancing Architecture" section.

Installing and Managing Network Load Balancing

Network Load Balancing is automatically installed and can be optionally enabled on the Advanced Server and Datacenter Server versions of the Windows 2000 operating system. It operates as an optional service for local area network (LAN) connections and can be enabled for one LAN connection in the system; this LAN connection is known as the cluster adapter. No hardware changes are required to install and run Network Load Balancing. Since it is compatible with almost all Ethernet and Fiber Distributed Data Interface (FDDI) network adapters, it has no specific hardware compatibility list.

IP Addresses

Once Network Load Balancing is enabled, its parameters are configured using its Properties dialog box, as described in the online help guide. The cluster is assigned a primary IP address, which represents a virtual IP address to which all cluster hosts respond. The remote control program provided as a part of Network Load Balancing uses this IP address to identify a target cluster. Each cluster host also can be assigned a dedicated IP address for network traffic unique to that particular host within the cluster. Network Load Balancing never load-balances traffic for the dedicated IP address. Instead, it load-balances incoming traffic from all IP addresses other than the dedicated IP address.

When configuring Network Load Balancing, it is important to enter the dedicated IP address, primary IP address, and other optional virtual IP addresses into the TCP/IP Properties dialog box in order to enable the host's TCP/IP stack to respond to these IP addresses. The dedicated IP address is always entered first so that outgoing connections from the cluster host are sourced with this IP address instead of a virtual IP address. Otherwise, replies to the cluster host could be inadvertently load-balanced by Network Load Balancing and delivered to another cluster host. Some services, such as the Point-to-Point Tunneling Protocol (PPTP) server, do not allow outgoing connections to be sourced from a different IP address, and thus a dedicated IP address cannot be used with them.

Host Priorities

Each cluster host is assigned a unique host priority in the range of 1 to 32, where lower numbers denote higher priorities. The host with the highest host priority (lowest numeric value) is called the default host. It handles all client traffic for the virtual IP addresses that is not specifically intended to be load-balanced. This ensures that server applications not configured for load balancing only

receive client traffic on a single host. If the default host fails, the host with the next highest priority takes over as default host.

Port Rules:

Network Load Balancing uses port rules to customize load balancing for a consecutive numeric range of server ports. Port rules can select either multiple-host or single-host load-balancing policies. With multiple-host load balancing, incoming client requests are distributed among all cluster hosts, and a load percentage can be specified for each host. Load percentages allow hosts with higher capacity to receive a larger fraction of the total client load. Single-host load balancing directs all client requests to the host with highest handling priority. The handling priority essentially overrides the host priority for the port range and allows different hosts to individually handle all client traffic for specific server applications. Port rules also can be used to block undesired network access to certain IP ports.

When a port rule uses multiple-host load balancing, one of three client affinity modes is selected. When no client affinity mode is selected, Network Load Balancing load-balances client traffic from one IP address and different source ports on multiple-cluster hosts. This maximizes the granularity of load balancing and minimizes response time to clients. To assist in managing client sessions, the default single-client affinity mode load-balances all network traffic from a given client's IP address on a single-cluster host. The class Affinity mode further constrains this to load-balance all client traffic from a single class C address space. See the "Managing Application State" section below for more information on session support.

By default, Network Load Balancing is configured with a single port rule that covers all ports (0-65,535) with multiple-host load balancing and single-client affinity. This rule can be used for most applications. It is important that this rule not be modified for VPN applications and whenever IP fragmentation is expected. This ensures that fragments are efficiently handled by the cluster hosts.

Remote Control

Network Load Balancing provides a remote control program (Wlbs.exe) that allows system administrators to remotely query the status of clusters and control operations from a cluster host or from any networked computer running Windows 2000. This program can be incorporated into scripts and monitoring programs to automate cluster control. Monitoring services are widely available for most client/server applications. Remote control operations include starting and stopping either single hosts or the entire cluster. In addition, load balancing for individual port rules can be enabled or disabled on one or more hosts. New traffic can be blocked on a host while allowing ongoing TCP connections to complete prior to removing the host from the cluster. Although remote control commands are password-protected, individual cluster hosts can disable remote control operations to enhance security.

Managing Server Applications:

Server applications need not be modified for load balancing. However, the system administrator starts load-balanced applications on all cluster hosts. Network Load Balancing does not directly monitor server applications, such as Web servers, for continuous and correct operation. Monitoring services are widely available for most client/server applications. Instead, Network Load Balancing provides the mechanisms needed by application monitors to control cluster operations—for

example, to remove a host from the cluster if an application fails or displays erratic behavior. When an application failure is detected, the application monitor uses the Network Load Balancing remote control program to stop individual cluster hosts and/or disable load balancing for specific port ranges.

Maintenance and Rolling Upgrades

Computers can be taken offline for preventive maintenance without disturbing cluster operations. Network Load Balancing also supports rolling upgrades to allow software or hardware upgrades without shutting down the cluster or disrupting service. Upgrades can be individually applied to each server, which immediately rejoins the cluster. Network Load Balancing hosts can run in mixed clusters with hosts running the Windows NT® Load Balancing Service (WLBS) under Windows NT 4.0. Rolling upgrades can be performed without interrupting cluster services by taking individual hosts out of the cluster, upgrading them to Windows 2000, and then placing them back in the cluster. (Note that the first port in the default port range has been changed for Windows 2000 from 1 to 0, and the port rules must always be compatible for all cluster hosts.)

Working mode of Network Load Balancing:

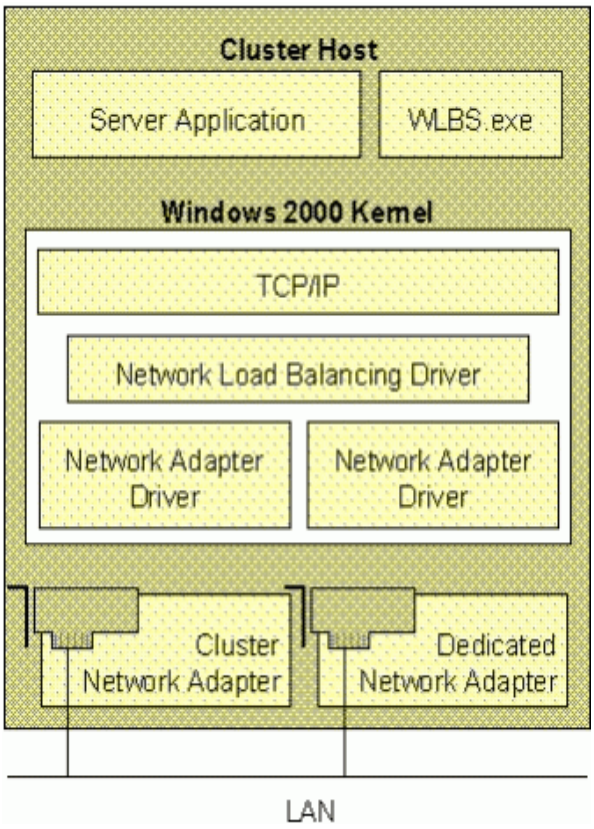
Network Load Balancing scales the performance of a server-based program, such as a Web server, by distributing its client requests among multiple servers within the cluster. With Network Load Balancing, each incoming IP packet is received by each host, but only accepted by the intended recipient. The cluster hosts concurrently respond to different client requests, even multiple requests from the same client. For example, a Web browser may obtain the various images within a single Web page from different hosts in a load-balanced cluster. This speeds up processing and shortens the response time to clients.

Each Network Load Balancing host can specify the load percentage that it will handle, or the load can be equally distributed among all of the hosts. Using these load percentages, each Network Load Balancing server selects and handles a portion of the workload. Clients are statistically distributed among cluster hosts so that each server receives its percentage of incoming requests. This load balance dynamically changes when hosts enter or leave the cluster. In this version, the load balance does not change in response to varying server loads (such as CPU or memory usage). For applications, such as Web servers, which have numerous clients and relatively short-lived client requests, the ability of Network Load Balancing to distribute workload through statistical mapping efficiently balances loads and provides fast response to cluster changes.

Network Load Balancing cluster servers emit a heartbeat message to other hosts in the cluster, and listen for the heartbeat of other hosts. If a server in a cluster fails, the remaining hosts adjust and redistribute the workload while maintaining continuous service to their clients. Although existing connections to an offline host are lost, the Internet services nevertheless remain continuously available. In most cases (for example, with Web servers), client software automatically retries the failed connections, and the clients experience only a few seconds' delay in receiving a response.

Architecture:

File is partitioned to fixed-size chunks Name node manages a centralized directory for accesses like



create, delete, append, etc. Could have a backup standby Data node stores file chunks Data nodes may fail arbitrarily, and be added dynamically Scale: ×10, 000 Both name node and data node are capable of computation and storage. Network Load Balancing runs as a network driver logically situated beneath higher-level application protocols, such as HTTP and FTP. Figure 2 below shows the implementation of Network Load Balancing as an intermediate driver in the Windows 2000 network stack.

Network Load Balancing Performance

The performance impact of Network Load Balancing can be measured in four key areas:

- **CPU overhead** on the cluster hosts, which is the CPU percentage required to analyze and filter network packets (lower is better).
- **Response time** to clients, which increases with the non-overlapped portion of CPU overhead, called *latency* (lower is better).
- **Throughput** to clients, which increases with additional client traffic that the cluster can handle prior to saturating the cluster hosts (higher is better).
- **Switch occupancy**, which increases with additional client traffic (lower is better) and must not adversely affect port bandwidth.

Existing System:

State-of-the-art distributed file systems (e.g., Google GFS and Hadoop HDFS) 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 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 (e.g., the master in Google GFS) become a performance bottleneck, as they are unable to accommodate a large number of file accesses due to clients and Map Reduce applications.

Disadvantages of Existing System:

The most existing solutions are designed without considering both movement cost and node heterogeneity and may introduce significant maintenance network traffic to the DHTs.

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. Additionally, we 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. 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.

- Our proposal not only takes advantage of 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.

Advantages of Proposed System:

- This eliminates the dependence on central nodes.

- Our proposed algorithm operates in a distributed manner in which nodes perform their load-balancing tasks independently without synchronization or global knowledge regarding the system.

- Algorithm reduces algorithmic overhead introduced to the DHTs as much as possible.

Virtual Response:

A novel load-balancing algorithm to deal with the load rebalancing problem in large-scale, dynamic, and distributed file systems in clouds has been presented in this paper. Our proposal strives to balance the loads of nodes and reduce the demanded movement cost as much as possible, while taking advantage of physical network locality and node heterogeneity. In the absence of representative real workloads (i.e., the distributions of file chunks in a large scale storage system) in the public domain, we have investigated the performance of our proposal and compared it against competing algorithms through synthesized probabilistic distributions of file chunks. The synthesis workloads stress test the load-balancing algorithms by creating a few storage nodes that are heavily loaded. The computer simulation results are encouraging, indicating that our proposed algorithm performs very well. Our proposal is comparable to the centralized algorithm in the Hadoop HDFS production system and dramatically outperforms the competing distributed algorithm in terms of load imbalance factor, movement cost, and algorithmic overhead. Particularly, our load-balancing algorithm exhibits a fast convergence rate. The efficiency and effectiveness of our design are further validated by analytical models and a real implementation with a small-scale cluster environment.

Reference:

Load Rebalancing for Distributed File Systems in Clouds Hung-Chang Hsiao, Member, IEEE Computer Society, Hsueh-Yi Chung, Haiying Shen, Member, IEEE, and Yu-Chang Chao http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6175892

Hadoop Distributed File System, http://hadoop.apache.org/hdfs/.

S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in Proc. 19th ACM Symp. Operating Systems Principles (SOSP'03), Oct. 2003, pp. 29-43.

K. McKusick and S. Quinlan, "GFS: Evolution on Fast-Forward," Commun. ACM, vol. 53, no. 3, pp. 42-49, Jan. 2010.

Full Text: Access at ACM

F. B. Schmuck and R. L. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," in Proc. USENIX Conf. File and Storage Technologies (FAST'02), Jan. 2002, pp. 231-244.

Y. Hua, Y. Zhu, H. Jiang, D. Feng, and L. Tian, "Supporting Scalable and Adaptive Metadata Management in Ultra Large-scale File Systems," IEEE Trans. Parallel Distrib. Syst., vol. 22, no. 4, pp. 580-593, Apr. 2011.

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

Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," LNCS 2218, pp. 161-172, Nov. 2001.

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. Operating Systems Principles (SOSP'07), Oct. 2007, pp. 205-220.

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.

M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., 1979.

D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," RFC 3174, Sept. 2001.

J. W. Byers, J. Considine, and M. Mitzenmacher, "Simple Load Balancing for Distributed Hash Tables," in Proc. 1st Int'l Workshop Peer-to-Peer Systems (IPTPS'03), Feb. 2003, pp. 80-87.

[CrossRef]

M. Raab and A. Steger, "Balls into Bins-A Simple and Tight Analysis," LNCS 1518, pp. 159-170, Oct. 1998.

M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-Based Aggregation in Large Dynamic Networks," ACM Trans. Comput. Syst., vol. 23, no. 3, pp. 219-252, Aug. 2005.

M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. V. Steen, "Gossip-Based Peer Sampling," ACM Trans. Comput. Syst., vol. 25, no. 3, Aug. 2007.

Apache Hadoop, http://hadoop.apache.org/.

P. Ganesan, M. Bawa, and H. Garcia-Molina, "Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems," in Proc. 13th Int'l Conf. Very Large Data Bases (VLDB'04), Sept. 2004, pp. 444-455.

H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, and A. Donnelly, "Symbiotic Routing in Future Data Centers," in Proc. ACM SIGCOMM' 10, Aug. 2010, pp. 51-62.

Raicu, I. T. Foster, and P. Beckman, "Making a Case for Distributed File systems at Exascale," in Proc. 3rd Int'l Workshop Large-Scale System and Application Performance (LSAP'11), June 2011, pp. 11-18.