www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume - 3 Issue - 8 August, 2014 Page No. 7857-7860

Load Balancing In Clustered Network

[1]Suwaathy Kayalvily. D , [2]Mangayarkarasi. S

[1]Research Scholar,

[2]Assistant Professor,

School of Computing Science

Vels University, Chennai – 43.

Abstract

Network has lots of connections and the server has n number of requests to be handled. To reduce the work load of the server, we create subsequent sub servers and they are used to handle requests from clients. Each client can login and get their required information from the sub server. A report is generated for all the requests handled by the sub servers and the server maintains the log. A fully distributed load balancing algorithm is presented to scope with the load imbalance problem. DSBCA algorithm is compared against a centralized approach in a production system and a competing distributed solution presented in the literature. The Real time 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. The performance of our proposal implemented in the Ha-doop distributed file system is further investigated in a cluster environment. The main reason for the formation of such clusters is that clustered overlays enable their participants to find and exchange data relevant to their queries with less effort.

Keywords- server, cluster, load imbalance, data

1 Introduction

Many distributed real-time applications, such as audio and video-conferencing, require the network to construct a multicast path(tree) from a sender to multiple receivers. Furthermore, real-time applications have quality-of-service(Qos) requirements(e.g.bandwidth). The objective of the routing protocol is to build a tree that is both feasible (i.e. satisfies the requested Qos) and least costly. The cost of a tree depends on the costs of its links. The cost of a link should reflect the effect of allocating resources to the new connection on existing and future connections. In this paper, we examine the effect of various link cost functions on the performance of two classes of multicast routing algorithms under both uniform and skewed real-time workload. We also investigate the impact of inaccurate network state information. It aims at improving the performance of the system and decrease the total execution time. The goal is to find a minimum-cost (sub) network that satisfies some specified property such as k-connectivity or connectivity on terminals (as in the classic Steiner tree problem).Such a formulation captures the (possibly incremental) creation cost of the network, but does not incorporate the actual cost of using the network.

2 System Model

Network Load Balancing forwards each client request to a specific host within a cluster according to the system administrator's load-balancing policy. Each server in the cluster is fully self-contained, which means it should be able to function without any other in the cluster with the exception of the database (which is not part of the NLB cluster). This means each server must be configured separately and run the Web server as well as any Web server applications that are running. If you're running a static site, all HTML files and images must be replicated across servers. If you're using ASP or ASP.Net, those ASP pages and all associated support files must also be replicated. Source control programs like Visual SourceSafe can make this process relatively painless by allowing you to deploy updated files of a project (in Visual Studio.Net or FrontPage for example) to multiple locations simultaneously. To construct scalable Web servers, system builders are using distributed designs. An important challenge that arises in distributed Web servers is the need to direct incoming connections to individual hosts. Previous methods for connection routing have employed a centralized node that acts as a switchboard, directing incoming requests to backend hosts.

A certain number of additional servers can be added to the load-balanced cluster to maximize scalability and stay ahead of increasing demand. In addition to load balancing the key is redundancy – if any machine in the cluster goes down, NLB will re-balance the incoming requests to the still running servers in the cluster. The servers in the cluster need to be able to communicate with each other to exchange information about their current processor and network load and even more basic checks to see if a server went down.

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. To maximize throughput and high availability, Network Load Balancing uses a fully distributed software architecture. An identical copy of the Network Load Balancing driver runs in parallel on each cluster host. 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. 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.

3 DSBCA ALGORITHM

To generate clusters with more balanced energy and avoid creating excessive clusters with many nodes, we have used DSBCA algorithm. The basic idea of this algorithm is based on connectivity on density and the distance from the base station to calculate *k*(clustering radius). The clustering radius is determined by density and distance: if two clusters have the same connectivity density, the cluster much farther from the base station has larger cluster radius; if two clusters have the same distance from the base station, the cluster with the higher density has smaller cluster radius. With farther distance from the base station and lower connectivity density, the cluster radius is larger; on the contrary, with closer distance from the base station and lower connectivity density, the cluster radius is smaller. In order to reduce clusters structure change, we also involve in weight computation of the nodes such parameters as residual energy, connection density and times of being elected as cluster head nodes. DSBCA follows a distributed approach to establish hierarchical structure in self-organizing mode without central control. The purpose of DSBCA is to generate clusters with more balanced energy and avoid creating excessive clusters with many nodes. The clusters near the base station also forward the data from further clusters (all clusters need to communicate with the base station, but long-distance wireless communication consumes more energy), and as we all know, too many members in a cluster may bring excessive energy consumption in management and communication. Hence, based on the above concerns, DSBCA algorithm considers the connectivity density and the location of the node, tries to build a more balanced clustering structure.

Using this algorithm the total number of distributed system or node is selected and the size of each node is specified. Then the distributed nodes is balanced to form hierarchical structure (balancing) and all the sub-nodes also forms a hierarchical structure (rebalancing). The nodes are added to the cluster based on its size. After forming the cluster the authenticated users can access the files. DSBCA sets the threshold of cluster size. The number of cluster nodes cannot exceed the threshold to avoid forming large clusters, which will cause extra overhead and thus reduce network lifetime. When the cluster head node receives Join_message sent by the ordinary node, it will compare the size of cluster with threshold to accept new member and update the count of cluster nodes if the size is smaller than threshold, or reject the request. If the rejected node has cluster head already, the clustering process terminates. Otherwise, it finds another appropriate cluster to join. DSBCA algorithm avoids the fixed cluster head scheme (cluster head manages cluster and forwards data, so it consumes energy faster than other nodes), with periodic replacement to balance the node energy consumption. The cluster head gathers the weight of all member nodes, and then selects the node with highest weight as the next head node. DSBCA can form more reasonable cluster structure to avoid frequent exchange of the nodes weight information and temporary cluster head broadcasting after the first clustering. As a result, the energy consumption decreases effectively. The clusters formed by DSBCA based on the distance from base station, distribution of nodes and residual energy accord with actual network. Hence, it achieves a better performance when the number of nodes changes.

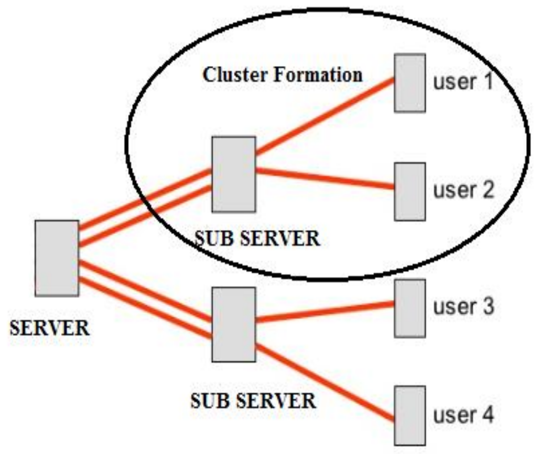

Figure 1

Figure 1 shows the cluster formation and various users accessing the cluster (sub server) to get their required data. Depending upon the technology used to provide this functionality, a certain number of additional servers can be added to the load-balanced cluster to maximize scalability and stay ahead of increasing demand. The performance of the nodes in load balancing environment depends upon the selection of balancing instants and the load exchange allowed between nodes. If the network delays are large it would be more advisable to reduce the amount of load exchange so as to avoid the time wasted. So the amount of load transfer has to be chosen carefully and scheduling has to be done regularly in order to maintain load balancing in the system. When an external load arrives at a node, only the receiver node executes, which aims to minimize the average completion time.

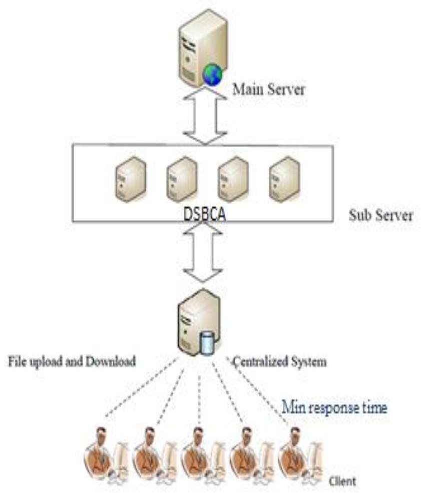

Figure 2

The figure 2 depicts the use of DSBCA algorithm. This eliminates the dependence on central nodes. The storage nodes are structured as a network based on distributed hash tables and they enable nodes to self-organize and repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management. Our algorithm is compared against a centralized approach in a production system and a competing distributed solution presented in the literature. The simulation results indicate that although each node performs our load balancing algorithm independently without acquiring global knowledge.

4 Conclusion

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 and a fully distributed load balancing algorithm is presented to cope with the load imbalance problem. The simulation result shows that the algorithm is feasible and has better performance. In addition, the scenario we propose is scalable and works for different network sizes.

References

[1] Rajkumar Buyya et.al. "High Performance Cluster Computing" Vol 1, Prentice Hall PTR, 1999

[2] G. Pfister. "In Search of Clusters. Prentice Hall", 2nd Edition, 1998

[3] "Clustering" by Rui Xu, Don Wunsch, ieee press series on computational intelligence.

[4] "Cluster Analysis" ,5th edition, Brian S. Everitt , Sabine Landau , Morven Leese , Daniel Stahl

[5] "Server load balancing" By Tony Bourke Copyright © 2001 O'Reilly & Associates,Inc. Allrights reserved. Printed in the United States of America. August 2001.

[6] Zhu Y, Hu Y. Efficient, proximity-aware load balancing for dht-based p2p systems. IEEE Trans Parallel Distributed Syst (TPDS);2005.

[7] Dahlin et al, "Eddie: A Robust and Scalable Internet Server". Ericsson Telecom AB. Sweden. 1998.

[8] Damani et al. "ONE-IP: Techniques for Hosting a Service on a Cluster of Machines", Sixth International WWW Conference, April 1997.

[9] Daniel M. Dias, William Kish, Rajat Mukherjee, and RenuTewari, "A Scalable and Highly Available Web Server",Proceedings of IEEE COMPCON'96.

[10] M. Chatterjee, S. K. Das, and D. Turgut, “WCA: A weighted clustering algorithms for mobile ad hoc networks,” Cluster Computing., vol. 5, no. 2, pp. 193–204, 2002.

[11] Y. Fernandess and D. Malkhi, “K-clustering in wireless ad-hoc networks,” in Proc. 2nd ACM Workshop Principles Mobile Compuingt. Conf.,Oct. 2002, pp. 31–37.

[12]http://msdn.microsoft.com/en-us/library/ff648960.aspx

[13]http://docs.oracle.com/cd/E13222_01/wls/docs81/cluster/load_balancing.html

[14]http://www.iis.net/learn/web-hosting/configuring-servers-in-the-windows-web-platform/network-load-balancing

[15]https://www.vmware.com/files/pdf/implmenting_ms_network_load_balancing.pdf

[16]http://technet.microsoft.com/en-us/library/cc725691.aspx

[17] Paul Barford and Mark Crovella. Generating Representative Web Workloads for Network and Server Performance Evaluation. In Proceedings of ACM SIGMETRICS, 1998.

[18] E.D. Katz, M. Butler, and R. McGrath, “A scalable HTTP server: The NCSA prototype. In Proceedings of the First International World-Wide Web Conference, May 1994.

[19] D. Anderson, T. Yang, V. Holmedahl, and O.H. Ibarra.“SWEB: Towards a Scalable World Wide Server on Multicomputers”. In Proceedings of IPPS'96, April 1996.

[20] Eric Anderson, David Patterson, and Eric Brewer. “The MagicRouter: An application of fast packet interposing.” Available from http://HTTP.CS.Berkeley.EDU/~eanders/projects/magicrouter/osdi96-mr-submission.ps, May1996.

[21] Jeffery Mogul. “Network behavior of a busy Web server and its clients”. Research Report 95/5, DEC Western Research Laboratory, October 1995.