Performance Study of Satellite-linked Web Caches and Filtering Policies

Xiao-Yu Hu Pablo Rodriguez * Ernst W. Biersack

Institut EURECOM, Sophia Antipolis, France.
{rodrigue,erbi}@eurecom.fr

Abstract. The exponential growth of World-Wide Web (WWW) requires new and more efficient content distribution mechanisms. Web caching has shown to be a very efficient way to scale the WWW by reducing traffic in congested network links, decreasing latency to the clients, and reducing load in the origin servers. In this paper we study an emerging caching strategy called cache-satellite distribution. In a cache-satellite distribution caches are inter-connected via a satellite channel, effectively increasing the client population connected to a cache. Using Markov chain analysis, we demonstrate that the higher the number of clients connected to a cache-satellite distribution, the higher the probability that a document request is hit in the cache. Our analytical results strongly advocate a large scale interconnection and cooperation of ISP-caches via a satellite distribution.

Due to the large amount of documents distributed through the satellite and the limited disk capacity and processing power of local ISP caches, it is impossible to store all documents coming from the satellite distribution. Therefore, it becomes necessary for an ISP-cache connected to the satellite distribution to perform efficient filtering policies. Based on the stability in the behavior of the clients of an ISP cache, we propose novel filtering policies which rely on the Web servers visited on the previous days. Using trace-driven simulation, we study different filtering policies and show how simple filtering policies can exhibit excellent performance in rejecting non-desired documents while assuring high hit rates.

Key Words: World-Wide Web, Web Caching, Satellite Distribution, Filtering Policies.

1 Introduction

The Web today is characterized by high volume of accesses to popular Web pages. Thus, identical copies of many documents pass through the same congested network links. As a result, network administrators see a growing utilization of their network that requires upgrading their links or replacing their servers and end users experience longer and longer latencies to retrieve a document. These problems can be alleviated by widespread migration of copies of popular documents from servers to points closer to the users [2]: caching at the origin server, caching at the client (e.g., caches built into Web browsers), and most importantly, proxy caching servers inside the network, also called Web caches.

World-Wide Web caches can potentially reduce the number of requests that reach popular servers, the volume of network traffic resulting from document requests, and the latency that an end-user experiences in retrieving a document. However, caches offer limited performance since there is a large percentage of requests that are not satisfied by the cache. Requests satisfied in the cache are called *hits*. Requests not satisfied in the cache are called *misses*. Misses can be classified into: 1) *First-Access*: misses occurring when requesting documents for the first time. 2) *Capacity*: misses occurring when accessing documents previously requested but discarded from the cache due to space limitations. 3) *Updates*: misses occurring when accessing documents previously requested but already expired. 4) *Non-cacheable*: misses occurring when accessing documents that need to be delivered from the origin server (e.g. dynamic documents generated from cgi-bin scripts)

Even when a cache has an infinite storage capacity, the number of misses in an institutional cache can be very high ($50-70\%$) [2]. While non-cacheable documents typically do not account for more than 10% of all requests [14] and update misses do not account for more than 9% of all requests, there is a large percentage of requests that result in first-access misses (30-50%) [14]. One way to reduce the number of first-access misses and update misses is to prefetch the cache; documents are pushed into the cache even if the cache has never requested them. The idea is to get documents in the cache expecting that clients will likely request them. When documents are prefetched through the Internet and no client requests them, the bandwidth waste is considerable. An emerging alternative to prefetch or push web documents into caches is using a satellite distribution [13] [4] [10]. A satellite distribution has fewer packet losses and congestion problems than a distribution in the Internet. Also, a satellite distribution can reach a very large population with relatively little effort; adding a new additional client does not increase the cost of transmission.

The principle of pushing popular Web documents into caches via a satellite is quite simple: every requested document that results in a miss in any ISP-cache, is automatically broadcasted by a *master distribution center* to all the other ISP caches

* Pablo Rodriguez is supported by the European Commission in form of a TMR (Training and Mobility for Researchers) fellowship. EURECOM's research is partially supported by its industrial partners: Ascom, Cegetel, France Telecom, Hitachi, IBM France, Motorola, Swisscom, Texas Instruments, and Thomson CSF.

through a satellite channel. As the number of clients in all caches interconnected by the satellite continually increases, the probability that a specific client is the first requesting a certain document steadily decreases, thereby obtaining higher hit rates. Recently several companies like Sky Cache [13] and Edgix [7] have started to offer this kind of service.

In this paper first we develop a discrete time-evolving model to describe the hit-or-miss behavior of the requested object, enabling a very simple numerical expression for the hit rate (HR). By means of this explicit HR formula, we can easily determine the hit rate for a document as a function of the client population, and we can also study filtering policies at the master distribution center. Using trace-driven simulations we show how the hit rate of an ISP can be easily improved when it gets connected to a satellite distribution where many clients share the same caches.

Second, we study different filtering policies at the ISP-cache side. As more and more clients get connected into the satellite caching distribution, a large number of documents are to be stored at each ISP-cache, which might overflow the cache storage capacity. Therefore, there is a need for efficient filtering techniques that block non-desired documents coming from the satellite link into the ISP caches. Based on the assumption that the ISP clients' interest do not change dramatically on a day-by-day basis, we propose a novel filtering policy which relies on the Web severs visited on the previous day. The decision rule is simple: if an incoming document matches the filtering database with a list of Web servers visited on the previous day, then store the document for a possible future hit; otherwise just discard it. The philosophy behind this filtering policy is that if a Web server was visited by local clients on the previous day, then this Web server is very likely to be requested again by a local client. We also consider other filtering policies where only those documents coming from previously visited Web sites with more than a certain number of requests are considered. Using trace-driven simulation, we study different filtering policies and show that simple filtering policies can exhibit excellent performance in blocking non-desired documents while retaining high hit rates.

2 Internet Topology: An ISP Perspective

At its current stage, the Internet connecting the server and the clients can be modeled as a hierarchy of ISPs, with each ISP having its own autonomous administration. Without loss of generality, we can make a reasonable assumption that the Internet hierarchy consists of three levels of ISPs: local, regional, and national, as shown in Figure 1. All the clients are connected to the institutional ISP, which might serve a corporation or university. The local ISP is usually characterized by local area coverage and thus a high speed bandwidth. The local ISPs are connected to the regional networks and the regional networks are upwardly connected to the national ISP. In hierarchical caching, caches are usually placed at the access points between two different networks to reduce the cost of transmitting across a new network [5]. One popular protocol which allows Web clients to coordinate and share a hierarchy of Web caches is the Internet Caching Protocol (ICP) [15]. However, the hierarchical topology and ICP have several drawbacks: First, the disk space utilization is relatively low due to *mirroring effect*— only a fraction of the total storage capacity contains unique objects. Second, if the document is not hit in the cache, additional latency is added when searching the documents among the siblings and traversing the caching hierarchy. Third, higher levels of the caching hierarchy may easily become highly congested and add considerably delays [11].

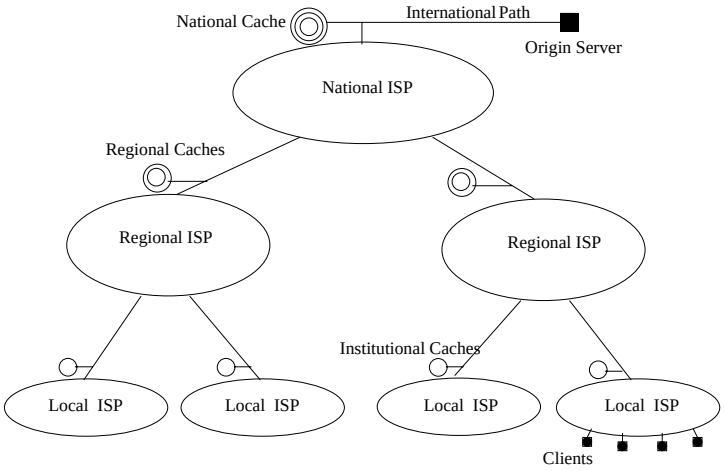

Fig. 1. Internet topology: hierarchical structure.

Recently, an emerging cache pushing mechanism called cache-satellite distribution is receiving more and more attention [10]. The topology structure for a cache-satellite distribution is shown in Figure 2. Since satellite links usually have broad bandwidth and cover a large scale of geographical areas, there is no need for a caching hierarchy, which aims at creating a large effective client community. Compared with hierarchical caching, a satellite distribution has the ability to cover more clients with less cost.

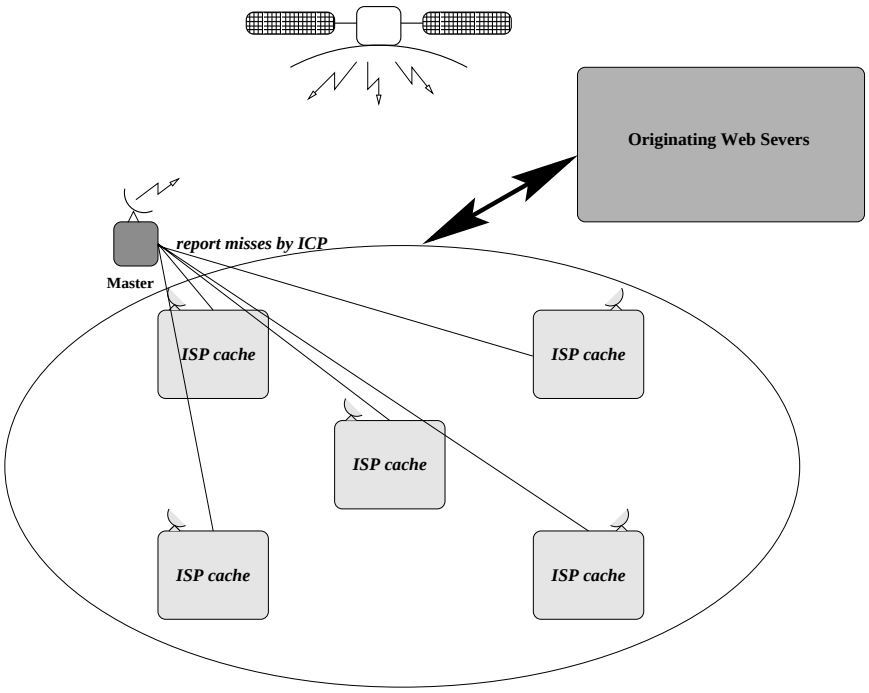

Fig. 2. Satellite distribution.

The satellite caching distribution works as follows. Whenever there is a miss at an ISP cache, the ISP cache obtains the document from the origin Web server via HTTP protocol. The ISP cache reports the missed URL (Universal Resource Location) to a master site via ICP [15]. The master site then obtains the document from the origin server and transmits the document into the satellite channel. As a result, all ISP caches receive the broadcasted document and can decide to keep it or not. Each ISP cache receives all the documents requested by any client connected to the satellite caching distribution, therefore, the probability that a specific client is the first client requesting a document is very small and the hit rate increases.

3 Analytical Model and Performance Study

In this section we present analytical models to derive an explicit hit rate expression depending on the client community connected to the cache, and the life-time of a document. Assume that we have totally N clients covered by satellite distribution and that each ISP cache has infinite disk space. Let λ_i be the request rate of a single user for a *cacheable* document, where $i = 1, 2, \cdots, N$. Assume that the request rate for a single document is Poisson distributed [9], therefore, the total request rate λ for a document is

$$\lambda = \sum_{i=1}^{N} \lambda_i \tag{1}$$

which is also a Poisson process with mean λ.

We define the mean life time duration of a document in a Web cache by the term *residential time* T_s which depends on several factors: estimated time-to-live (TTL), modification cycle, and filter/removing policy. We assume that the residential time is exponentially distributed with mean T_s [6].

Now, we define an arbitrary small time-slot τ, which tends to zero, and denote T_M, T_{M-1}, $\cdots$, T_1 as the remaining residential time of a specific document in a Web cache after a time τ, where $M = T_s/\tau$ (see Figure 3). When the residential time T_s expires, the state T_0 is entered again. Therefore, the probability to enter state T_M is given by $p = \tau\lambda$.

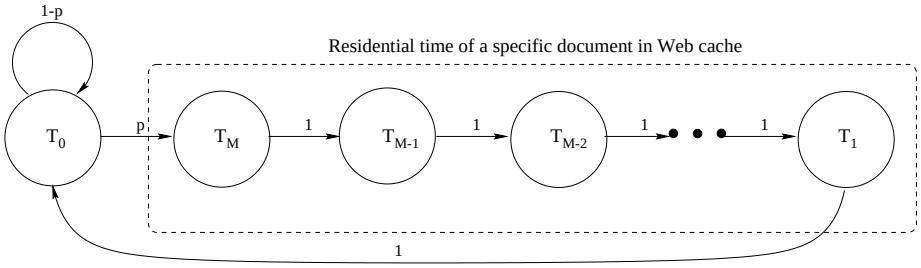

Fig. 3. Markov chain model.

Denote the steady-state probability to be in state T_n by $\pi(T_n)$. From the Markov chain, we know that

$$\begin{cases} \pi(T_0) &= \pi(T_1) + (1-p)\pi(T_0) \\ \pi(T_M) &= p\pi(T_0) \\ \pi(T_{M-1}) &= \pi(T_M) \\ \pi(T_{M-2}) &= \pi(T_{M-1}) \\ & \cdots \\ \pi(T_1) &= \pi(T_2) \end{cases}$$

and furthermore $\sum_n \pi(T_n) = 1$.

Solving the equations above, we obtain that $\pi(T_M) = \pi(T_{M-1}) = \pi(T_{M-2}) = \cdots = \pi(T_1)$, and $Mp\pi(T_0) + \pi(T_0) = 1$, which leads to

$$\pi(T_0) = \frac{1}{1+pM} \tag{2}$$

Substituting $M = T_s/\tau$ and $p = \tau\lambda$ into equation 2 yields $\pi(T_0) = \frac{1}{1+\lambda T_s}$. The probability that a miss occurs is the probability that a request finds a document in state T_0. Therefore the miss rate (MR) is equal to $\pi(T_0)$, i.e. $MR = \pi(T_0) = \frac{1}{1+\lambda T_s}$ and the hit rate (HR) for a given document is

$$HR = 1 - MR = \frac{\lambda T_s}{1+\lambda T_s} \tag{3}$$

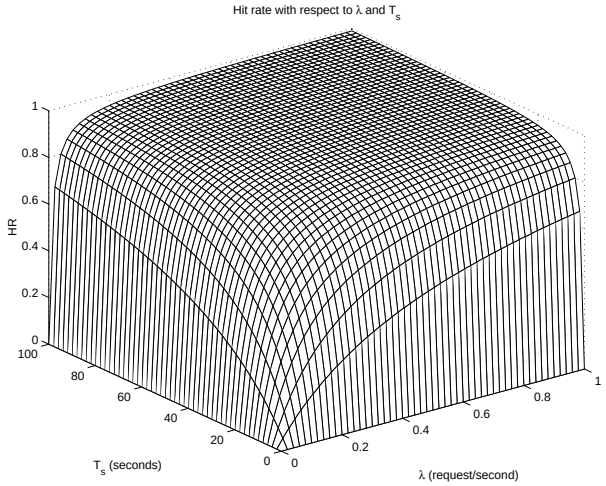

Fig. 4. Hit rate with respect to λ and T_s

In Figure 4 we plot the hit rate HR with respect to λ and T_s. From Figure 4 we see that for a specific cacheable document, as λ or T_s increases, the HR tends to 100%. Therefore, the larger the number of clients connected to satellite distribution, or the larger the residential time of a document T_s, the higher the probability that a document requested by a client is hit in the Web cache.

4 Filtering Policies

It is worth pointing out that the results obtained in equation 3 assume that the satellite bandwidth and the cache disk storage are infinity. Practically, the satellite bandwidth and the cache's storage capacity are limited, thus, filtering or removing policies are needed. One simple filtering policy at the master side could be to avoid sending those documents that are not very popular. Thus, the master site of a satellite distribution would simply set up a threshold TSH on the number of requests per update period λT_s and distribute through the satellite only those documents with $\lambda T_s >= TSH$ requests. This, simple filtering policy at the master side can keep the overall hit rate high, while making the bandwidth and disk space requirements affordable [10].

At the cache side (i.e. ISP caches), it may be difficult or impossible to obtain (or estimate) the value of λT_s. Also, clients at a given cache may be interested in a different set of documents than clients at a different cache, therefore, cache side filtering policies should be implemented. Good filtering policies at the cache side would require to block most of the unnecessary documents or documents that are unlikely to be hit in the future, while retaining a high hit rate. For instance, if most of our clients are college students, it might be necessary to keep all the documents related to *.edu*; and if most of our clients speak only Chinese, it may be reasonable to discard all documents coming from non-Chinese, non-English speaking country. Of course this kind of prior information is very helpful to establish an efficient filtering policy. However, in this paper we assume no such prior knowledge on the background of clients; we consider a more general case: only the history of the requests of local clients is available in the form of caching server's logs.

Compared with the fast changing rate of the Internet and its documents, the relish and interest of each member of the local users (clients) usually remains stable, at least for a short period, and so does the membership of the audience itself as a whole. Accordingly, the requesting pattern from all ISPs could also be thought to be relatively stable. Based on this assumption, we suggest a *generic filtering policy* which depends solely on the previous day's request pattern at each local site. For a specific day, we first summarize the total Web servers that were visited on the previous day and build a database with each entry representing an origin Web server. Then, for each document coming from the satellite the ISP's cache decides whether to accept it or not by comparing the host name of the document's URL with the filtering database. If there exists a match, the document is stored into the cache for possible future hits; otherwise, it is simply discarded. The philosophy behind this filtering policy is that if a Web server was visited by local clients on the previous day, then it is very likely to be requested again by local clients.

We then extend the proposed filtering policy further to take into account the popularity of a document. Based on the popularity of a Web server, we can set up different thresholds to determine which documents to keep. We call this method a *threshold filtering policy*.

Next, we present via trace-driven simulations the performance of a generic and a threshold filtering policy in terms of disk space usage, hit rate (HR), and weighted hit rate(WHR) – the fraction of client-requested bytes returned by the web cache. We use the access logs provided by an ISP in the USA (AYE [1]). This ISP gives access to about 1000 residential and business clients and has a cache with 48 GBytes of disk space. The logs of AYE's cache are from Dec. 18-23, 1998, which account for nearly 10 million requests. We considered all documents, cacheable and uncacheable, to study different filtering policies regardless of the cacheability of a document (non-cacheable documents account for about 10% of all document requests).

5 Filtering Policies: Trace-driven Simulation

5.1 Original hit rate and weighted hit rate

In order to measure the influence of the filtering policies on the hit rate, we first calculate the original hit rate without the use of the filtering policies. To calculate the hit rate at AYE's cache we considered *all* hits, including those hits that needed a consistency check (if-modified-since) with the origin server and resulted in a document not-modified [8]. From Figure 5 we can see that the average hit rate HR of AYE's cache is about 45% and the average weight hit rate WHR is 30%. Both the HR, and the WHR are quite low since most of the document requests result in a miss in the cache. Misses are mostly due to requests for documents that no other client has ever requested before (first-access misses). The significant difference between HR and WHR is because Web caches usually discard large documents to help keeping high HR [12].

5.2 Impacts of a generic filtering policy on the hit rate

In Figure 6 we consider a generic filtering policy where ISP caches only keep those documents which host name matches that of any Web site visited during the previous day. Comparing Figures 5 and 6, we see that only a quite negligible reduction in the HR and the WHR has been caused by the generic filtering policy. For example, on Dec. 19, the corresponding HR reduction is only 4.3%, on Dec. 20 only 3.0%, on Dec. 21 only 3.2%, and so forth.

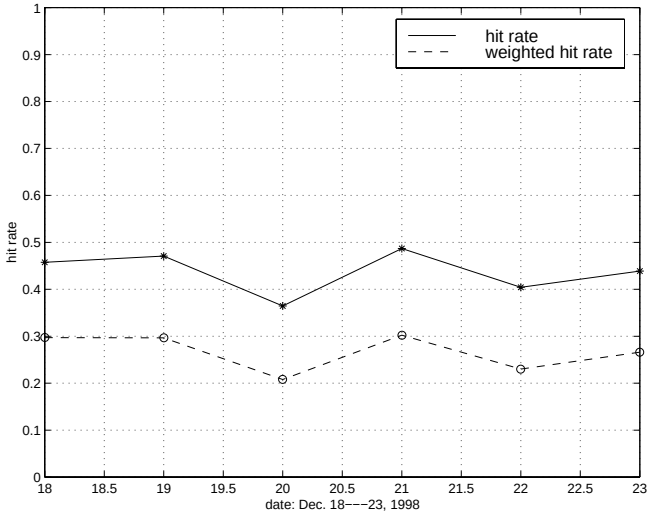

Fig. 5. The original HR and WHR of AYE's cache without the use of a filtering policy.

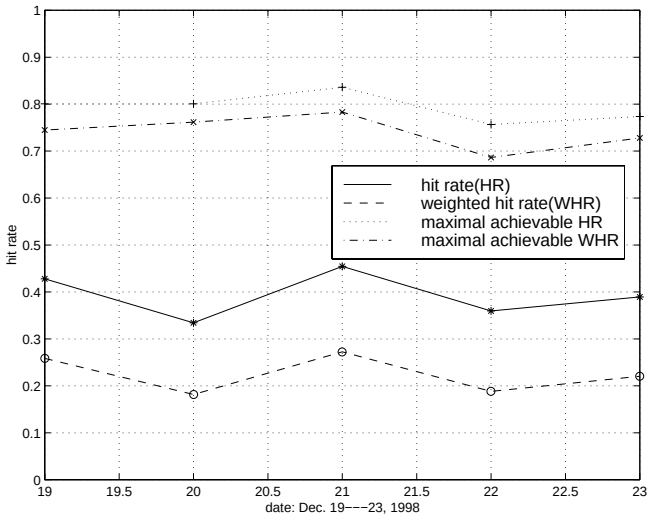

Fig. 6. HR and WHR of AYE's cache with a generic filtering policy based on those servers visited on the previous day.

In Figure 6 we also calculated the *maximal achievable hit rate* and the *maximal achievable weighted hit rate*, which is the HR and WHR achieved in an ideal scenario where all documents requested in the cache were previously prefetched in the cache. If the percentage of cacheable documents is 100% then, the maximal achievable HR equals 100%. In Figure 6 we show the maximal achievable HR and WHR for the generic filtering policy described before. We observe that the maximal achievable HR and WHR is as high as 80% and 75%, respectively. Therefore, the correlation between Web sites visited from one day to the following day is very high; there are only few new Web sites that are visited on one day and are not visited on the following day. The percentage of new Web sites that are visited every day and that were not visited before only accounts for about 20% of all requests.

Not surprisingly, there currently exists a significant gap between the maximal achievable HR and the measured HR, this is due to the fact that the client population connected to AYE's cache is not large enough. However, as we will see in Section 6 as more and more ISP Web caches get connected through a satellite distribution, this gap decreases rapidly.

Next, we have also considered another interesting case where we employ a unique database built on Dec. 18 to implement filtering policies for Dec. 19, 20, 21, 22, and 23 respectively. Thus, during the days Dec. 19-23, the cache only keeps those documents which host name matches that of a Web site visited on Dec 18. The results, shown in Figure 7, exhibit a surprising resemblance with Figure 6 where the filtering databases where built based on the immediate previous day. This results, strongly support the fact that the relish and interest of clients connected to an ISP remain stable on a day-by-day basis.

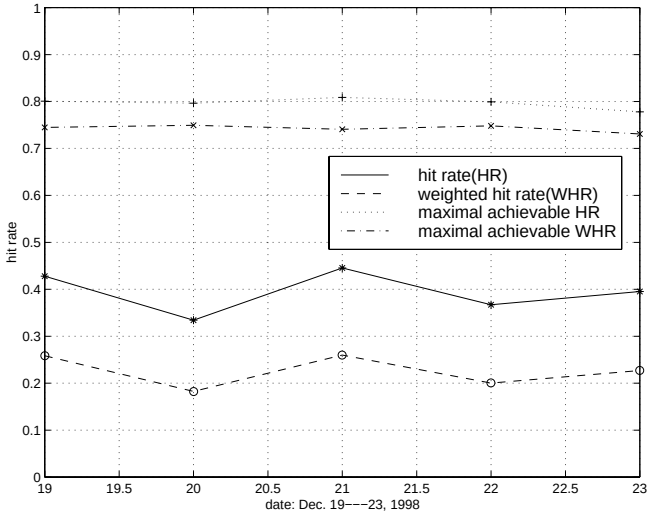

Fig. 7. HR and WHR of AYE's cache with filtering policy based on servers visited on a specific day: Dec. 18.

5.3 Performance of a threshold filtering policy

In this section we consider a threshold filtering policy. The only difference between a threshold filtering policy with the previous one (generic) is that all the entries in the database from a Web site with less than a certain number of requests during the previous day (i.e. threshold) are removed. That is to say, if a Web server was visited less than threshold times on the previous day, it means this server is not a "hot" site, and therefore the cache decides not to store the documents coming from it. In this way, we can readily reduce the number of entries used to implement the filtering policy, thereby, alleviating the demands for processing power and disk space. However, reducing the number of Web sites for which the cache keeps documents may decrease the hit rate. The goal is to reduce the number of Web sites for which the cache keeps documents while retaining the HR and the WHR at an acceptable level.

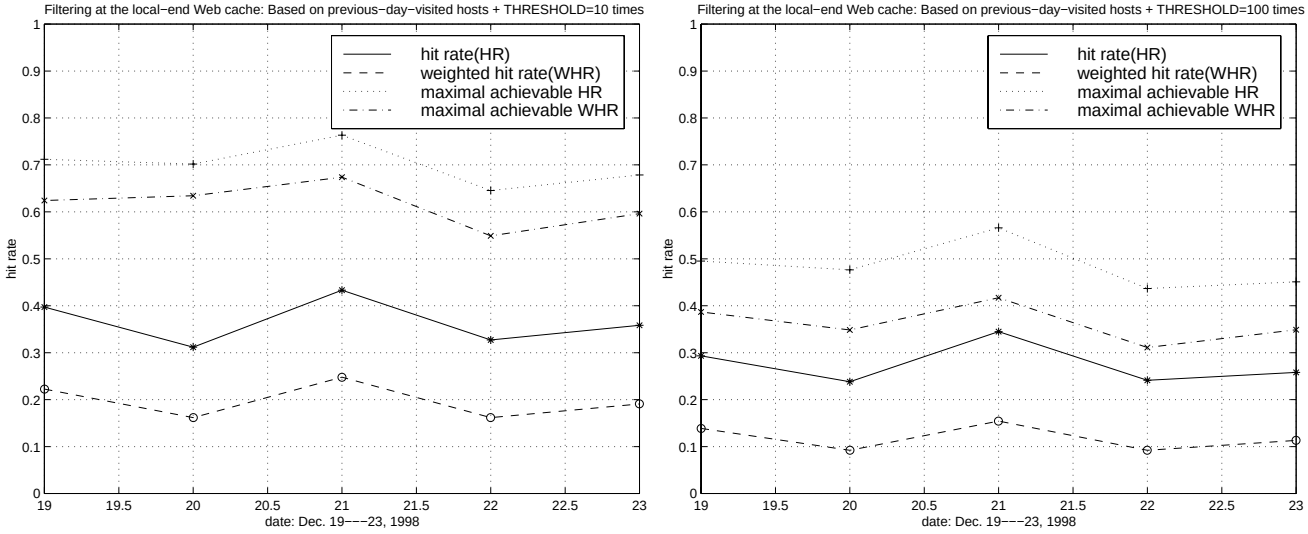

Fig. 8. Performance of threshold filtering policy with thresholds equal 10 req/day, and 100 req/day, respectively.

In Figure 8 we show the HR and the WHR for different threshold filtering policies based on previous day logs. Moreover, in Figure 9 we also plot the reduction in HR and WHR of each threshold filtering policy, when compared to the non-filtering case. The results in both figures show that setting a small threshold popularity, does not significantly decrease the hit rate. For instance, setting a threshold of 10 requests per day, the hit rate is only decreased by about 3% compared to the case where no

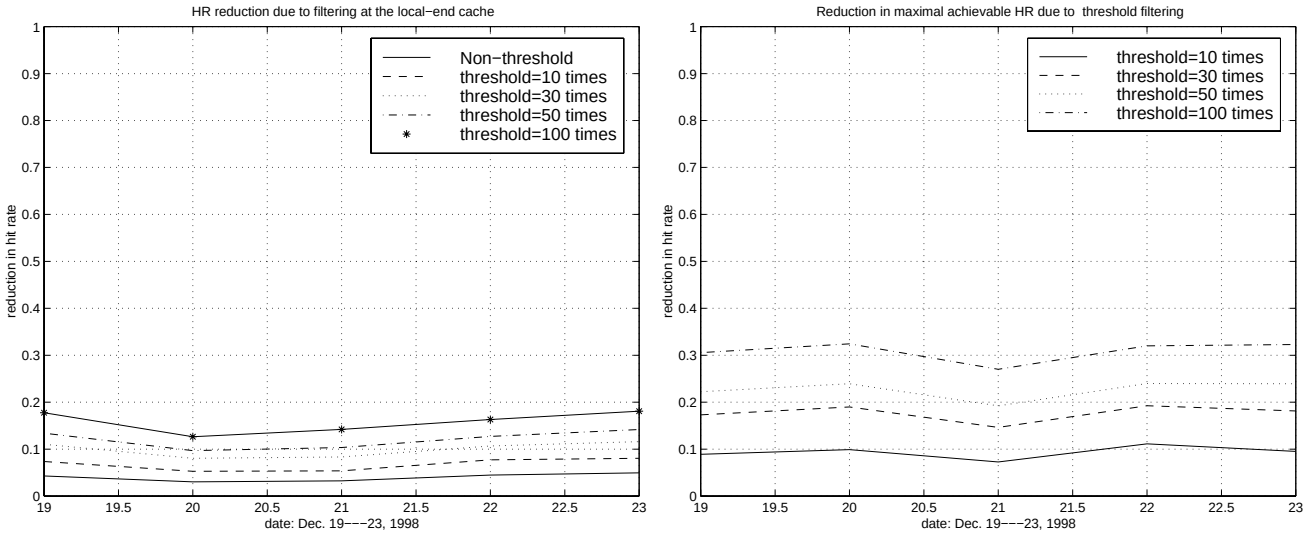

Fig. 9. Comparison of HR and WHR reduction due to different filtering policies.

threshold is used. The main reason for this results from the Zipf distribution of document requests [3] [16] where only few Web sites account for most of the requests and there is a large set of documents that have very few requests.

5.4 Complexity and disk requirements of filtering policies

Up to now we have considered the performance of generic filtering policies based on previous requesting patterns and threshold filtering policies with different thresholds. As we saw, the mentioned filtering policies only have a negligible (or tolerable) reduction in both HR and WHR. However, other issues arise with filtering policies, such as the computing requirements for executing such filtering policies, and the disk requirements to store the documents coming from satellite link depending on the filtering policies.

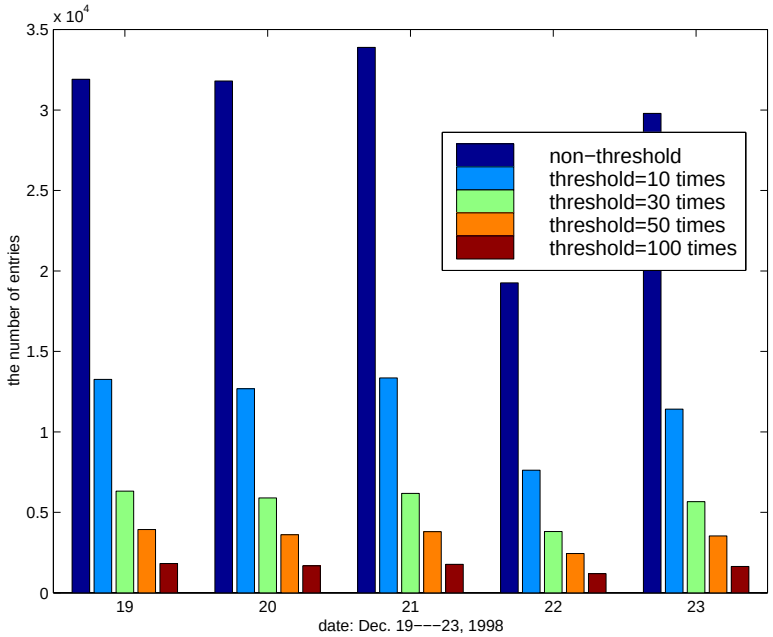

Fig. 10. The number of entries used in different filtering policies.

For each document coming from the satellite, it is necessary for a filtering policy to examine whether the document's host name matches one of the filtering database entries. Therefore, the execution complexity of the filtering policy depends on the number of entries in the database. Also, since all the documents with a host name equal to an entry in the database are stored in the cache, the number of entries in the database is directly related to the cache disk requirements. The more entries in the filtering database, the more the storage space is required. In Figure 10 we plot the number of entries needed for the proposed filtering policies. From Figure 10 we see that the number of entries decreases quickly as the threshold value increases, resulting in a drastic reduction in the execution complexity and the needed disk space. At the same time, from Figures 8 and 9 we also saw that the decrease in the hit rate was small even for large threshold values. For instance, on Dec. 19, the total number of entries used for a generic filtering policy (i.e. the non-threshold case) is 31913, and the hit rate is 42.8%; the threshold filtering policy with threshold=10 req/day uses 13266 entries, less than half of the generic policy, while still achieving a hit rate of 39.7%. Therefore, implementing threshold filtering policies helps reducing the execution complexity and disk requirements while keeping high hit rates.

5.5 Filtering policy based on consecutive days of access history

Next, we investigate how many days of access history are needed to build the filtering database. Obviously, using more days of history, the probability to hit a document will increase at the cost of a large filtering database and processing power. In Table 1 and 2 we present the increased hit rate and increased number of entries, respectively, as compared to those of filtering policy based solely on the previous day. We only considered the generic filtering policy, i.e., the threshold is set to zero.

Table 1. Increased hit rate based on consecutive days of history in the case of a generic filtering policy.

filtering database based on previous . . .	increased HR on Dec. 20	increased HR on Dec. 21	increased HR on Dec. 22	increased HR on Dec. 23
two days	1.2%	1.2%	2.6%	1.4%
three days	N/A	1.7%	3.1%	2.9%
four days	N/A	N/A	3.4%	3.5%
five days	N/A	N/A	N/A	3.8%

Table 2. Additional number of entries for consecutive days of history in the case of generic filtering policy.

filtering database based on previous . . .	increased entries for Dec. 20	increased entries for Dec. 21	increased entries for Dec. 22	increased entries for Dec. 23
two days	16995(53.4%)	16498(48.7%)	27389(142.2%)	9264(31.1%)
three days	N/A	29130(85.9%)	37981(197.2%)	26411(88.6%)
four days	N/A	N/A	49621(257.6%)	38678(129.8%)
five days	N/A	N/A	N/A	48769(163.7%)

From these two tables we see that increasing the number of days used to build the database, rapidly increases the number of database entries, while, it does not significantly improve the hit rate. For example, on Dec. 20 the filtering policy based on the two previous days has a 1.2% higher hit rate than that based only on the previous day, however, the number of entries in filtering database grows by 16995 items, which is a 53.4% more. As more days of history are employed, the increased hit rate is almost negligible, while the increase in the number of filtering entries is quite significant. Therefore, using only the previous day of logs to build the filtering database is a reasonable choice.

6 Case study: Filtering schemes on a cache-satellite distribution

Now, we proceed to study the situation where AYE's cache is connected to a cache-satellite distribution. As shown in Section 3, as more and more clients get connected to the cache-satellite distribution, the hit rate increases gradually, however, the disk space requirements will increase explosively if no filtering policy is employed. The primary goal of this section is to verify that the filtering policies presented before have the ability to keep most of the documents coming from the satellite that are of interest to the local clients, while blocking as many non-desired documents as possible.

For this purpose we obtain one day of logs from the top-level log of NLANR [1] cache on Dec. 23 and simulate a scenario where the NLANR top-level cache gets connected to cache-satellite distribution. We assume that AYE's cache is connected to the same satellite distribution. We consider that all objects requested in the NLANR log, whether hit or missed, are pushed over the satellite link and therefore received by AYE's cache. We are interested on the increased hit rate at AYE's cache due to satellite pushing, and on the additional required disk space at AYE's cache.

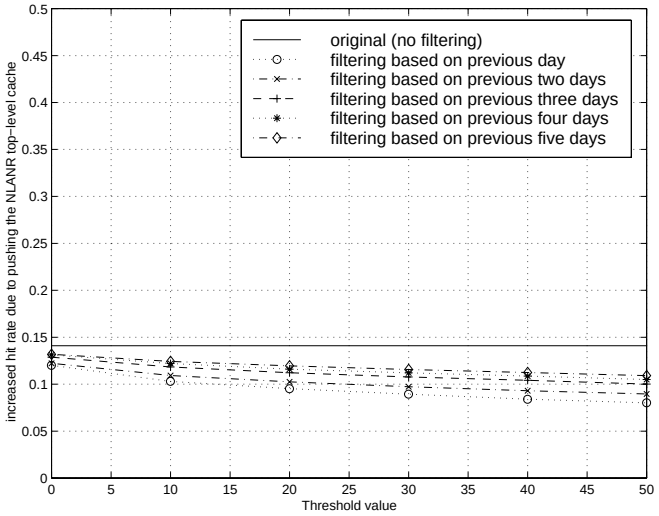

Fig. 11. Additional hit rate at AYE's cache due to pushing documents requested at the NLANR's top-level cache through the satellite.

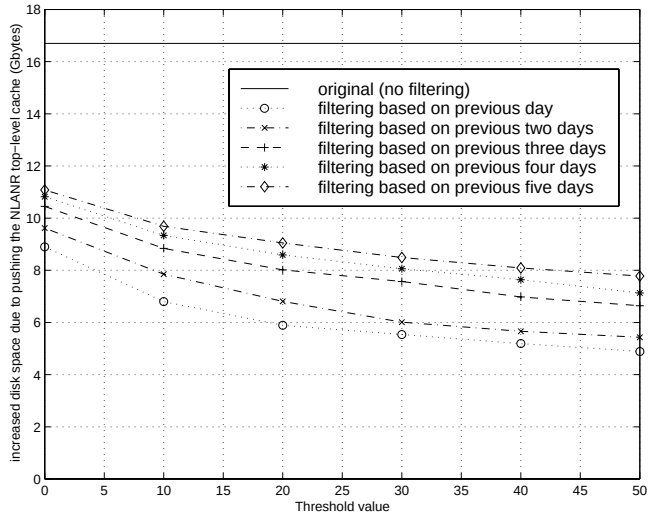

Fig. 12. Additional disk requirements at AYE's cache due to pushing documents requested at NLANR's top-level cache through the satellite.

In Figure 11 and Figure 12 we show the percentage increase of the hit rate for different filtering policies and the increase of the disk requirements. From these two figures we see that in the case that no filter is used, an additional 16.7 GBytes of disk space are required to store those documents coming from the satellite at AYE's cache, and the resulting hit rate increases by 14.1%. Using the generic filtering algorithm, i.e., keeping only the incoming documents that match Web servers visited on

[1] National Lab of Applied Network Research. http://ircache.nlanr.net. NLANR top-level cache consists of four major cooperating caches that share their load.

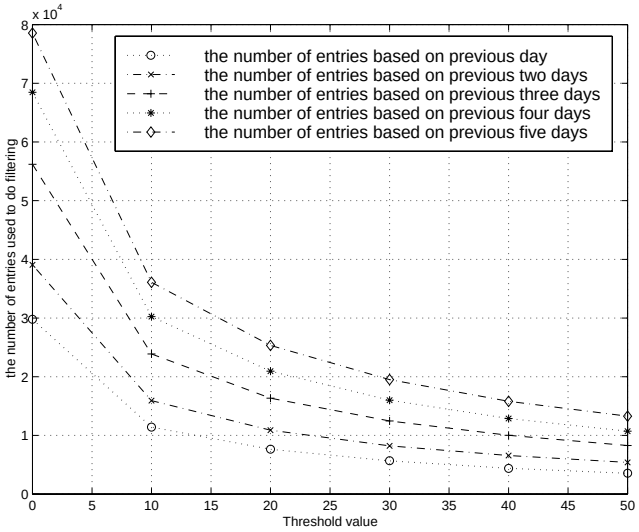

Fig. 13. Number of entries in the database used to filter documents coming from the satellite.

the previous day, the additional disk space requirement drops to 8.9 Gbytes while the increased hit rate is still about 12.0%. Therefore, the generic filtering policy achieves 46.2% disk-space saving at the cost of a negligible decrease in HR.

We further study the increased hit rate corresponding to threshold filtering policies based on several previous days. As shown in Figure 11, we find that, as more and more days' logs are incorporated into filtering database, the increase in the hit rate is quite negligible. On the other hand, in Figure 12 we see that as more days are considered, the needed disk space to store those documents coming from the satellite is significantly enlarged; a similar thing happens to the number of database entries (Figure 13). For instance, using the previous day of logs from AYE, the additional hit rate increases by 12.0%, AYE's cache needs additionally 8.9 Gbytes of disk space, and the database has 29793 filtering entries; based on the previous two days of logs, the increase in the hit rate is of 12.5%, however, the increase in disk space is 9.62 Gbytes and the number of entries is 39058, and so forth. Based on these figures, we find that the use of a generic filtering policy or a threshold filtering policy with small threshold values, e.g. 10 req/day, based solely on the previous day, yields to a good trade-off between disk space requirements to store Web documents, processing capacity to run the filtering policies, and hit rate.

7 Conclusions

A cache-satellite distribution is emerging as a very promising technology to alleviate the problems related to the rapid growth of the Web. In this paper, we have analyzed and evaluated the performance of cache-satellite distribution. Our theoretical and trace-driven results demonstrate that as the population of clients connected to the satellite distribution increases, the hit rate goes up steadily. Using trace-driven simulations we showed that connecting more clients to the satellite distribution scheme, ISP-caches can easily increase their hit rates by 15%.

As more and more clients (or ISPs) get connected to a cache-satellite distribution, a large number of documents is being distributed through the satellite, which might overflow the disk storage capacity of the caches and considerably increase their load. We have studied different filtering policies at the ISP caches, aiming at discarding documents unlikely to be requested. In particular we have considered filtering policies which take into account access patterns from clients connected to a certain cache. Using trace driven simulations, we show that simple filtering policies can greatly reduce the disk requirements of Web caches connected to a satellite distribution while reducing the hit rates only by about 2%-3%. In addition, filtering policies that include the site's popularity to decide whether to keep a document or not, further reduce the size of the filtering database while hardly modifying the obtained hit rates.

References

1. AYE, "http://www.aye.net/".
2. M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "World Wide Web Caching: The Application-Level View of the Internet", *IEEE Communications Magazine*, pp. 170–178, June 1997.
3. L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "On the Implications of Zipf's Law for Web Caching", In *Proceedings of IEEE INFOCOM'99*, New York, USA, March 1999.

4. Broadcast Satellite Services, "http://www.isp-sat.com".
5. A. Chankhunthod et al., "A Hierarchical Internet Object Cache", In *Proc. 1996 USENIX Technical Conference*, San Diego, CA, January 1996.
6. F. Douglis, A. Feldmann, B. Krishnamurthy, and J.Mogul, "Rate of change and other metrics: A live study of the World Wide Web", In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, December 1997.
7. Edgix, "http://www.edgix.com".
8. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, et al., "RFC 2068: Hypertext Transfer Protocol — HTTP/1.1", January 1997.
9. S. Gribble and E. Brewer, "System Design Issues for Internet Middleware Services: Deductions from a Large Client Trace", In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, December 1997.
10. P. Rodriguez and E. W. Biersack, "Bringing the Web to the Network Edge: Large Caches and Satellite Distribution", In *To appear in MONET. Special issue on Satellite-based information services*, January 2000.
11. P. Rodriguez, K. W. Ross, and E. W. Biersack, "Distributing Frequently-Changing Documents in the Web: Multicasting or Hierarchical Caching", *Computer Networks and ISDN Systems. Selected Papers of the 3rd International Caching Workshop*, pp. 2223–2245, 1998.
12. A. Rousskov, "On Performance of Caching Proxies", In *ACM SIGMETRICS*, Madison, USA, September 1998.
13. SkyCache, "http://www.skycache.com".
14. R. Tewari, M. Dahlin, H. M. Vin, and J. S. Kay, "Beyond Hierarchies: Design Considerations for Distributed Caching on the Internet", In *Proceedings of the ICDCS '99 conference*, Austin, Texas, May 1999.
15. D. Wessels and K. Claffy, "Application of Internet Cache Protocol (ICP), version 2", Internet Draft:draft-wessels-icp-v2-appl-00. Work in Progress., Internet Engineering Task Force, May 1997.
16. G. K. Zipf, *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*, Addison-Wesley, Reading, MA, 1949.