Volume 3, No. 5, Sept-Oct 2012
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
REVIEW ARTICLE
Available Online at www.ijarcs.info

Load Balancing Techniques for Web Proxy Cache Clusters

Najat O. Alsaiari*
Department of Computer Sciences
King Abdulaziz University
Jeddah, Saudi Arabia
nalsaiari@kau.edu.sa

Ayman G. Fayoumi
Department of Information System
King Abdulaziz University
Jeddah, Saudi Arabia
afayoumi@kau.edu.sa

Abstract: Web caching is a crucial technology in Internet because it represents an effective means for reducing bandwidth demands, improving Web server availability and reducing network latencies. Web cache cluster which is a potent solution to enhance Web cache system's capability still have limited capacity and cannot handle tremendously high workload. How to maximize resource utilization and system capability is an all important problem in Web cache cluster and load balancing is an effective method to solve this problem. The aim of this paper is to present performance analysis of various loading balance techniques used in traditional Web cache proxy systems based on identified qualitative parameters, considering two typical load balancing techniques static and dynamic. The analysis indicates that i) static and dynamic both types of algorithm can have advancements as well as weaknesses over each other, ii) dynamic algorithms are always considered better than static algorithms due to the frequent load state of cache cluster and iii) both types of algorithm have a common issue related to resource utilization i.e. existing load balanced Web proxy systems are lack of flexibility in resource provision.

Keywords: Proxy Cluster, Loading Balance, Web Caching, Resource Utilization.

I. INTRODUCTION

In the recent years, the World Wide Web (WWW) application provides a simple access to a wide range of information and services. As a result, the amount of traffic over the Internet has experienced tremendous growth, and obtaining ease and high speed of browsing Web and loading files through the Internet is often a high requirement among clients. Unfortunately, clients often experience delays when accessing the Internet due to hardware limitations, or low quality service management systems [1]. In order to satisfy clients' expectations, the Internet services delay needs to be bounded by a small value and thus caching Web documents is an effective solution of reducing this delay. Caching involves storing copies of Web pages on a local disk. If the same pages are requested at a later time, and the cached copy is still valid, it will be sent directly instead of contacting the origin server again.

The main purpose of a Web proxy server is to save network resources and to reduce user-perceived network latency by filtering and caching Web traffic [1]. Proxy servers were originally used in allowing internet access to users who were in the same firewall. During those times, companies would use a special type of HTTP servers called "proxy" on their firewall machines for security reasons [1].

This proxy server typically configures those requests that are in a firewall by forwarding them to the remote servers, receiving the responses, and sending them back to the clients. As a result, this was seen as an opportunity to cache documents since clients that are within the same firewall typically share the same proxy servers and thus likely share similar interests. The documents that they would request probably would be the same and it would be easy to browse them within shorter periods. Cooperative proxy system, such as web proxy cache cluster, is a potent solution to enhance web proxy cache system's capacity. Since the 1990's, studies on cooperative proxy system become more and more intensive. Researches of cooperative proxy system are focus on proxy cooperation protocol and load balancing strategy. Proxy cooperation is usually implemented through proxy cooperation protocol, which indicates how the proxies share the local cache with each other (e.g. ICP [2], Summary Cache [3] and CARP [4]).

Loading balancing actually means distributing the traffic evenly across the network along maintaining the response time. Load balancing can be achieved by many ways out of which the ways which will be discussed in this paper are Time Round Robin, Hashing based algorithm, Least Loaded, Threshold algorithm, Central Queue algorithm and Local Queue algorithm.

Section 2 identifies the benefits of Web caching. Section 3 describes the web caching architectures. In section 4, we focus in common loading balance techniques used in proxy cache clusters. Section 5 gives the differentiating parameters to analyze loading balance algorithms. Section 6 concludes the paper with some final remarks and identifies the future research directions.

II. THE APPEAL OF WEB CACHING

The potential benefits of Web caching are multifold [5]. Deploying caches close to clients can reduce the user perceived network performance and improve their Web experience in two ways. First, when serving users locally, caches hide network latencies. Second, network outages will be hidden to users of a caching system since local cache can be leveraged regardless of network availability and thus making the network appear to be more reliable. In other hand, deploying caches close to content providers (e.g. the reverse proxy server approach) can improve the availability

and scalability of Web server under existing or anticipated demand. In addition, caching can reduce amount of Web traffic. For enterprises that pay the ISPs for wide area network bandwidth based on the amount of network traffic, reduced traffic means lowered costs.

III. WEB CACHING ARCHITECTURES

A. Single Proxy Cache:

To reduce the required bandwidth over costly dedicated Internet connections, proxy caches are normally situated close to network gateways. Figure1 shows a configuration of a standalone proxy.

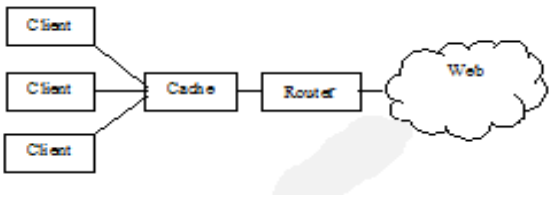

Figure 1. A Single proxy Cache

There are two disadvantages to the design shown above. One is that the cache signifies a single network failure point. The other is that no more caches can be dynamically added if there is need to. Fact is that the single proxy caches have a limited capacity making it a must to replace objects in order to create room for other new objects that may need to be put in the cache. This problem gave rise to proposals for policies on cache replacement. The policies were aimed at using the limited storage space in order to bring the highest caching results. Cache replacement policies are well presented in [6]. One of these policies is the least frequently used (LFU) and the least recently used (LRU) which were among the earliest to be proposed in the disk caching system and computer memory framework. LRU displaces those object that were least used recently if storage space is required while LFU displaces those objects that were least used frequently. An-other policy is known as "size policy" which was designed to be used by Web proxy caches to evict the biggest object in the cache if storage space is required. The eviction decisions are done by using a cost function which considers a number of relevant factors. These factors include size, retrieval latency of documents, and reference popularity. In general, the cache replacement algorithms reviewed in [6], [7] and [8] usually maximize the cache hit ratio (the number of times that objects in the cache are referenced) by attempting to cache the data items which are most likely to be referenced in the near future. Unfortunately, it is very difficult, if not impossible, to predict the future user needs [7].

a. Optimized Disk I/O:

The constraints of proxy's computational resources (e.g., storage and CPU cycles) are still a major drawback of single proxy system. Thus, many systems, especially those that are commercial, have spent substantial time tuning their disk I/O, treating the object cache as one does a high performance database. Markatos et al. [9] has studied the disk I/O over-head of traditional world-wide web proxy, and proposed a set of techniques aimed at reorganizing the file system layout to improve performance in Squid system. To avoid CPU or disk arm overhead in Active cache system [10], the proxy simply refuses to serve the hit cache request that consuming its local resources and redirects it to the original server. In [11], the Web server plays a role in reducing proxy resources overhead. The former technique weights the performance tradeoffs and determines whether migration of a data file to a proxy add burden to the proxy or not.

Other disk I/O optimizations include using in-memory data structures and improving the spatial locality of objects to avoid disk I/O altogether [12]. This technique exploits in-memory data structures (i.e., hash tables) to summarize the contents of a cache so it can be used to quickly determine if an object has been cached; if querying the data structures finds that the object has indeed been cached, disk operations can begin to actually locate the object (if not already in RAM). Otherwise, costly disk access can be avoided altogether and the object can be retrieved from the originating server.

B. Cooperative Proxy Cache:

Increase in demand of internet services and expansion of internet in recent years made it very hard for a single proxy server with resource constraints to operate and serve the needs of clients. Therefore, many studies suggest cooperation among the caching proxies to address these scalability issues associated with single caches [13, 14, 15]

Cache networks bring about several advantages [5]. First, through sharing caches among a large number of users, more efficient utilization of caching resources can be realized when compared with a single cache approach. Second, caching networks provide a natural solution to applications that involve serving a large, geographically dispersed user population in support of their diverse Web requests, since multiple caches can be strategically located between the users and original Web servers. Third, caches networks help improve the overall performance of the caching system by balancing loads between proxies. Furthermore, they improve the network fault tolerance and robustness by removing the single point of failure. Though web caching offers much hope for better performance and improved capability, there remain a number of ongoing issues such as replacement strategies and cache consistency. These two issues are out of scope in this paper; rather, we mainly focus in a third issue which is related to the coordination among participating caches. Since loading balance among participating caches has to be carefully designed in order to optimize the use of resources and maximize throughput as well as minimize response time.

IV. LOADING BALANCE IN COOPERATIVE PROXY SYSTEM

Load balancing is a computer networking methodology to distribute workload across multiple computers or a computer

cluster, network links, central processing units, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload.

Load imbalance is the primary obstacle for maximizing performance of cooperative proxy system. Load imbalance in cooperative proxy system is caused by unbalanced re-quests to objects in different proxies' cache. The existence of hot spot worsens the imbalance. In order to overcome the load imbalance, enhance system efficiency, many load balancing strategies are proposed. Kun-Lung Wu [16] presented adaptable controlled replication (ACR) to reduce load imbalance. In ACR, a two level LRU stack is implemented. With redundant hot spot, the imbalance in proxy system could be alleviated. But ACR can only relieve the imbalance caused by hot spot.

a. Loading Balance Algorithms:

Load balancing algorithms can be broken down into dynamic and static load balancing techniques [17]. These two load balancing techniques are discussed below.

Static Load Balancing. The performance of the proxies determined at the beginning of execution. Then depending upon their performance the work load is distributed in the start by the master proxy. The slave proxies calculate their allocated work and submit their result to the master. A task is always executed on the proxy to whom it is assigned. This type of algorithm has a benefit since it brings about ease of implementation and overhead reduction. This is because monitoring the performance statistic of workstations is not needed. However, a general disadvantage of all static schemes is that the final selection of a proxy for task allocation is made when the task is created and cannot be changed during process execution to make changes in the system load (i.e. non primitive scheme).

Dynamic Load Balancing. This kind of load balancing technique adjusts the distribution of tasks based on run time by using recent or current load information whenever they make a choice on distribution of a task. Unlike static algorithms, dynamic algorithms allocate tasks dynamically when one of the proxies becomes under loaded. As a result, dynamic load balancing algorithms can provide a significant improvement in performance over static algorithms. How-ever, this improvement carries an additional cost of gathering and maintaining load information.

A. Time Round Robin

This algorithm determines the destination proxy server based on the time that the user sends the request. It distributes requests evenly to all proxies. With equal workload round robin algorithm is expected to work well. In general, Time Round Robin is a stateless non adaptive algorithm which does not consider the information of cached objects at each proxy.

B. Hashing Based Algorithm:

A proxy server is selected based on a hash value computed from the requested URL (e.g. SuperProxy [18]). When a client or a proxy needs to locate a copy of the requested Web object, it applies this shared hash function to the requested URL and then contacts the proxies identified by the returned hash value. The hash function-based method utilizes cache space efficiently because no multiple copies of the Web objects need to be maintained. The main disadvantage of this method is the need for all clients and proxies to use the same global hash function. The coordination overhead is nontrivial when this global function needs to be updated because of the changes in the cache network. Moreover, one requested URL may get various kinds of objects returned including images, banners, or flash video, and they all have different sizes. Thus, at any moment, it is possible that all proxy servers are in use or only one server is used. Thus, this technique does not guarantee the load balancing among proxy servers.

C. Threshold Algorithm:

The According to this algorithm [20], the tasks are assigned immediately upon creation to proxies. Proxies for new tasks are selected locally without sending remote messages. Each proxy keeps a private copy of the system's load. The load of a proxy can characterize by one of the three levels: underloaded, medium and overloaded. Two threshold parameters tender and topper can be used to describe these levels.

Under loaded - load < tender
Medium - tender ≤ load ≤ topper
Overloaded - load > topper

Initially, all the proxies are considered to be under loaded. When the load state of a proxy exceeds a load level limit, then it sends messages regarding the new load state to all remote proxies, regularly updating them as to the actual load state of the entire system. If the local state is not overloaded then the task is allocated locally. Otherwise, a remote under loaded proxy is selected, and if no such host exists, the task is also allocated locally. Thresholds algorithm have low inter process communication and a large number of local task allocations. The later decreases the overhead of remote process allocations and the overhead of remote memory accesses, which leads to improvement in performance. A disadvantage of the algorithm is that all tasks are allocated locally when all remote proxies are overloaded. A load on one overloaded proxy can be much higher than on other overloaded proxies, causing significant disturbance in load balancing, and increasing the execution time of an application.

D. Lowest Load Algorithm [19]:

It determines the destination proxy server based on the current workload of each proxy server. The current workload is determined from the number of log records created by Squid on each proxy server where each log record represents one object request. Unlike Time Round Robin and Hashing based algorithms, Least Loaded takes into account the number of requests handled by each proxy and the requested sizes that reflect the cache size on Squid and then send the request to the target proxy server that currently have the lowest workload. In fact, using static weight to integrate load information can not reflect system load state precisely. Thus, in Web cache proxy system which has dynamic task resource

demand feature this technique is inefficient [17]. Moreover, the response time of a request can be affected by network delay since the location of each proxy was not taken into account when assigning a request to a proxy server located far away.

E. Central Queue Algorithm:

Central Queue Algorithm [21] works on the principle of dynamic distribution. It stores new activities and unfulfilled requests as a cyclic FIFO queue on the master server. Each new activity arriving at the queue manager is inserted into the queue. Then, whenever a request for an activity is received by the queue manager, it removes the first activity from the queue and sends it to the requester. If there are no ready activities in the queue, the request is buffered, until a new activity is available. If a new activity arrives at the queue manager while there are unanswered requests in the queue, the first such request is removed from the queue and the new activity is assigned to it. When a slave proxy load falls under the threshold, the local load manager sends a request for a new activity to the central load manager. The central load manager answers the request immediately if a ready activity is found in the process request queue, or queues the request until a new activity arrives.

F. Local Queue Algorithm:

Main feature of this algorithm [21] is dynamic process migration support. The basic idea of the local queue algorithm is static allocation of all new processes with process migration initiated by a proxy when its load falls under threshold limit; the parameter defines the minimal number of ready processes the load manager attempts to provide on each proxy. Initially, new processes created on the master proxy are allocated on all under loaded slave proxies. The number of parallel activities created by the first parallel construct on the master proxy is usually sufficient for allocation on all remote proxies. From then on, all the processes created on the master proxy and all other proxies are allocated locally. When the proxy gets under loaded, the local load manager attempts to get several processes from remote proxies. It randomly sends requests with the number of local ready processes to remote load managers. When a load manager receives such a request, it compares the local number of ready processes with the received number. If the former is greater than the latter, then some of the running processes are transferred to the requester and an affirmative confirmation with the number of processes transferred is returned.

V. IDENTIFICATION OF DIFFERENTIATING PARAMETERS

The performance of various load balancing algorithms is measured by the following parameters.

A. Nature of Loading Balancing Techniques:

Static load balancing assigns load to nodes probabilistically or deterministically without consideration of runtime events. It is generally impossible to make predictions of arrival times of loads and processing times required for future loads. On the other hand, in a dynamic load balancing the load distribution is made during run-time based on current processing rates and network condition. However, static algorithms only work well when there is not much variation in the load on the workstations. For Cache proxy system which is having significant variations of loads using dynamic load technique is more sufficient than static techniques [17].

B. Overload Rejection:

If Load Balancing is not possible additional overload rejection measures are needed. When the overload situation ends then first the overload rejection measures are stopped. After a short guard period Load Balancing is also closed down.

C. Cooperation:

This parameter gives that whether proxies share information between them in making the process allocation decision other are not during execution. What this parameter defines is the extent of independence that each proxy has in concluding that how should it can use its own resources. In the cooperative situation all proxies have the accountability to carry out its own portion of the scheduling task, but all proxies work together to achieve a goal of better efficiency. In the non-cooperative individual proxies act as independent entities and arrive at decisions about the use of their resources without any effect of their decision on the rest of the system.

D. Forecasting Accuracy:

Forecasting is the degree of conformity of calculated results to its actual value that will be generated after execution. Dynamic algorithms using single resource load information does not reflect the system load state precisely, rather study shows that correlation coefficient between resource load information and system performance index represents the extent of resource's influence on system performance. In general, the static algorithms provide more accuracy than of dynamic algorithms as in former most assumptions are made during compile time and in later this is done during execution.

E. Centralized or Decentralized

Centralized schemes store global information at a designated node. All sender or receiver nodes access the designated node to calculate the amount of load-transfers and also to check that tasks are to be sent to or received from. In a distributed load balancing, every node executes balancing separately. The idle nodes can obtain load during runtime from a shared global queue of processes.

F. Process Migration:

Process migration parameter provides when a system decides to export a process. It decides whether to create it locally or create it on a remote processing element. The algorithm is capable to decide that it should make changes of load distribution during execution of process or not.

G. Resource Utilization:

Resource utilization include automatic load balancing A distributed system may have unexpected number of processes that demand more processing power. If the algorithm is capable to utilize resources, they can be moved to under loaded proxies more efficiently.

The comparison of various load balancing algorithms on behalf of the above parameters is shown in Table 1.

VI. CONCLUSION

In this paper, we review various loading balance techniques used in Web cache proxy clusters. The comparison table shows static algorithms are more accurate and stable in compare to dynamic and it is also ease to predict the behavior of static. In other hand and due to the frequent load state of cache cluster, dynamic algorithms are always considered better than static algorithms. However, there is a common issue in the reviewed load balancing techniques related to resource utilization. As can be provisioned there is always a trade-off between improving service quality and enhancing resource utilization in Web proxy cache cluster. When system load is low, it will cause resource wasting and when system load becomes high, it will lower service quality. Thus, utilize the resource efficiency with QoS support is an open issue and needs to be addressed in order to achieve outstanding performance, higher resource efficiency and lower system cost.

Table I. Parametric Comparison of Load balancing Techniques

Parameters	time round robin	Hashing based algorithm	threshold algorithm	lowest Load	Central queue	local queue
Dynamic/static	S	S	S	Dy	Dy	Dy
overload rejection	No	No	No	No	Yes	Yes
cooperative	No	No	Yes	No	Yes	Yes
forecasting accuracy	More	More	More	Less	Less	Less
centralized/decentralized	D	D	D	C	C	D
process migration	No	No	No	No	No	Yes
resource utilization	Less	Less	Less	Less	Less	Less

VII. ACKNOWLEDGMENT

This paper contains the results and findings of a research project that is funded by King Abdulaziz City for Science and Technology (KACST), Grant No: T-T-12-0938.

VIII. REFERENCES

[1] J. Wang. A survey of web caching schemes for the internet. ACM Computer Communication Review, 29(5):36–46, Oct. 1999.

[2] D. Wessels, K. Claffy. “ICP and the Squid Web Cache”,IEEE Journal on Selected Areas in Communications, Vol.16, No.3, 1998, pp 345-357.

[3] L. Fan, P. Cao, J. Almeida, “Summary Cache: A Scalable Wide-Area Web Cache sharing Protocol”, IEEE/ACM TRANSACTIONS ON NETWORKING, Vol.8, No.3, 2000, pp: 281- 293.

[4] Cache Array Routing Protocol, http://technet.microsoft.com/en-us/cc723281.aspx

[5] D. Zeng; F.Y. Wang; Mingkuan Liu; , "Efficient web content delivery using proxy caching techniques," Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on , vol.34, no.3, pp.270-280, Aug. 2004.

[6] K.-Y. Wong; "Web cache replacement policies: a pragmatic approach," Network, IEEE , vol.20, no.1, pp.28-34, Jan.-Feb. 2006.

[7] M. Arlitt, R. Friedrich, and T.Jin, “Performance Evaluation of Web Proxy Cache Replacement Policies,” Performance Evaluation, 39(1-4):149–164, February 2000.”

[8] J. Xu; Q. Hu, W.-C. Lee, D. L. Lee; "An optimal cache replacement policy for wireless data dissemination under cache consistency," Parallel Processing, International Conference on, 2001. , vol., no., pp. 267- 274, 3-7 Sept. 2001.

[9] E.P. Markatos; D.N. Pnevmatikatos; M. D. Flouris; M.G.H Katevenis; "Web-conscious storage management for Web proxies," Networking, IEEE/ACM Transactions on , vol.10, no.6, pp. 735- 748, Dec 2002.

[10] P. Cao, J. Zhang, and K. Beach. Active cache: Caching dynamic contents on the web. Proc of IFIP Intl Conf on Distributed Systems Platforms and Open Distributed Processing, 1998.

[11] W. Hao, Q.K. Ma, I. L. Yen, I. Chen; A Weblet environment to facilitate proxy caching of web. In: Proceedings of Parallel & Distributed Computing and Systems, Marina del Rey, California, November, 797–802 (2003).

[12] G. Tomlinson, D. Major, and R. Lee. High-capacity internet middleware: Internet caching system architectural overview. Second Workshop on Internet Server Performance, 1999.

[13] Y. Wang; G. Y. Du; T. S. Huang; Y. Wang; "A load balancing model for web cache proxy based on ant colony behavior," Machine Learning and Cybernetics, 2008 International Conference on , vol.4, no., pp.2192-2197, 12-15 July 2008.

[14] S. Michel, K. Nguyen, A. Rosenstein, L. Zhang, S. Floyd, and V. Jacobson, “Adaptive web caching: Toward a new global caching architecture,” in Comput. Networks ISDN Syst., vol. 30, Nov. 1998, pp. 2169–2177.

[15] Z. Duan, Z. Gu, X. Ding; "WPCC: A novel web proxy cache cluster," Advanced Communication Technology, 2009.

ICACT 2009. 11th International Conference on , vol.03, no., pp.2205-2208.

[16] K. L. Wu, P. S. Yu, “Load Balancing and Hot Spot Relief for Hash Routing among a Collection of Proxy Caches”, Proceedings of the 19th IEEE International Conference on Distributed Computing Systems, IEEE, Austin, Texas, USA , 1999, pp. 536 -543.

[17] Z. Duan; Z. Gu; , "Dynamic Load Balancing in Web Cache Cluster," Grid and Cooperative Computing, 2008. GCC '08. Seventh International Conference on , vol., no., pp.147-150, 24-26 Oct. 2008.

[18] R.B. Bunt, D.L. Eager, G.M. Oster, and C.L. Williamson, “Achieving Load Balance and Effective Caching in Clustered Web Servers”, Proceedings of the 4th International Web Caching Workshop, San Diego, California, USA, pp.159-169, April 1999.

[19] S. Ngamsuriyaroj; P. Rattidham, P; I. Rassameeroj; P. Wongbuchasin,; N. Aramkul; S. Rungmano; "Performance Evaluation of Load Balanced Web Proxies," Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on , vol., no., pp.746-750, 22-25 March 2011.

[20] S. Sandeep, S. Sarabjit and S. Meenakshi, Performance Analysis of Load Balancing Algorithms, World Academy of Science, Engineering and Technology, 2008.

[21] W. Leinberger; G. Karypis; V. Kumar; R. Biswas; "Load balancing across near-homogeneous multi-resource servers," Heterogeneous Computing Workshop, 2000. (HCW 2000) Proceedings. 9th , vol., no., pp.60-71, 2000.