Journal of Southeast University (English Edition) Vol. 20 No. 4 Dec. 2004 ISSN 1003—7985

Research and implementation of a new web cache strategy

Yi Faling[1,2] Xie Changsheng[1] Han Dezhi[1] Cai Bin[1]

([1] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)
([2] College of Computer Science and Technology, Yangtze University, Jingzhou 434023, China)

Abstract: This paper presents a new shared cache technique — the grouping cache, which can solve many invalid queries in the broadcast probe and the control bottleneck of the centralized web cache by dividing all cooperative caches into several groups according to their positions in the network topology. The technique has the following characteristics: The overhead of multi-cache query can be reduced efficiently by the cache grouping scheme; the compact summary of the cache directory can rapidly determine if a request exists in a cache within the group; the distribution algorithm based on the web-access logs can effectively balance the load among all the groups. The simulation test demonstrates that the grouping cache is more effective than any other existing shared cache techniques.

Key words: web cache; shared cache; distribution algorithm; grouping cache

The world wide web (WWW) has experienced exponential growth in recent years. This growth has created a tremendous increase in network loads that have subsequently affected user response time. This demand has motivated many research efforts aiming at improving WWW performance. And web caching has been widely acknowledged as an important and promising method. Sun, et al. [1] find that using a well-designed web cache with a 50% hit rate is more effective than doubling the bandwidth for an ISP's (Internet service provider) access link to the Internet, with respect to decreasing retrieval latency.

The web cache technology experiences three phases: the client cache, the proxy cache, and the distributed proxy cache[2]. The distributed proxy cache is popularly adopted for its high availability and scalability[3]. A fundamental question which the distributed proxy architectures must resolve is how a cache receives a request for an object when it does not find out if any other cache has the object before declaring a global miss[4]. The four main existing approaches include a broadcast probe, a hash-partitioning of the object namespace among caches, a directory service and a summary cache, with which the global miss can be reduced. But these approaches may increase the latency of the web accesses, in general, when the number of shared caches increases. The proposed web cache system contains several cache groups, and its performance improvement can be achieved by the group routing and querying.

Received 2004-03-28.
Foundation items: The National Natural Science Foundation of China (No. 60173043), the National Basic Research Program of China (973 Program) (No. G1999033006).
Biographies: Yi Faling (1969—), male, graduate; Xie Changsheng (corresponding author), male, professor, csxie@263.net.

1 Related Work

The broadcast probe is based on the Internet cache protocol (ICP), which is an application level protocol built on top of the user datagram protocol or any other transport layer protocol[5]. When a request comes from the client, the cache will first check to see if it can meet the request. If it cannot do so, the proxy cache queries all the sibling caches. As the number of collaborating proxies increases, the overhead quickly becomes prohibitive. The hash routing protocol schemes are deterministic approaches for mapping a web page to a unique sibling cache[5]. Hashing distributes the uniform resource locator (URL) space among the sibling caches creating a single logical cache spread among many caches. Conceptually, whenever a user asks for a page, it is the responsibility of the client to extract the URL and to find the hash value with the hash function. Depending upon the hash value, the request for the page is sent to the cache server corresponding to that hash value. Since all web requests require the central hashing service, this service might become a bottleneck or a single point of failure.

The simple directory service is implemented in caching and replication for Internet service performance (CRISP)[6]. CRISP servers cooperate to share their caches, using a central mapping service with a complete directory of the cache contents of all participating proxies. To probe the cooperative cache, the proxy forwards the requested URL to a mapping server. Similar to the hash routing protocol, a central mapping service may become a bottleneck when the

number of cooperative cache becomes larger. Li and Pei, et al. [7] present a cache sharing protocol of the summary cache, in which each proxy keeps a compact summary of the cache directory of every other proxy. When a cache miss occurs, a proxy first probes all the summaries to see if the request is a cache hit in other proxies, and sends a query message only to those proxies whose summaries show promising results. The key to the scalability of this scheme is that summaries do not necessarily have to be up-to-date or accurate. But an inaccurate summary will quickly enhance the probability of false hits and false misses along with an increase in the number of proxy caches, and a reduction in the the performance of the system.

2 Grouping Cache

In this section, a proposed grouping cache strategy for wide-area web cache sharing will be presented, in which all cooperative caches are organized into small groups according to their positions in the network topology. Each cache group only saves web pages of the given URLs dispatched by the distribution algorithm. Each cache keeps an information table, which includes two parts: ① The group routing information; ② The summary of the cache directory within the group. When the request comes from the client, the cache receiving a request determines the group to which the request belongs, by group routing information. If the request does not belong to the current group, the request is forwarded to the new group. Then the summary of the cache directory is checked to see if the request is granted in the new group. In this way, a cache hit/miss can be decided through forwarding at most once.

There are two important research issues associated with this approach: ① How to keep load-balance among all the groups; ② How to construct the URL routing table. Since network stations which a group of users have accessed are relatively invariable during a period of time, it is reasonable to determine which network stations they will access through their recent web-access logs. Therefore, we extract the network station information from URLs which users have accessed recently, and determine what should be saved by each group with the distribution algorithm.

2.1 Implementation of URLs' interval

Each URL is composed of a network location and a path (including file name) such as http: //news.sohu.com/78/40/news213174078.shtml. The front part (http: //news.sohu.com/) is a network location, and the latter is its path. In a cache mesh, contents are distributed to a cache group according to the network location of the URL (corresponding to network station approximately). In order to keep a balanced load among groups, we adopt the distribution algorithm according to the web-access logs of regional network concerned with the shared caches.

When constructing the group routing table, we mainly take into account the front three parts of the URL, which are used to make the URL's interval for a cache group. Although these parts cannot entirely express domain names (DNs), they do not affect the correctness of the distribution algorithm. The process of making the URL's interval includes the following steps. Firstly, construct a string (called DNab) by fetching the initial two characters from three DNs' area in sequence. If there are fewer than two characters or nothing in some domains, the null character is used to fill in. E.g., the DNab corresponding to "hust.edu.cn" is "huedcn", and the string corresponding to "a.b" is "a b". Secondly, count each string's accesses by the web-access logs. Thirdly, distribute DNs to every group according to DNabs' accesses with the distribution algorithm. Finally, make discrete DNabs become a continuum. In this way, the continuous character interval of URLs is captured.

2.2 Distribution algorithm

In web-access logs, the number of each DN's accesses is definite. How to assure the balance of DNs' accesses which are assigned to each group is an NP-hard in fact. Provided that the total capacities of each group are equal, this problem is described as follows.

Assume that $S = \{s_1, s_2, \cdots, s_k\}$ is the set of DNabs' accesses, $G = \{g_1, g_2, \cdots, g_n\}$ is the set of cache groups, A denotes an assignment scheme, and $N_p(A)$ denotes the total accesses of the p-th cache group. Thus the objective function of the optimal assignment scheme A_{opt} is

$$N(A_{\text{opt}}) = \min_{A}(\max_{p} N_p(A))$$

Because each DN's accesses cannot be partitioned, this problem resembles non-separable optimal task assignment(NSOTA)[8]. It is very difficult to solve the problem, so an effective approximate algorithm is designed in which the elements of S are sorted out firstly, then DNs are assigned to each group in order according to the average to which each group should be assigned. To balance the dispatch numbers of different groups, a compensation factor is added. With the like-C language, the algorithm is described as follows:

Input: n is the number of cache-groups, G_i is the i-th cache-group, k

is the number of strings, $\{s_1, s_2, \cdots, s_k\}$ is the set of DNabs' accesses.

Output: $s_i (1 \leqslant i \leqslant k)$ is distributed to the corresponding group.

Sort $\{s_1, s_2, \cdots, s_k\}$ in descending order, namely $s'_1 \geqslant s'_2 \geqslant \cdots \geqslant s'_k$;

$s'_1 + s'_2 + \cdots + s'_k = c$;

$i = 1$;

while($s'_i \geqslant c/n$)

{

dispatch s'_i to G_n; $c = c - s'_i$; $i = i + 1$; $n = n - 1$;

if($n = = 1$) break;

}

if ($n = = 1$){

dispatch $s'_i + s'_{i+1} + \cdots + s'_k$ to G_1;

exit;

}

$t = 0$;

while ($n > 1$){

while ($s'_k < (c - t)/n$) {$s'_k = s'_k + s'_{k-1}$; $k = k - 1$; }

dispatch s'_k to G_n; $t = s'_k - c/n$; $n = n - 1$; $c = c - s'_k$;

}

dispatch $s'_i + s'_{i+1} + \cdots + s'_{k-1}$ to G_n;

In the above algorithm, t is the compensation factor. When a dispatch number exceeds the average number, t is positive, and the next dispatch number becomes slightly smaller, and vice versa. Apparently, t assures the balance of the dispatch number efficiently. In the rear of the algorithm, n's value is very important. If n is not equal to 1, the distribution is incomplete evidently. In fact, as long as k is greater than or equal to n, n will be equal to 1 in the end. Since k is related to the number of web stations accessed, it is certain that k is greater than or equal to n.

2.3 Implementation of group routing table

Because the data of the distribution algorithm are derived from the web-access history, it is difficult to contain all the later accessed DNs. Dispatch results of the above-mentioned algorithm applicable to all the DNs by expanding the later DNab. And the expanding method is as follows: ① Sorting out all the DNabs in dictionary order; ② Expanding DNab's area in a way in which the front is open interval, and the rear is close interval; ③ Combining vicinity area within a cache group. For example, there are k DNabs which are sorted according to the dictionary order — $N_1, N_2, \cdots, N_k$. And N_1 expands to $(0, N_1)$; N_2 expands to (N_1, N_2); $\cdots$; N_k expands to (N_{k-1}, ∞) (N_i means a string, N_k is the last DNab).

After the calculation of the relevant DNabs' area of each cache group through the distribution algorithm and the interval-continuum, the group routing table is established. The table contains the continuous character interval of DNs, the corresponding group number, and the destination IP address for each group. The destination IP address is retrieved using the Internet control message protocol (ICMP) to detect all IP addresses in a cache group and selecting the shortest reply time while the whole web-cache system is initialized. When the system's configuration changes, the destination IP address will be redetected and reconfigured. Each cache keeps a group routing table, so the group routing table's size should be considered. The group routing table's size is relative to the number of DNs' intervals combined, so it is less than the DN's number in the distribution algorithm. In fact, it is not necessary to consider all the DNs accessed in the distribution algorithm. Because DNs rarely accessed do not actually belong to a DN's interval, we only consider the front DNs accessed, which account for 90% of the total DNs after sorting out all the DNs accessed in descending order. We find that the front 110 DNs accessed frequently can meet the above condition by analyzing the web-access log of a college in a week. Fig. 1 is a DNs accessed frequency distribution graph of the college in a week, which only illustrates the top 20 of the total 2 770 DNs. And the accesses of the top 20 is 44% of the the total.

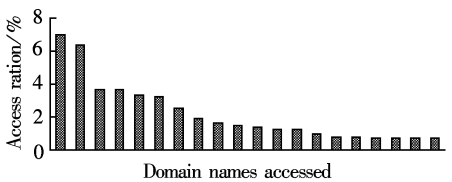

Fig.1 DNs accessed frequency distribution graph of a college

With the above-mentioned process, the group routing table contains the destination IP addresses for all the URLs. Therefore, when a client sends the request queried in the nearby proxy cache, the corresponding cache-group and destination IP address will certainly be found through querying the group routing table. In succession, the request is forwarded to the corresponding cache-group's proxy cache. In the end, the cache directory table within the group is checked to see if the request can be found.

2.4 Summary cache

In the cache group, we adopt a summary cache strategy[7] with each proxy cache keeping a compact summary of the cache directory of every other proxy cache. When a user request is forwarded to the relevant cache group, the proxy cache receiving the request checks the stored summaries to see if the request may be stored in proxy caches within this group. If it appears so, the proxy sends out the request to the relevant proxies to fetch the documents, otherwise the

proxy sends the request directly to the web server.

In the summary cache, it is not necessary to update the summary every time when the cache directory is changed; the update can occur at regular time intervals or when a certain percentage of the cached documents are not reflected in the summary instead. By this approach, two kinds of errors are tolerated: ① False misses: The document requested is cached at some other proxy but its summary does not reflect the fact. ② False hits: The document requested is not cached at some other proxy but its summary indicates that it is. The errors affect the total cache hit ratio or the interproxy traffic, but do not affect the correctness of the caching scheme. Apparently, the more proxy caches a group contains, the more the system's performance is affected by the interproxy traffic. So it is necessary to limit the caches' number within a group.

3 Simulation

To examine the benefits of several kinds of distributed web caches under a fixed cache size, we design a web-model. By this model, we simulate the following existing web cache schemes:

- Non-sharing cache Proxies do not collaborate to serve each other's cache misses.
- Simple cache sharing(SCS) Proxies serve each other's cache misses. Once a proxy fetches a document from another proxy, it caches the document locally. The sharing is implemented by the ICP protocol.
- Global cache including hash-partitioning(GCH) Proxies share cache contents and coordinate replacement so that they appear as one unified cache with global replacement to the users.
- Summary cache (SC) Each proxy cache keeps a compact summary of the cache directory of every other proxy cache.

In the meantime, we also simulate the proposed **grouping cache(GC).**

3.1 An introduction to the web-model

Since the focuses of the research are mainly on web cache's architecture and strategy, the design of the web-model concentrates on the forwarding of the web request, bandwidth and latency of the network transmission, transmission path of the web request and implementation of the cache. Fig. 2 shows the web-model graph. In the test model, we simulate router, switch, server, transmission line, proxy cache and client. The web-model mainly simulates the processes in which the web-request on HTTP is sent, forwarded and responded through the Internet. So the model focuses on the implementation of the functions, rather than the details of networks. The model's work contains the following steps:

① The client sends the request analogous to HTTP, which contains an address of accessed web-server and a requested file's path and name. Because the DNS servers are not simulated, the address of the web-server is given directly.

② The request is transmitted by the line, the switch and the router, and queried in proxy caches according to the web cache strategy. If the request ishit, the data are directly passed to the client, otherwise step ③ is implemented.

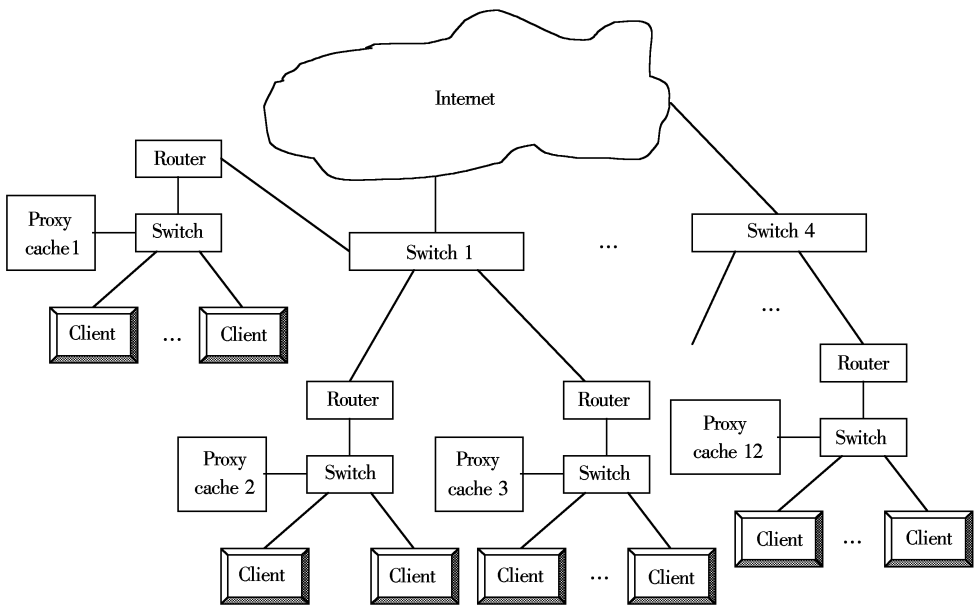

Fig.2 Schematic drawing of web-model

③ The request is forwarded to the web-server via the Internet. In fact, in the model the Internet is composed of several routers and lines. In the course of forwarding the request, a virtual path is constructed, and after the web server responds to the request, the data are passed to the client and the relevant proxy cache through the virtual path.

3.2 Design of web-model and user access model

The router and the proxy cache are important parts of the web model. The router's functions mainly contain logical connections, routing computing and data packet forwarding. To simplify the routing algorithm and keep data transmission stable in the test, a static routing protocol is adopted in the model. The proxy cache's functions comprise storing the recently requested data, adopting the relevant replacement strategy when caches are full, and disposing the web-request's query. Because we have not researched the performance of the single cache, we adopt first in first out (FIFO) replacement strategy in the design of the proxy caches, not considering the cache's prefetch technology. In addition, we place three web servers which receive the requests from the Internet, and pass data in accordance with the parameters of the requests. The bandwidth and latency of the network communication are both important factors affecting web cache performance, which are also simulated in the design of the communication line.

Fig.2 only shows the parts which are relative to proxy caches. There are four first-level switches, namely switch 1 to switch 4, and each switch links three proxy caches. The system has twelve proxy caches in total. The three web-servers and the Internet's interior structure are not displayed.

In the test, the clients send requests for web documents incessantly. These requests are generated by a user access model, which was developed by Simon and Van Wormer. And the model is formalized as follows[9].

Let $y_j(k)$ be the indicator of the access of the j-th document during the k-th time interval, where $y_j(k)$ is either 1 or 0 (1 means that the document is accessed, and 0 means that the document is not accessed). Then the total number of accesses of the j-th document at the end of the k-th time interval is simply $\sum_{\tau=1}^{k} y_j(\tau)$. The probability of accessing the j-th document at the $(k+1)$-th access can be formalized as

$$p[y_j(k+1) = 1] = \frac{1}{W_k}\sum_{\tau=1}^{k} y_j(\tau)\gamma^{k-\tau}$$

where W_k is the sum weighted usage of all documents, which is a function of time k and is the same for all documents; γ is the parameter that determines how rapidly the influence of past accesses on a new selection dies out[9]. The above formula only provides the method of computing the probability of accessing the old document. To determine the probability of accessing the new document, the user access model uses parameter α, the value of which is approximately equal to R/T, where R is the total number of different documents accessed and T is the total number of documents.

When implementing the user access model s, which is a stochastic number between 0 and 1, is regarded as the sign of accessing the new document. If s is less than α, a new document is accessed, otherwise the old is accessed. In the test, we suppose that $\alpha = 0.65$, and $\gamma = 0.99$.

3.3 Test results evaluation

The test model configures 12 proxy caches with 12 clients of sending requests corresponding to them, and each proxy cache's capacity is 8 MB. In the test, each client sends 10^4 requests continually with the above user access model. The size of each request's web document is 8 kB, which can be found in web servers. To analyze the performance and efficiency of the shared cache, we test the response time and the hit rate of the no cache, the non-sharing cache and the four kinds of shared cache. It takes 381 s to finish the total 12×10^4 requests in the no cache, and takes 330 s in the non-sharing cache (hit rate is 16.9%). The shared cache strategies' configuration is as follows: ① Three contiguous caches in network topology are shared in SCS, GCH and SC; ② Six contiguous caches in network topology are shared in SCS, GCH and SC; ③ Twelve caches are shared in SCS, GCH and SC; ④ In GC, all the caches are shared in the 3-grouping and the 6-grouping.

Fig.3 shows the response time and the hit ratio in the above configurations of shared cache. From Fig.3, we can see that all cache sharing schemes significantly improve the hit ratio and reduce the response time over the non-sharing cache. In general, the more the web caches are shared, the higher performance they can gain. But owing to the cost of multi-cache query, communication and management, the improvement of the hit ratio is not in proportion to the total capacity of the caches shared. Secondly, the overhead of SC is the least in three existing cache strategies. The response time of SC is less than SCS, even when the hit ratio of the former is slightly lower than the latter. The overhead of GCH is very much larger, especially when the shared caches increase. Thirdly, the performance of

GC is higher, when compared with three existing cache strategies. Because the GC dispatches web documents to every group according to their URL, once the client sends a request, the cache group in which the request should be queried is determined. The GC strategy not only avoids the cost of invalid query in the broadcast probe, but also solves the bottleneck problem of the centralized web cache.

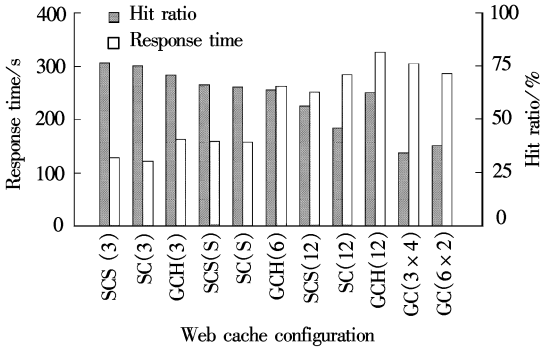

Fig.3 Response time and hit ratio under different shared cache configurations

4 Conclusion

In the proposed grouping cache strategy which adopts the technique of routing according to the URL among different groups, each shared proxy cache contains a group routing table and a compact summary of the cache directory within a group. The group routing table is responsible for selecting a proper group, and the compact summary of the cache directory determines if a request exists in a cache within the group. When the number of the cooperative proxy caches increases, the grouping cache can greatly enhance the web access efficiency. With the well-designed web-model and the user access model, we can demonstrate the benefits of the grouping cache **over the existing shared cache.**

References

[1] Sun H, Zang X, Trivedi K S. The effect of Web caching on network planning [J]. *Computer Communication*, **1999, 22**(14): 1343－1350.

[2] Barish G, Obraczka K. World wide web caching: trends and techniques [J]. *IEEE Communications Magazine*, **2000, 38**(5): 178－185.

[3] Michal K, Wojtek S, Adam W. A distributed WWW cache [J]. *Computer Networks and ISDN Systems*, **1998, 30**(22): 2261－2267.

[4] Michael R, Jeff C, Syam G. Not all hits are created equal: cooperative proxy caching over a wide-area network[J]. *Computer Networks and ISDN Systems*, **1998, 30**(22): 2253－2259.

[5] Selvakumar S, Prabhakar P. Implementation and comparison of distributed caching schemes [J]. *Computer Communication*, **2001, 24**(7): 677－684.

[6] Gadde S, Rabinovich M, Chase J. Reduce, reuse, recycle: an approach to building large internet caches [A]. In: *The Sixth Workshop on Hot Topics in Operating Systems* [C]. 1997. 93－98.

[7] Fan L, Cao P, Almeida J, et al. Summary cache: a scalabe wide-area web cache sharing protocol [J]. *IEEE/ACM Transactions on Network*, **2000, 8**(3): 281－293.

[8] Xu Y F, Ye J C, Chi X B. An approximate algorithm for optimizing task assignment in a multiprocessor system [J]. *Journal of Xi'an Jiaotong University*, **1999, 33**(4): 98－101. (in Chinese)

[9] Watson E F, Shi Y, Chen Y. A user-access model-driven approach to proxy cache performance [J]. *Decision Support Systems*, **1999, 25**(4): 309－338.

一种新的 Web Cache 技术的研究及实现

易法令[1,2] 谢长生[1] 韩德志[1] 蔡 斌[1]

([1] 华中科技大学计算机科学与技术学院, 武汉 430074)
([2] 长江大学计算机科学学院, 荆州 434043)

摘要: 提出了一种新的网络共享 Cache 技术——分组 Cache, 通过把所有的共享 Cache 按其在网络中的位置进行分组, 该方法能够有效地解决多 Cache 组织查询效率不高和 I/O 瓶颈等问题. 该技术有以下特点: ①采用分组技术能够有效地降低多 Cache 间的查询开销; ② 组内的 Cache 数据目录表能够很快确定请求的内容是否在某个 Cache 中; ③根据网络访问日志确定的分配算法能够较好地平衡各 Cache 组的负载. 模拟测试证明: 分组 Cache 技术与其他已有的共享 Cache 技术相比有较高的效率.

关键词: web 缓存; 共享缓存; 分配算法; 分组缓存

中图分类号: TP393