2002 International Computer Symposium (ICS2002)
Workshop on Computer Networks
December 18-21, 2002

Analysis and Design of Routing Based Hierarchical Internet Cache Protocol Architecture

Kuo Chang Ting, ting@mail.mhit.edu.tw
Tel: 03-5593142~3571 Fax: 03-5593142-3579
Dept. of Computer Science & Information Engineering, National Taiwan University, Taipei 106, Taiwan
Feipei Lai, flai@cc.ee.ntu.edu.tw
Tel:02-23639352
Dept. of Computer Science & Information Engineering,
Dept. of Electrical Engineering,
National Taiwan University, Taipei 106, Taiwan

Abstract

The rapid growth of the WWW has caused the Internet traffic serious congestion in recent years. Proxy cache is a good technology to solve this problem by distributing web server file objects. However, standing alone proxy cache often suffers from low hit ratio due to huge WWW cache working set size. Internet Cache protocol (ICP) was used to construct cooperative proxy caching architectures to increase the Internet cache hit ratio or make a load balance between cooperative proxies to share requests of large scale clients. However, ICP especially hierarchical cache architecture has the disadvantage of poor response time due to extra network communications when the parent cache misses occur. In fact, if the hit rate of the parent cache is high, and this parent cache is on the *nearby routing path* [1]of the children proxies to the Web server sites, extra network traffic will be minimized. It will decrease the clients' response time surely. We demonstrate our viewpoint that the importance of parent

[1] On the near side of some router, this resides on the routing path to the Web server.

cache is on the *nearby routing path* of the child cache to the Web server sites by showing a simple formula and analyzing proxy server log of MHIT (Ming-Hsin Institute of Technology). We construct a mathematical model to describe some hierarchical Internet cache behaviors. These Internet cache behaviors are "*reverse lookup"* and "*reverse cache effect"*, which are described in this article. Furthermore, we develop a methodology to evaluate the cache time of the selected parent cache and analyze the proxy cache log to estimate the routing effect on the cache time of our simulated proxy servers. Evaluation shows that routing factor, whether the proxy server is on the near routing path of the child cache to the Web server sites or not, is more much subtle to the hit time than the miss time of the parent cache. The miss time of the parent cache is dependent on the relationship between the web distribution and the routing position of this proxy server.

Keywords: ICP, DCM, proxy, hierarchical proxy cache, reverse effect, reverse hit effect

1. Introduction and Related Work

WWW traffic has grown rapidly in the recent year. Proxy caches were introduced to avoid the bottlenecks of the classical clients/server Internet architecture [1, 2, 4, 9, 10, 11, 16, 17]. The Internet cache to the WWW is like the fast cache, which is used to cover the huge speed gap between processor and main memory. For modern CPU, its clock rate can be up to 1.5 GHz. In one clock cycle, it can issue several instructions simultaneously. One instruction can be less than one *ns*. However, for common fast DRAM, one memory access may be up to 80 ns or more. Fast memory cache can solve this speed gap by caching hot memory blocks. Memory

cache plays an important role in modern architecture of computer, because its hit ratio can be up to 0.998. However, the hit ratio of WWW cache over 0.6 is not common because of the huge size of WWW caches and one-referenced documents. ICP was proposed and used to expand the scalability of the proxy server. If the proxy server is on behalf of large-scale of clients, the huge size of requests may overwhelm proxy server cache. The proxy server can not grape the working set size leading to low hit ratio and low access latency. According to our long-term experiment results, we find that disk bandwidth is also critical to the performance of the proxy server. Squid is a famous freeware to construct a proxy server. For squid 2.3 stable version, synchronous I/O is the default setting. Every object is a file, so whenever a cache object is cached or purged out there is a file creation or deletion disk operation in a proxy system. It will consume a lot disk I/O time. According to our polygraph simulation, the throughput of common PC with IDE hard disk is never over 15 requests/seconds. Under this circumstance, the disk bandwidth is just about 15 * 4KB / seconds = 60 KB /seconds if we assume average object size is 4KB. Asynchronous I/O and disk optimization for the file system are good solutions. However, every system has its disk and network bandwidth limitation. Mesh (or sibling or flat) cooperative proxy cache architecture is used to discover this problem. On the other hand, hierarchical cache architecture is used to decrease the cache miss time of the child cache and extra network traffics. In contrast, careful design of mesh architecture of ICP is also critical to the access latency for clients [12, 20]. Every request from the client must poll the sibling proxy server if

the master proxy encounters a cache miss under this mesh architecture of ICP. If all sibling proxies reply a cache miss, the master proxy must redirect this request to the original Web server. The total of all these round trip time is large compared to that of no ICP proxy. In order to avoid this long round trip time, the master proxy must be able to predict whether the requested objects are in the sibling proxies or not. Summary cache [21] was proposed to solve this problem. The motivation of the summary cache is that the master cache keeps the summary information of all the sibling proxies' caches. If requested by the clients for some objects, this master proxy will search for his cache and summary cache to resolve whether the requested objects are in the cooperative proxy group or not. If this requested object is not in the cooperative proxy group, this master proxy will redirect the request to the original server to avoid long round trip time. However, summary cache will consume a lot of the master proxy server resource. In order to keep cache summary coherent, master cache must poll the sibling cache for the summary information periodically. This will also complicate the design of master proxy and burden the network traffic of this cooperative proxy group. Sinisa Srbljic et al. [21] classify the distributed cache management (DCM) into two classes that are ICP and Berkley protocol. The DCM protocol consists of five parts, which are admission algorithm, search algorithm, decision algorithm, replication algorithm, and termination algorithm. ICP communicates with sibling cache by TCP/IP, so it has high network traffic between cooperative caches, but the access latency is low because of this connection-oriented communication, which can avoid long timeout penalty. On the contrary, Berkley protocol

communicates with other sibling caches by UDP, which is a light network traffic and connectionless service. The Berkley protocol queries the sibling cache based on timeout, so it has high clients requests response time, but the scalability of the proxy number can be up to over 200 because of this low network traffic. Dykes et al. [22] also make a viability analysis of cooperative proxy caching. They tried to model the speedup in average user response time for proxy cooperation and derived expressions for the upper bound, a mesh organization and a hierarchical organization. However, all these analyses lack overall network architecture about the Internet architecture concept. Any network (commonly is LAN) of the enterprise or the school runs the same routing protocol, either static link, or distance vector or the other, is an Autonomous System (AS). However, for the extra-AS, hierarchical routing is applied in the Internet world. If the parent cache is on the nearby routing path of the children cache, it may not only reduce the network traffic between these two Autonomous Systems, but also can decrease the average miss time of this child cache. The hops count between these two ASes, network bandwidth of its link, parent cache replacement policy and parent cache size are also critical to the response time of the clients. After all, if the child cache encounters a cache miss, it must redirect the request to the original server through the routing path if there is no hierarchical proxy cache at all. However, if the parent proxy is on the *nearby routing path*, *it will serve for the child cache in passing*. If there is a cache hit in the parent cache, this request will response to the child cache immediately without routing the packet to this distant web site. However, if it is a miss, the parent cache can retrieve this object from the web server

through the remaining routing path and response this cache object to the child cache. There is no network overhead in this architecture. On the contrary, if the parent cache is not in the *near routing path*, it will not only burden the hops between parent and child cache, but also incurs a long access latency and extra network traffics. We illustrate this concept in Figure 1.

In this article, first, we will analyze the advantages of routing based hierarchical ICP[2] compared to non-routing based hierarchical ICP mathematically. In the next section, we will make an analysis of proxy log of MHIT, which is a child cache of NCTU parent cache, and run many simulations to demonstrate the routing effect on the routing and non-routing base hierarchical ICP architectures. Finally, we will make a brief conclusion.

2. Analysis of the Routing-based and Non-routing-based Hierarchical ICP Architectures

Now the Internet architecture is based on the TCP/IP protocol. All the messages are segmented into data packets. The routers based on layer three, route the packet to its destination IP address, which carried on its IP header. Optimal routing is an important issue of the computer network, since there are numerous papers discussing the related issue. Distance vector and link state have been the most famous routing protocols since several teen years ago. However, for the scale and administrative autonomy, hierarchical routing is used. As the number of routers becomes large, the overhead involved in computing, storing, and communicating the routing table information (for example, link-state updates or least-cost

[2] The parent cache is on the near routing path of the child cache to the Web server.

path changes) becomes prohibitive. Today's public Internet may consist of millions of interconnected routers and more than 50 million hosts. Storing routing table entries to each of these hosts and routers would clearly require enormous amounts of memory. The overhead required to broadcast link state updates among millions of routers would leave no bandwidth left for sending data packets! A distance vector algorithm that iterates among millions of routers would surely never converge! To solve this scale and administrative autonomy, routers, which are set up in one region, are aggregated together to run the same routing algorithm and have information about each other. The region is commonly called *Autonomous System.* The routing algorithm running within an autonomous system is called an *intra-autonomous system routing protocol*. However, routers in an AS that have the responsibility of routing packets to destinations outside the AS are called gateway routers. The gateway routers must know how to route (that is, determine routing paths) among themselves in order to route packets from one AS to another AS (possibly passing through multiple other ASs before reaching the destination AS), The routing algorithm, which used among the various ASes is known as an inter-autonomous system routing protocol. Considering Figure 1, we can get the simple concept that the child cache will suffer large miss penalty and hit penalty gap between selecting *B* and *F* as the parent cache.

If *D* is a child cache, setting all link cost to 1 for simplicity, the average cost of the link to the Internet is set to 3, all the children caches' hit ratio is 0.6, and all the parent caches' hit ratio is 0.5, we can get the following reduction by considering three cases. We define child cache miss response time as T_{Miss}.

(1) No parent cache: $T_{Miss} = 2 * (1+1+3) = 10$, $T_{tot} = 0.4 * 10 = 4$.

(2) Selecting *B* as the parent cache: $T_{Miss} = 0.5 * 1*2 + 0.5 *((1+3)+(1+3+1)) = 1 + 4.5 = 5.5$, $T_{tot} = 0.4 * 5.5 = 2.2$.

(3) Selecting *A* as the parent cache: $T_{Miss} = 0.5 * 2*2 + 0.5 *(1+1+3+3+1+1) = 2+5 = 7$, $T_{tot} = 0.4 * 7 = 2.8$.

(4) Selecting *F* as the parent cache: $T_{Miss} = 0.5 *4 * 2 + 0.5 * (1*4+2*1 + 3+ 3 + 2*1 + 1*4) = 13$, $T_{tot} = 0.4 * 13 = 5.2$.

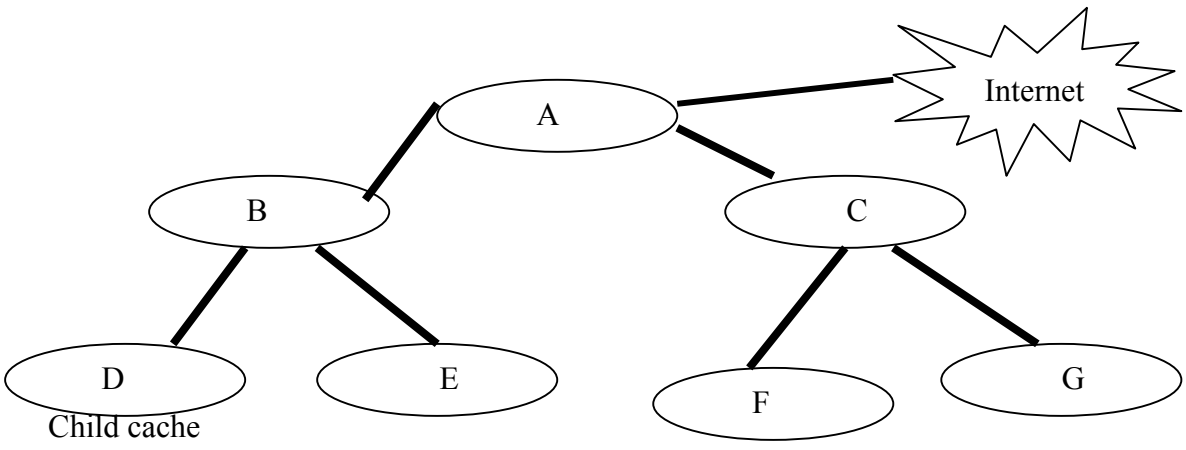

Figure 1: concept for inter-AS connection

You can find that the cache miss time of selecting *F* as the hierarchical parent cache is worse than that of no parent cache at all. This is due to the long parent cache hit access time and overlapped routing path miss time. We can demonstrate this problem by the following model

in the general case:

Let $p = (p_1, p_2, ..., p_n)$ is an optimal hierarchical routing path to the Internet. We define *nearby routing path* as if proxy cache is located on the same AS with any gateway router p_i (for $i = 1,2,...,n$) and p_i and proxy cache is linked directly without any hop delay, we say that the proxy cache is on the *nearby routing path* of p_i to the Web server sites. We assume p_i is either a proxy cache or a gateway router because hop delay is low for p_i and cache. Now, we assume the following conditions hold:

(1) All costs between hops are 1 for simplicity.

(2) p_1 is the child cache and p_n is the final gateway router connected to the Internet.

(3) p_j is the branched gateway router if we select non-routing base parent cache q. $c(p_j, q) = \varepsilon$

(4) Reverse Lookup: If p_k is a branch of some web server of requested objects and selecting $p_i (k < i)$ as a parent cache, the requested objects' time may be longer than that of directed from the web server even if the requested objects are hit accesses. We call this phenomenon is *reverse lookup*. We describe the concept as in Figure 2.

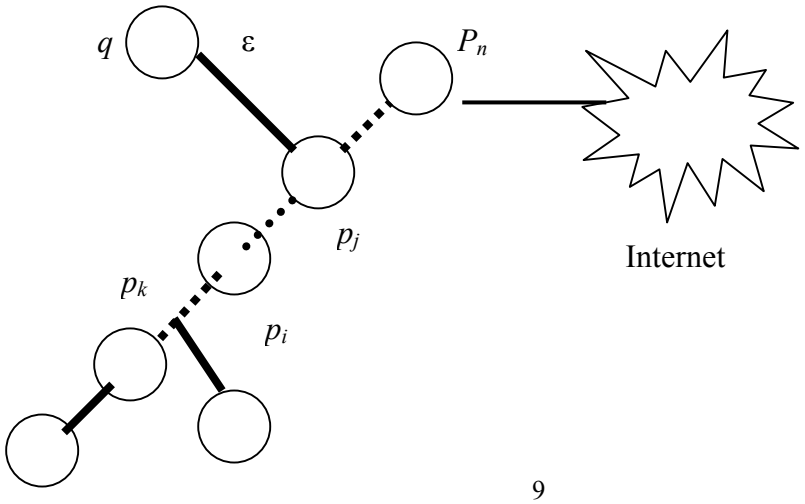

Figure 2: Concept of routing relationship between child

Now we only consider the no reverse lookup condition:

$$Miss(p_i) = hit\_ratio * hit\_penalty + miss\_ratio * miss\_penalty$$
$$= 2 * h * (i-1) + (1-h) * ((n-1) + (n-1+2\lambda))$$
$$= 2h(i-1) + 2(1-h)(n-1+\lambda) = 2(h(i-1) + (1-h)(n-1+\lambda))$$
$$Miss(q) = hit\_ratio * hit\_penalty + miss\_ratio * miss\_penalty$$
$$= 2 * h * (j-1+\varepsilon) + (1-h) * ((n-1) + (n-1+2\lambda+2\varepsilon))$$
$$= 2h(j-1) + 2(1-h)(n-1+\lambda+2\varepsilon) = 2(h(i-1) + (1-h)(n-1+\lambda+\varepsilon))$$

$\delta(q, p_i) = Miss(q) - Miss(p_i) = 2(h * (j-i+\varepsilon) + (1-h) * \varepsilon) = 2(h(j-i) + \varepsilon)$, where λ is the hop delay between Web server and p_j, and $i, j > 1$, $i \neq j$. We conclude that the cost gap between the cost of selecting *q* as the parent cache and that of selecting p_j as the parent cache is $2(h(j-i) + \varepsilon)$. From this reduction, first, we can get that improperly selecting parent cache will cause this situation that the parent cache with high hit ratio tends to have much worse performance than that with low hit ratio. We call this situation as *reverse hit effect*. Second, ε value affects the cache time of the parent cache.

3. Evaluation & Methodology

However, in real Internet world, the cost of hop delay is not simple at all. It is too simple to define the cost of hop delay as one. Hops delay consists of queuing delay, transmission delay, and propagation delay of every hops, and hop count. Propagation delay can be ignored if the distance of two hops is short, but it is large if the connected routers are far from each other. In generally, the propagation speed is in the range of $2 \times 10^8 meters/\sec$ to $3 \times 10^8 meters/\sec$ that depends on the physical link media. It is equal to or less than the light speed. We can get $propagation\_delay = D_{prop} = s/d$, where *d* is the distance between these two routers, and *s* is their physical link speed. In the wild-area networks, propagation

delays are on the order of milliseconds. As to queuing delays of hops, it is the most complicated network behavior in the Internet network. Routers may just queue and transmit a packet without any waiting time in the best case. This is a basic cost for one hop delay. However, the transmission delays will diverse very much. Some links of hops may be ATM backbone. Its transmission rate may be up to several hundred Mbps, but some links may be just 64 Kbps lease lines.

3.1. Methodology:

In order to evaluate the different cache miss time penalties caused by selecting the different parent caches, which reside in different routing paths, we get the cache latency of the parent cache as shown in the following formula:

$cache\_latency = hit\_latency + miss\_latency$. So, when we want to select one Internet proxy server as the parent cache, the main routing path of the hit time, the various possible miss routing paths' time, this parent proxy server delay time, and hit ratio are critical factors to the cache time of this parent proxy server. The algorithm of our methodology is:

(1) Estimate the parent proxy server delay time: The Internet cache size is very huge generally. It is hard to cache all objects in main memory. Disk cache is the only choice up to now. The improved file and optimal I/O system can have an average 20ms/object (4KB average) service time. However, main memory can be as low as 80 *ns*. We conclude that it may be just about 15 *ms* if the proxy server is optimal for commercial purpose.

(2) Estimate the cache hit time: The routing time of the parent and the child cache may be stable, so it is not hard to estimate this latency.

(3) Estimate the hit ratio of the parent cache: According to our simulation, the results show that about 50% of the misses are un-cacheable requests, but these un-cacheable requests may suffer long server latency compared to the routing time. We can bypass it to predict real routing time precisely. The hit ratio may be just about 50% excluding the un-cacheable requests.

(4) Estimate the miss time: The child miss requests take about 45% of all the child cache requests if its hit ratio is 0.55. All these miss requests will be redirected to the parent cache, so estimating the miss time is very important and complex. Web objects' distribution of this child cache is one of critical factors affecting the miss time of the parent cache. We can find the web objects' distribution by tracing the proxy log of the child cache.

(5) Estimate the cache time: cache time = (average hit time) × h + (average miss time) ×(1-h). average hit time = proxy server delay time + 2 × (hit routing time). Average miss time = 2 × (average miss routing) + web server delay. Hit routing time can be estimated as total hops delay (in *ms*) plus transmit time that is $\sum(D_i + ((((object\_size / packet\_size) + 1) \times packet\_size + 5\_level\_header\_size) / T_i) \times 1000ms) / \alpha)$, where D_i is the i*th* hop delay time, T_i is the i*th* link speed, α is the bandwidth usage percentage ($1 \geq \alpha > 0$) in the best case.

3.2 Example:

The proxy of Ming-Hsin Institute (MHIT) is hierarchically connected to the proxy of the National Chiao-Tung University (NCTU). The MHIT proxy is a child node relative to the NCTU proxy. The NCTU campus network is on the backbone of the TaNet, which is an AS of Education Department, Taiwan. One the contrary, the MHIT campus network must route the traffics to the NCTU to connect to the Internet. The link between these two proxies is 45 Mbps T3 line. All the two proxies use the Cache Flow 3000, which is a mature commercial product. We draw the architecture in Figure 3.

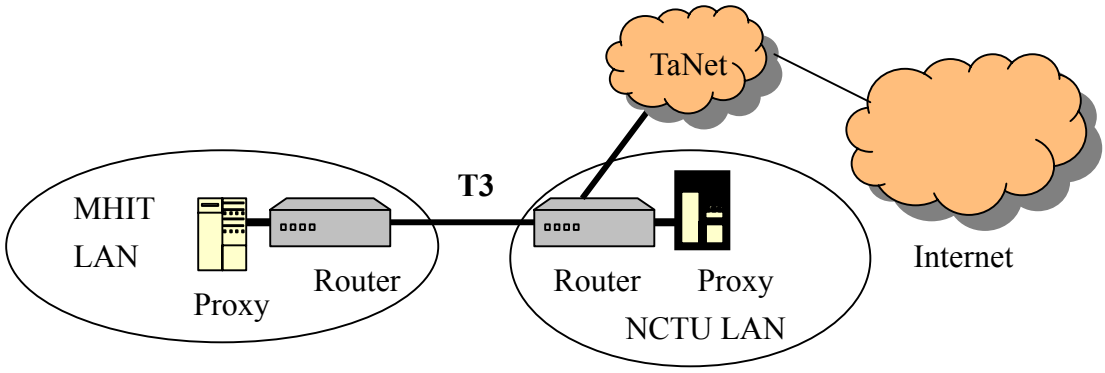

Figure 3: the Internet architecture of Mhit and

In order to analyze about 15 days (2002/05/07, 2002/05/14~2002/05/23) huge trace log data, we write a C program to simulate the situations that if selecting NCHU as the parent cache. The log data shows that short latency or large file such as 40kbytes and not long latency requests are hit requests. We exclude all the non-cacheable miss requests such as CGI and PERL queries because these are too long latencies, which are the search engine servers' time and transfer time that will affect our study results.

The hit latency is the hop delay and transfer time between the child cache and the parent cache. We find that the hit latency of the NCTU parent cache is about 15*ms* to 30*ms* for about 2.5 K object size. The link speed of MHIT and NCTU is T3 (45Mbps). However, the 15~30 *ms* is dominated by several factors, such as server delay, hop propagation delay and transmission delay. The server delay time may be short or long , and it is up to the hit is a memory hit or a disk hit. We can consider that the 15~30*ms* request latency may be a disk hit, but the 10~20 ms may be a memory hit. Under this assumption, we could set one hot delay for transferring 2.5k object size is 15*ms* to evaluate different parent caches routing time. In this experiment, we want to know that the routing effect on the cache time of selecting different parent proxies, and it will cause routing and non-routing base architectures. The link speed between NCHU and backbone is T1, which is 1.5 Mbps. We use the "traceroute" command on the Sun Unix's Platform to measure the average hop delay. The test results are about 30 *ms*. In order to evaluate the cache time of the selected parent proxy, we set the hit time of NCHU cache as (15 ms + (60*ms* + (((*object_size* + 200 *bytes*) / *packet_size* + 1) × *packet_size* × (1 / 4500) + 1 / 150)) / α)), where α is bandwidth usage percentage ($0 < \alpha \leq 1$), 15 *ms* is the average server delay, and 60 *ms* is the average hop delay, 200 bytes is about 5 levels header overhead. As to the miss time, it is more complex than the hit time evaluation. In order to simplify our estimation, we set the cache miss time of the NCHU is larger than that of the NCTU when requesting the URLs such as

".com," ".com.tw", ."ntu.edu.tw", but it will be smaller when requesting URLs such as "nsysu.edu.tw", which resides in northern part of Taiwan, but are not so many in our proxy log, so we set the miss time gap is about the same as the gap of hit time. We draw the architecture of our target study network architecture as in Figure 4.

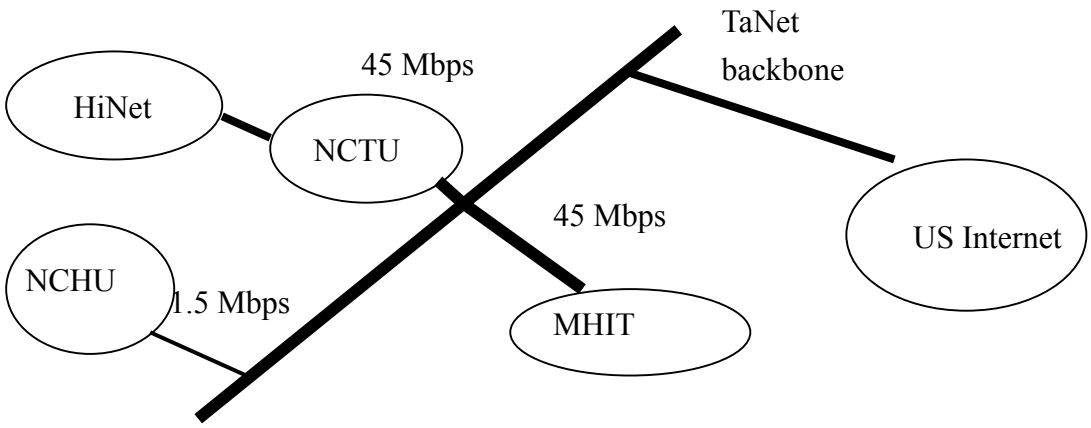

Figure 4: Simple architectures of NCTU

We get the results as in Table1, which are described as in Figures 5, 6, 7.

Table 1: NCTU proxy and simulated NCHU proxy cache time

Parent	Hit Time (Tot)	Hit Count	Hit (ms)	Tot. M.T.	Miss Count	Miss (ms)	Non_Cache_Miss	N.C.M count	Cache Time
NCTU	157664647	1303355	121.0	5788471904	1611617	3591.7	53529876806	8538872	2039.9
NCHU	364394618	1303355	279.6	7236702794	1611617	4490.3	57311958348	8538872	2607.6
No Parent									3571.7
								Bias	27.83%

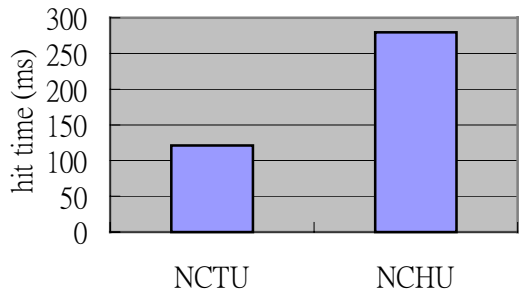

Figure 5: Average Hit Time

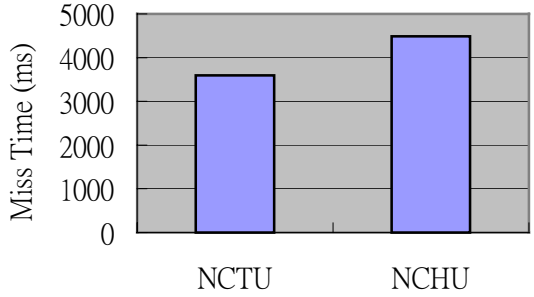

Figure 6: Average Miss Time

Figure 5 shows the average hit time of the NCTU parent cache, which is routing-based

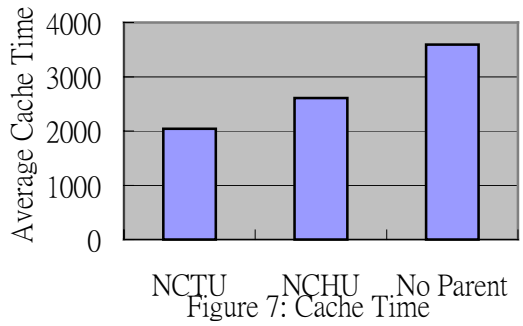

Figure 7: Cache Time

architecture is 120 *ms*. On the contrary, the average hit time of the simulated NCHU is up to 279 *ms*. Figure 6 shows that the average miss time of the NCTU is 3591 *ms*, but the average miss time of the NCHU hit requests is 4490 *ms*. Figure 7 shows that when we select NCTU as MHIT proxy's parent node, the average miss penalty of the MHIT proxy is 2039 *ms*, but it is about 2607 *ms* when we select NCHU as its parent proxy. The bias cache time is about 27%. On the contrary, if there is no parent cache available, all the hit requests of the parent will be the miss requests of this child node.

After all, if we keep it in mind that the link cost between child and parent must be small. It is important for the parent cache is on the *near routing path* to the web server, else the child cache will suffer large miss penalty even if the hit ratio of the parent cache is high, leading to much more serious client response time (*Reverse Hit Effect*). In order to avoid *reverse lookup*, *the parent cache must avoid caching the web contents, which are much closer to the child cache*. One simple implementation is that the child cache must avoid requesting the web contents that belong to its domain.

4. Conclusion

Web cache is a good approach to solve the WWW traffic congestion. However, the WWW traffic and the number of clients increase very quickly recently. Any machine has its limitation such as disk and network bandwidth limitation. In order to solve this scale

problem, cooperative proxy architectures were proposed. There are two main cache architectures, sibling and hierarchical. The hierarchical architecture has the advantage that can reduce the child cache miss penalty if a proper parent cache is selected. In this paper, we try to study the routing factor on the cache miss time effect to select two different parent caches, which are routing base and non-routing base. First, we use a simple mathematical model to describe the network behaviors of the hierarchical cache architecture. These network behaviors are "*reverse lookup"* and *"reverse hit effect"*. Second, we use the real data logs of the MHIT proxy cache and mathematical formula to estimate the cache time of the simulated parent cache (NCHU). Our evaluations show that the parent cache hit time could be rising sharply if a non-routing base parent cache is selected. However, the cache miss time is not so critical especially when non-cacheable misses are considered. In order to avoid "*reverse hit effect"* and *"reverse lookup"*, the relationship of the child's main routing path and parent cache response time is one of the critical issues in the hierarchical proxy cache architecture.

5. Reference:

[1]. "Squid Internet object Cache", *http://www.nlanr.net/Squid/.*

[2]. Burns, R.C.; Rees, R.M.; Long, D.D.E, "Efficient data distribution in a Web server farm", IEEE Internet Computing, Vol. 5, Issue 4, July-Aug. 2001 pp. 56 –65.

[3]. Chankhunthod, P. Danzig, C. Neerdaels, M. Schwartz, and K. Worrell, "A Hierarchical Internet Object Cache", Proc. USENIX 1996 Ann. Technical Conf., URL: http://excalibur.usc.edu/cache-html/cache.html, 1996, pp. 1 – 12.

[4]. Charu Aggarwal, Joel L. Wolf, and Philip S. Yu, “Caching on the World Wide Web”, IEEE Transactions on Knowledge and Data Engineering, Vol. 11, Issue 1, January/February 1999, pp. 94 – 107.

[5]. F. Douglis, A. Feldmann, and B. Krishnamurthy, “Rate of Changes and Other Metrics: A Live Study of the World Wide Web,” USENIX Symp. Internet Technologies and Systems, Montercy, Calif., Dec. 1997, pp. 147-158.

[6]. D. Foygel, D. Strelow, ”Reducing Web latency with hierarchical cache-based prefetching”, 2000 International Workshops on Parallel Processing , pp. 103 – 108.

[7]. James F. Kurose, Keith W. Ross, “Computer networking”, Addison Wesley, pp. 38 - 44.

[8]. John Dilley, “Improving Proxy Cache Performance: Analysis Of Three Replacement Policies”, IEEE Internet Computing, November-December 1999, pp. 44 - 50.

[9]. Junho Shim, Peter Scheuermann and Radek Vingralek, “Proxy Cache Algorithms: Design, Implementation, and Performance”, IEEE Transactions on Knowledge And Data Engineering, Vol. 11, Issue 4, July/August 1999, pp. 549 – 562.

[10]. Luigi Rizzo and Lorenzo Vicisano, “Replacement Policies For a Proxy Cache”, IEEE/ACM Transactions on Networks, Vol. 8, No. 2, April 2000, pp. 158 – 170.

[11]. M. Abrams, C. Standridge, G. Abdulla, S. Williams, and E. Fox, “Caching proxies: Limitations and Potentials,” Proc. Fourth Int.’l World Wide Web Conf., 1995. http://ei.cs.vt.edu/~succeed/WWW4/WWW4.html.

[12]. M. Sayal, Y. Breitbart, P. Scheuermann, and R. Vingralek, “Selection Algorithms for Replicated Web Servers,” Proc. Workshop on Internet Server Performance, 1998. http://www.cs.wisc.edu/~cao/WISP98/html-versions/mehmet/SelectWeb1.html.

[13]. Meira, W., Jr.; Fonseca, E.; Murta, C.; Almeida, V, ”Analyzing performance of cache server hierarchies”, Computer Science, 1998. SCCC '98. XVIII International Conference of the Chilean Society, 1998, pp. 113 – 121.

[14]. Narlikar, G.Lakshman, Y.N. Tin Kam Ho, “TaBLA: a client-based scheduling algorithm for

Web Proxy Clusters", IEEE International Conference on Performance, Computing, and Communication, 2001, pp. 217 – 227.

[15]. Norifumi Nishikawa, et al., "Memory-Based architecture for distributed WWW caching proxy", Computer Networks and ISDN System 30 (1998) pp. 205 – 214.

[16]. P. Cao and S. Irain, "Cost-Aware WWW Proxy Caching Algorithms," Proc. USENIX Symp. Internet Technologies and Systems, 1997, pp. 1 – 14.

[17]. S. Glassman,, "A caching relay for the World Wide Web", 1998, http://pigeon.elsevier.nl/cgi-bin/ID/WWW98.

[18]. S.J. Caughey, D.B. Ingham, and M.C. Little, "Flexible Open Caching for the Web," Proc. Sixth Int'l World Wide Web Conf., 1998, pp. 1 – 14. http://www.scope.gmd.de/info/www6/technical/paper159/paper159.html.

[19]. Yu, H.; Estrin, D.; Govindan, R, "A hierarchical proxy architecture for Internet-scale event services ", Enabling Technologies: Infrastructure for Collaborative Enterprises, 1999. (WET ICE '99) Proceedings. IEEE 8th International Workshops, 1999, pp. 78 – 83.

[20]. Li Fan; Pei Cao; Almeida, J.; Broder, A.Z.,"Summary cache: a scalable wide-area Web cache sharing protocol", IEEE/ACM Transactions on Networking, Vol. 8, Issue 3, June 2000, pp. 281 – 293.

[21]. Srbljic, S.; Milanovic, A.; Hadjina, N., "Performance tuning of large-scale distributed WWW caches", Electrotechnical Conference, 2000 MELECON 10th Mediterranean, Vol. 1, 1998, pp. 93 – 96.

[22]. Dykes, S.G.; Robbins, K.A, "A viability analysis of cooperative proxy caching", INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, Vol. 3, 2001, pp. 1205 – 1214.