ISSN:2231-0711

V.Hema et al | IJCSET(www.ijcset.net) | October 2015 | Vol 5, Issue 10,344-346

A Survey On Locality Awareness Request Distribution In Cluster Based Network Servers

[1] V.Hema[2] DR. K.Kungumaraj

[1]*Research Scholar, Mother Teresa Women's University, Kodaikanal.*
[2]*Assistant Professor, Department of Computer Science, Arulmigu Palaniandavar Arts College For Women, Palani.*

Abstract – A World Wide Web (WWW) Server is normally a single machine dedicated to process a HTTP request for a single WWW site. The number of people using the Internet has been growing at a very fast rate, while the services provided over the Internet are increasingly becoming mission critical. Hence, enabling high performance, reliability, and availability, as well as the creation of management tools, have become key issues in the development and maintenance of Internet servers. Servers based on clusters of workstations or PCs are the most popular hardware platform used to meet the growth of traffic demands in World Wide Web. A cluster based network server consists of a front-end responsible for request distribution and a number of back end nodes responsible for request processing. In content-based request distribution Front-end takes into account both the service, and content requested with current load on back-end nodes. The current approach for handling these issues from the server perspective is based on the concept of load balancing. Locality-aware request distribution (LARD) is a specific strategy for content aware request distribution that improves cluster performance by simultaneously achieving load balancing and high cache hit rates in the back ends.

Keywords: Data Clustering, Server Load Balancing, Cache Hit Rate.

1. Introduction

1.1 CLUSTER SYSTEMS

A cluster consists of two or more computers working together to provide a higher level of availability, reliability, and scalability than can be obtained by using a single computer. A server cluster is a group of independent servers running and working together as a single system to provide high availability of services for clients. When a failure occurs on one computer in a cluster, resources are redirected and the workload is redistributed to another computer in the cluster. Server clusters are designed for applications that have long-running in-memory state or frequently updated data. Typical uses for server clusters include file servers, print servers, database servers, and messaging servers.

Cluster systems are being increasingly used in the web server management, file distribution and database transaction. The system based with a distributor has a front-end server (distributor), which receives all the requests from the clients. The requests are then forwarded to the bunch of backend servers that contain the actual content for the clients. The requests are forwarded to the backend servers based on various policies. The front-end consider the service/content requested and the current load on the back-end nodes when deciding which back-end node should serve a given request. All back-end nodes are considered equally capable of serving a given request with considering the current load information of the back-end nodes. The load between different back-ends might become unbalanced, resulting in worse performance. Building a LARD cluster is therefore to design a practical and efficient strategy that achieves load balancing and high cache hit rates on the back-ends.

1.2 Load Balancing

Web server serves web pages to clients across the Internet or an Intranet. The web server hosts the pages, scripts, programs, and multimedia files and serves them using HTTP, a protocol designed to send files to web browsers and other protocols. .In order to achieve web server scalability, more servers need to be added to distribute the load among the group of servers, which is also known as a server cluster. The load distribution among these servers is known as load balancing. Load balancing applies to all types of servers, the application server and database server.

Load balancing is a technique that distributes processing and communications activity evenly across a computer network so that no single device is overwhelmed. In other words, when multiple web servers are present in a server group, the HTTP traffic needs to be evenly distributed among the servers. The purpose of load balancing was done due to the increases of traffic, complexity of the application software and to satisfy the critical online transaction nowadays.

2. Related Work

2.1 Round Robin Technique (RR)

Round robin is a process used for network communication and operating system load balancing. A system that works in a round robin fashion distributes load based on the round robin algorithm. The round robin algorithm uses its scheduling techniques to assign processing time slices and transfer queued data packets. Network devices such as routers and switches implement special round robin algorithm buffer queues, which exist in device memory and store incoming and overloaded data for future processing. In this technique, multiple IP addresses are associated with a single domain name; clients are expected to choose which server to connect to. This technique exposes to clients the existence of multiple backend servers. This technique works particularly well where individual servers are spread geographically on the Internet.

Although easy to implement, round robin DNS has problematic drawbacks, such as those arising from record caching in the DNS hierarchy itself, as well as client-side address caching and reuse, the combination of which can be difficult to manage. Round robin DNS should not solely be relied upon for service availability. If a service at one of the addresses in the list fails, the DNS will continue to hand out that address and clients will still attempt to reach the inoperable service. There is no consideration for transaction time, server load, network congestion, etc.

2.2 Weitghted Round Robin (WRR)

It is frequently necessary to distribute processing load based on their individual server capabilities. Round-robin or random load balancing do not focus of this nature. The weighted load balancing policy allows you to specify a processing load distribution ratio for each server with respect to others. You can specify this as a positive processing weight for each server.

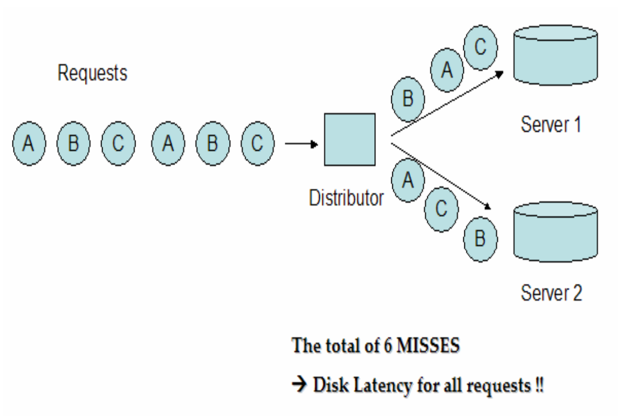

Figure-1: Weighted Round Robin

The weighted round robin policy is applied based on the current load at the backend servers. The policy is applied at the distributor, where the requests are forwarded to the backend servers. The distributor maintains the record of the current load at the backend servers and it forwards the request from the client, based on this information. The request is forwarded to the least loaded backend server among the bunch of servers. The request forwarding is thus weighted based on the current load on the servers. The server that is most loaded is relieved off the load by forwarding the requests to the least loaded server. So, at any given point of time, the load is evenly balanced among all the available servers and thus providing very good load balancing.

The main drawback of the system is that it does not concern about the locality of the requests and the power conservation among the servers. In case of large deployment of cluster systems, the power consumed becomes a very significant factor to be considered. Since all the servers are turned ON during the entire period of operation, the system turns out to conserve zero power. Also, as the system does not consider the locality of the data among the backend servers, the different data requests land up in different servers and incur large disk latencies. This increases the response time (service time) of the servers and hence the throughput. Considering this, power and locality based request distribution policies have more significance.

3. Basic Locality Aware Request Distribution

The objective of LARD is to combine good load balancing and high locality. The front end is responsible for handling new connections and passing data from the client to the back-end nodes. It must keep track of open and closed connections and not involved in handling outgoing data. In Weighted round-robin request distribution the incoming requests are distributed in round-robin fashion, weighted by some measure of the load on the difference back-ends. This strategy produces good balancing among back-ends but does not consider the type of service.

The locality aware request distribution strategy is a form of content-based request distribution, focusing on obtaining the improved cache hit rates in the back-ends, secondary storage capability. If the work set exceeds the size of main memory available for caching documents, frequent cache misses will occur. Figure 1 illustrates the principle of LARD in a simple server with two back ends and three targets (A,B,C) in the incoming request stream.

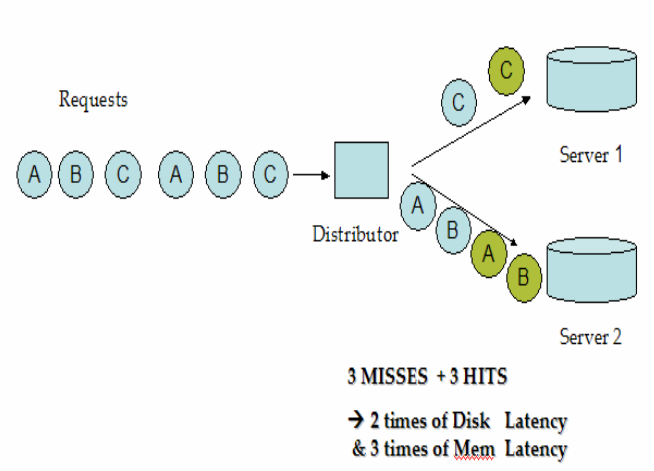

Figure-2: LARD with 3 Front ends and 2 servers

The distributor maintains a table of the data types available at the backend servers' memory. The data types are assigned to the backend servers based on the initial server/data partitioning and are initially distributed evenly across the servers. When a new request arrives at the distributor, its data type is looked up in the distributor table and the corresponding server is identified. The request is forwarded always to that server for that particular data type. By this assignment, the request will incur disk latency only during the first initial assignment to that backend server. The consecutive requests of the same data type end up as server memory hits, since it has already been fetched from the disk and is now in the memory. Once the requests starts overflowing at one of the servers, one of the least loaded servers is added to serve that data type and the server set for that data type starts growing. Similarly, when a server becomes underutilized, a server is removed from the server set. However, when there is no change in target server set for given K seconds, the most load server is removed from the server set. This ensures load balancing up to some extent.

4. Conclusion

The LARD strategy achieves high cache hit rates and good load balancing in a cluster server. The performance of our strategy go beyond that of WRR substantially. Our caching system uses cooperative and exclusive caching for static Web documents . Separate handling of the heavy tail of the request distribution curve may bring further benefits. Further research is required to increase the locality and therefore LARD can apply to dynamic content.

References

[1]. Vivek S. Pai, Mohit Aron, Gaurov Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel, Erich Nahum, “Locality-aware request distribution in clusterbasednetwork servers,” Proceedings of the eighth international conference on Architectural support for programming languages and operating systems, p.205-216, October 02-07, 1998, San Jose, California, United States.

[2] T. 13risco. DNS Support for Load Balancing. RFC 1794, Apr. 1995.

[3] M. J. Fceley, W. E. Morgan, F. H. Pighin, A. R. Karlin, II.M. Levy, and C. A.Thekkath. Implementing global memory management in a workstation cluster. In Proceedings of the FifteentACM Symposium on Operating System Principles, Copper Mountain, CO, Dec. 1995.

[4] K. Rajamani and C. Lefurgy, “On evaluating request-distribution schemes for saving energy in server clusters,” in Proc. Intl. Sym. Performance Analysis of Systems and Software, March 2003.

[5] E. Pinheiro, R. Bianchini, E. V.Carrera and T.Heath,“DynamicCluster Reconfiguration for Power and Performance.” Kluwer Academic Publishers, 2002.

[6] E. V. Carrera, S. Rao, L. Iftode, and R. Bianchini. "User-Level Communication in Cluster-Based Servers". Proceedings of the 8th IEEE International Symposium on High-Performance Computer Architecture (HPCA 8), February 2002.

[7] M. Aron, D. Sanders, P. Druschel, etc, Scalable,Content-Aware Request Distibution in Cluster-Based Network Servers., Proceedings of 2000 USENIX Annual Technical Conference, 2000.

[8] D. Andresen, T. Yang, O. H. Ibarra, .Toward a scalable distributed WWW server on workstation clusters., Journal of Parallel and Distributed Computing, Vol.42, No.1, 10 April 1997, pp.91-100.

[9] Chi-Chung Hui,Samuel T.Chanson. Improved Strategies for Dynamic Load Balancing.IEEE Concurrency,1999.

[10] C.-S. Yang, M.-Y. Luo. A content placement and management system for distributed Web-server systems.Proc. of IEEE 20th Int. Conf. on Distributed Computing Systems (ICDCS’2000), Taipei, Taiwan,Apr. 2000.

[11] Michele Colajanni, Philip S. Analysis of Task Assignment Policies in Scalable Distributed Web-server Systems,IEEE Transactions on Parallel and Distributed Systems,vol.9,No.6,June,1998.

[12] S. Gadde R. P. Doyle, J. S. Chase and A. Vahdat. The Trickle-Down Effect: Web Caching and Server Request Distribution. In Proc. of 6th Int. Workshop on Web Caching and Content Distribution (WCW’01), June 2001.

[13] V. Olaru and W. F. Tichy. CARDs: Cluster Aware Remote Disks. In Proc. of the Third IEEE/ACM Int. Symp. on Cluster Computing and the Grid (CCGrid 2003),May 2003.

[14] M. E. Crovella, R. Frangioso, and M. Harchol-Balter. Connection Scheduling in Web Servers. In Proc. Of the 2nd Usenix Symposium on Internet Technologies and Systems, October 1999.

[15] http://kb.linuxvirtualserver.org/wiki/Weighted Round- Robin Scheduling.

[16] N. Bonvin, T. G. Papaioannou, and K. Aberer, “An economic approach for scalable and highly-available distributed applications,” in Proc. Of IEEE CLoud. 498-505. 2010.

[17] A. Cohen, S. Rangarajan, and J. H. Slye, “On the performance of TCP splicing for URL-aware redirection,” in USENIX Symposium on Internet Technologies and Systems, 1999.

[18] Yuh-Ming Chin, Do Young Eun, 2008. Minimizing File Download Time in Stochastic Peer-to- Peer Networks. IEEE / ACM, 16(2) : 253-266.

[19] Balaji P., S. Narravula, K. Vaidyanathan, S. Krishnamoorthy and D.K. Panda, 2004. Sockets Direct Procotol over InfiniBand in Clusters: Is It Beneficial?. Proc. IEEE International Symp. Performance Analysis of Systems and Software (ISPASS ’04), pp.28-35.

[20] Kim J.H., G.S. Choi, D. Ersoz, and C.R. Das, 2004. Improving Response Time in Cluster - Based Web Servers through Co-scheduling. Proc. 18th International Parallel and Distributed Processing Symposium, pp. 88-97.