Evaluating Cache-Layering to Improve Web Cache System Performance

Bruno Abrahão **Fabrício Benevenuto**

[1]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627, Belo Horizonte, MG
CEP : 31270-901

{brut,fabricio}@dcc.ufmg.br

Abstract. *An ISP (Internet Service Provider) configures a multi-level cache system in order to reduce the amount of bandwidth needed to serve its clients and to improve user's access performance. We evaluate a cache system management policy within an ISP, which we call "cache-layering", using a new framework and a metric for the characterization of Web traffic and system design: the ADF(Aggregation, Disaggregation and Filtering) framework and Entropy. Using Entropy, the ADF framework and simulations with actual proxy traces, we were able to find that the use of "cache-layering" can improve system's efficacy and reduce costs within an ISP.*

1. Introduction

Growth, increasing demand and ascending number of users are terms that have become frequent in the literature to describe the Web phenomenon. In fact, the number of users, as well as the load that they impose on the network, have increased rapidly over the last few years. In this context, Web proxies have gained widespread popularity as it function as intermediaries between Web clients(browsers) and Web servers, aiming on reducing the amount of data being transmitted over the Internet and, therefore, increasing the Web performance experienced by a client.

A proxy server can be understood as an aggregator and a disaggregator of traffic. Such servers are usually used to delimit a portion of network, in which nodes have the same geographic location. For instance, it is often placed as a point of communication between a local area network(LAN) and the Internet. In this configuration, every request originated by a client belonging to the LAN, and every response from the Web, passes through the proxy server. When a file request is made by a client within the LAN, not only does it forward the request to the destination Web server, but it also receives the incoming response in the opposite direction. In addition to forwarding the received file to the destination client, it stores a copy in its local cache. This copy provides the ability to satisfy future clients requests to the same file without the necessity of contacting the end-server again. Since every client request passes through the proxy server, we say that it aggregates the sequences of requests, also known as streams of requests, into one single stream that will be processed by the cache application. In other sense, we can also

consider the proxy server as a traffic disaggregator, since it distributes the requests from its local clients to several TCP 'pipes' that go to different destinations over the Internet.

As the proxy server aggregates and disaggregates traffic, it plays its most important role — a filter. When the streams of requests for Web objects, originated by clients, pass through a proxy server, only the requests that caused a miss in the proxy cache are forwarded in the direction of the destination Web server. On the other hand, the hit requests are served by the proxy server that, transparently, originates a response to the client.

Aggregation, disaggregation and filtering are the three phenomena that streams of requests are subjected to as they pass through proxies servers.

The Web caching system consists of a multi-level proxy cache. In this configuration the cache located in the end-user machine (Web browser cache) is in the lowest level of the hierarchy. One level above, there are the intranet caches, which consist of proxies of universities and organizations. As we ascend in the hierarchy, regional proxies appear and so on. A request that cannot be satisfied by one proxy cache is sent to a parent proxy until it can be satisfied or, as the last option, the destination server be contacted. The design of an effective proxy hierarchy, regarding to the regional location of its components, the size of each cache and the configuration of the levels involves the study of how the properties of the streams change as they are being altered by the proxies servers. However, the Internet is a collection of independent systems (or networks) composed by independent components and, therefore, its configuration is not over one's control.

An ISP (Internet Service Provider) configures a multi-level cache system in order to reduce the amount of bandwidth needed to serve its clients and improve user's access performance. The components of this cache system are usually connected by a high-speed network (i.e. $100Mbps$ LAN) and the ISP administrator has the entire control over the configuration of the hierarchy and its components. This means that one can change all the structure and configuration of this system to improve efficacy.

This flexibility motivated us to evaluate a cache system structure within an ISP, which we call "cache-layering". To do this, we made use of a new framework and metric for the characterization of Web traffic and system design. For instance, we use the ADF(Aggregation, Disaggregation and Filtering) framework and the Entropy metric [Fonseca et al., 2003]. ADF is used to represent the transformations that occur in the streams of request as they pass through the Internet Topology, while Entropy is a metric proposed to characterize the popularity profile in a sequence of requests.

Previous works made use of hit ratio and curve fitting of the popularity distribution to evaluate Web cache systems. As this approach presents limitations [Fonseca et al., 2003], we show how the new metrics can be used to accomplish the task. Using Entropy combined with ADF and simulations with actual proxy traces, we were able to find that the use of "cache-layering" can improve access performance and reduce costs within an ISP.

2. Related Work

There has been considerable work done in the study Web request streams. In particular, many studies have looked at the locality properties [Almeida et al., 1996, Jin and Bestavros, 2000] of such streams. Locality is the property found in a stream that states that references to a set of objects tend to be close in time to references to another set of objects. These studies were motivated by the impact of locality on the design and performance of caching and prefetching systems.

As examples of the application of these studies in systems engineering, they have informed the development of cache replacement policies [Wang, 1999, Cao and Irani, 1997], inter-cache coordination protocols [Fan et al., 2000] and prefetching algorithms [Bestavros, 1995].

The first ideas on how to characterize the effects of proxies in a stream of requests were first introduced by [Weikle et al., 1998]. This work introduces the view of caches as filters, and compares properties of incoming and outgoing streams of references, in the context of program memory references. In the context of Web caching, Mahanti, Williamson and Eager [Mahanti et al., 2000] study how temporal locality changes at different levels in the caching hierarchy. They show that the concentration of references tends to diminish, and the tail of the Zipf's distribution increases, as one goes up the caching hierarchy. This effect is also noted and characterized in [Williamson, 2002, Doyle et al., 2001].

While [Williamson, 2002] does not consider transformations other than filtering, [Fonseca et al., 2003] introduced the study of two other transformations that streams are subject to: Aggregation and Disaggregation. They have also organized these transformations in a framework, proposed and validated metrics for the analysis of temporal locality as the streams move through this framework.

[Williamson, 2002] proposes a cache management policy, in which the first level of the cache hierarchy is allowed to cache only documents below a certain threshold S of file size, and the upper level is allowed to cache only documents that size is greater than S. They conclude that this scheme, called a *file-size-based* partitioning, can improve the overall performance of the system. We evaluate a similar idea, however we restrict only the first level from caching image files, allowing the second level to caching all objects. We call this scheme "cache layering". Instead of aiming on evaluate replacement policies, as in [Williamson, 2002], we are interested in exploiting and evaluating *file-type-based-layering*, using the tools proposed by [Fonseca et al., 2003].

3. ISP Cost-Optimal Problem

An ISP (Internet Service Provider) is a company that provides a backbone access for clients within a regional location. In addition to serving individuals, ISPs also serve large companies, providing a direct connection from the company's networks to the Internet.

Therefore, the ISP contracts an amount of bandwidth from one or more backbones. From the ISP point of view, it is interesting to maximize the number of clients. However, the growth of number of users cannot decreases the quality of service experienced by the

clients. In other words, it has to preserve the SLA (Service Level Agreement) established with each client. This means that it has to improve systems capacity, as more clients impose more load onto the system. On the other hand, it is also interesting to reduce costs with bandwidth purchases from the backbones. Nonetheless, also preserving the SLA.

It is worthy noting that not all client's requests must, necessarily, pass through the backbone link. Once a document was firstly requested by an ISP client, it can be stored in the local ISP cache system and, thus, future requests to the same file can be served locally. As a result, for the sake of cost reduction and improvement of user's access performance, the ISP configures a multi-level cache system. An illustration of an ISP scheme, just described, can be seen in Figure 1.

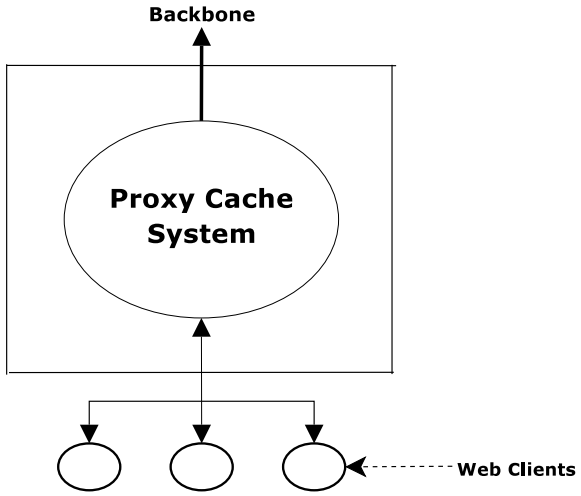
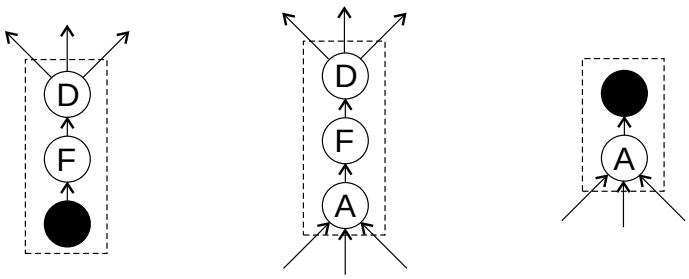


Figure 1: ISP Scheme: a set of Web clients connect to an ISP to request documents, from its local cache, which forwards miss requests to the Internet using the backbone link.

The ISP cache system can reduce the amount of bandwidth needed from the backbone and reduces the system's response time to serve user's requests. The components of this cache system (i.e. cache machines) are usually connected by a high-speed network (i.e. $100Mbps$ LAN) and the ISP administrator has the entire control over the configuration of the hierarchy and its components. Although a cache system can improve ISP capacity and reduce costs, there is an optimal cache system size that brings costs of infrastructure and bandwidth purchase to minimum and the system capacity to the performance level agreed with clients. Hipotetically, going to extremes, increasing cache system size to infinity does not discharge the purchase of some bandwidth. Conversely, cache system of size zero means that all the client's requests must pass through ISP backbone link. The ISP Cost-Optimal problem consists of finding the optimal size of the cache system, which would reduce the infra-structure and backbone costs and maximize the ISP capacity.

Cache hierarchies have proven to be a fundamental element for scalability of caching systems [Williamson, 2002]. Instead of tackling the ISP cost-optimal problem, we aim on improving cache system efficacy, improving the hierarchy scheme and maintaining the existing infra-structure resources. This obviously reduces the costs involved and increase system's capacity. We do this by evaluating a management policy for organizing cache hierarchy within an ISP.



(a) Client (b) Proxy Cache (c) End Server

Figure 2: ADF Abstractions. Within components, (A) stands for Aggregator , (D) for Disaggregator, and (F) for Filter. Points of generation and absorption of requests are represented in black

4. Framework and Metrics

In order to study and understand a system of caches, we are using the ADF framework and the metric Entropy, proposed by [Fonseca et al., 2003]. The ADF framework represents the transformations on request streams that commonly occur in the Web: stream aggregation, disaggregation, and filtering.

4.1. ADF Framework

This abstraction views the topology of the Web system as a graph, in which the nodes represent points where the streams may be altered (such as clients, servers and caches), and the edges are paths connecting these points.

The nodes in this graph are of three different types, depending on what their effect is on the Web traffic: Aggregator (A), Disaggregator (D) and Filters (F); different components of the Web topology may be represented by combinations of these three kinds of nodes. There are also endpoint nodes, which can generate and absorb requests; these are clients and servers, respectively.

The three types of nodes correspond to three phenomena affecting streams:

- **Client** : Figure 2(a) shows how a client might be represented in the framework. The request stream that a user generates goes through a filter node (F) (the browser cache). After this, this same request stream is split into different streams, that go to different destinations over the Web. This can be abstracted by a disaggregator (D) node.
- **Web Proxies** : Figure 2(b) shows the configuration of a Web proxy cache server in the graph. It is often located at some intermediate point of the topology, between many clients and many servers. When requests are received by a common cache server through different TCP connections they are aggregated in a single stream, so that the cache subsystem itself sees a single stream. This can be represented by an aggregator node in the graph. This aggregated stream is then processed by the cache, and the miss stream that leaves the cache is the result of a filtering operation by the cache. The next node is a disaggregator node, from which several edges leave, headed towards different servers as well as other proxy servers.

- **Web Server** : The representation for a Web server is shown in Figure 2(c). In these nodes, there occurs an aggregation operation, and several streams coming upward are multiplexed into a single stream.

Using the ADF framework to categorize Web stream transformations into these three kinds, we can start to study the popularity properties, before and after the effects of these operations, of the request streams.

4.2. Entropy

Many studies about Web traffic has been focused on the temporal locality properties presented in the traces. The intuition of temporal locality is: "An object just referenced has a *high* probability of being referenced in the near future" (*e.g.*, [Phalke and Gopinath, 1995]);

Recently, a number of authors have focused on two ways in which temporal locality can arise. The separation of temporal locality into these two effects was first suggested by Jin and Bestavros [Jin and Bestavros, 2000] and by Mahanti, Eager, and Williamson [Mahanti et al., 2000]. Jin and Bestavros termed these two effects *popularity* and *temporal locality*. In this paper we focus only in the popularity effect.

The popularity distribution of a workload are usually characterized by the term *Zipf's Law*[Glassman, 1994, Almeida et al., 1996, Breslau et al., 1999]. Zipf's Law states that the popularity of the n^{th} most popular object is proportional to $1/n$. More generally, "Zipf-like" distributions have been found to approximate many Web reference streams well. In such a distribution:

$$P[O_n] \propto n^{-\alpha}$$

in which $P[O_n]$ is the probability of a reference to the n^{th} most popular object; typically, $\alpha \leq 1$.

The implication of Zipf-like distributions for reference streams is to show that most references are concentrated among a small fraction of all of the objects referenced.

Rather, a direct measure of such deviation is available: *entropy*. [Shannon, 1948, T. M. Cover and J. A. Thomas, 1991] The entropy of a random variable X taking on n possible values with probability p_i is simply:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i. \tag{1}$$

The entropy measures the deviation of X's distribution from the uniform distribution. It takes on its maximum value ($\log_2(n)$) in the case where all symbols of X are equally likely (*i.e.*, $p_i = 1/n,\ i = 1, \ldots, n$.) It takes on its minimum value (zero) in the case where only one symbol occur.

The reason for preferring entropy over the Zipf exponent α is that real data often does not fit a Zipf-like distribution perfectly. As a result, measurement of the Zipf exponent can be subjective [Fonseca et al., 2003]. The strength of the entropy metric is that it requires no modeling assumption about the data, and so captures popularity skew equally well whether the trace adheres to a power law or not.

In estimating $H(X)$ from a given log, we view the requests in the log as a sequence of symbols, which are the requested objects; we approximate the probability p_i of a given object i being referenced as the number of times it appears in the log, divided by the total references in the log. We thus obtain an empirical probability distribution over the set of objects in the log. Then the entropy $H(X)$ is defined as in Equation (1). Note that $H(X)$ only depends on the probabilities of occurrence of the different objects, and not on the relative order in which they are occur.

Entropy needs a normalization the avoid the dependence on the number of distinct objects that are referenced in the log. The appropriate normalization is based on the largest possible value of $H(X)$, namely $H_0(N)$. Therefore the metric for popularity we will use is the normalized entropy:

$$H^n = H(X)/H_0(N) \tag{2}$$

where N is the number of distinct references in the log.

In [Fonseca et al., 2003], they use H^n with the following transformation of H^n:

$$H^s = -\log_{10}(1 - H^n) \tag{3}$$

H^s (called "scaled" normalized entropy) is used because H^n can often be quite close to 1, making it hard to distinguish on plots.

5. Exploiting Entropy to Improve Hierarchy Performance

Interesting design issues arise with cache hierarchies. As an example, these systems suffer from the phenomenon "diminishing returns" that means that the further up in the hierarchy the cache is, the less likely one can find a document of interest in a cache. This fact occurs despite the fact that higher-level caches are often large in size than lower-level caches [Williamson, 2002]. These observations suggest that the hierarchies are not well-designed. In this context, we evaluate one approach that might improve overall performance of a Web proxy caching system, which we called "cache-layering".

To explain the idea behind this approach, lets consider the configuration in Figure 3. This Figure shows a two-level cache system with two caches (child) in the first level and one (parent) in the second. The user's requests are received directly by the caches at the first level and the requests that cannot be satisfied by the first-level caches are forwarded to the parent cache. Requests from the two first-level caches are aggregated, forming the input stream of the second level cache. There is no directly interaction between the two lower-level caches.

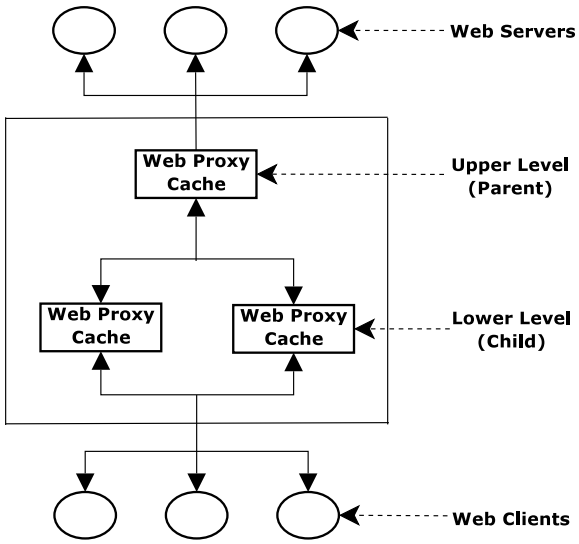


Figure 3: Two Level Caching System Hierarchy

When analyzing traffic aggregation in the second level, it falls into one of the following four cases:

1. Complementing Traces

   ```
   Trace 1: AAAB
   Trace 2: BBBA
   ```

2. Equal Traces

   ```
   Trace 1: AAAA
   Trace 2: AAAA
   ```

3. Disjoint Traces

   ```
   Trace 1: AABCDDA
   Trace 2: ZZKYHHR
   ```

4. Overlapping Traces

   ```
   Trace 1: AbAcAdAe
   Trace 2: AfAgAhAi
   ```

In case number 1 (complementing traces), despite the sub-traces have some concentration of popularity, it is lost when they are aggregated. The resulting trace will now have a uniform distribution, that is, the entropy increases. In the second case (equal traces), there is no variation of entropy when sub-traces are merged. The third case (disjoint objects), it was verified that the resulting entropy is the mean of the sub-traces' entropy. In the fourth case (overlapping traces), aggregation decreases entropy because there is inter-trace overlapping. In this case, when traces 1 and 2 are merged, the concentration of popularity increases, because objects' popularity is unchanged, but the popularity of all other objects decreases.

We can see that aggregation can result in different scales of entropy. In general, it is desirable, in order to increase hit ratio of the second-level cache, to aggregate outgoing links of the first level caches (or even clients), in which the fourth case occurs more frequently.

As a consequence of those facts, we evaluate a cache management policy consisting of two layers. In this approach, we restrict the first level from caching images files (i.e. jpeg and gif files). We call this scheme ”cache-layering”.

The motivation behind the use of this model is the following. Since the first-level caches try to filter out all the locality (i.e. popularity) present in the traces, this reduces the document hit ratio of the second-level cache. When we restrict the first-level caches from caching some kinds of documents, we expect that two phenomena occur:

- the likelihood that the overlap of two outgoing miss streams (from first-level) increases, since the documents with the same characteristics (images) output from the miss streams of the lower-level caches. This would produce a higher document hit ratio in the second-level cache.
- overall system storage increases, since some objects (images) are not replicated in the lower-level caches.

We simulated this scheme using actual proxies traces, described in the next subsection. We made use of two metrics to evaluate these schemes: *document hit ratio*, which is the traditional metric used by previous work and means the number of requests satisfied by the cache, divided by the total number of requests seen by the cache, and *Entropy*, described in Section 5.

5.1. Workload Characterization

In this section, we characterize the set of traces from POP-MG [pop,], a regional *Point of Presence* of the National Research Backbone (RNP-Brazil) [rnp,], which serves the state of Minas Gerais (Brazil). POP-MG serves, besides corporate clients, universities and individuals that use Internet via radio transmission.

POP-MG has a cache system hierarchy organized as follows. It has two levels of caches. In the first level, it has two caches machines, where they make load balance to receive client’s request directly. In the next upper level, it contains two machines: one to answer request for national files (i.e. domain.*br*), and other to answer request for non-national files (i.e. *not* domain.*br*). The lower level receives requests direct from clients and, if misses occur, it forwards them to the next level, demultiplexing into the two second-level machines, according to the domain suffix (.*br* or !.*br*). POP-MG scheme can be seen in Figure 4.

We obtained two days of access logs from the two level 1 caches machines, which we called *pop1* and *pop2*. The logs follow the *Squid* [National Laboratory for Applied Network Research,] format and the details of the workload are summarized in Table 1. The workload is assumed to contain only static objects.

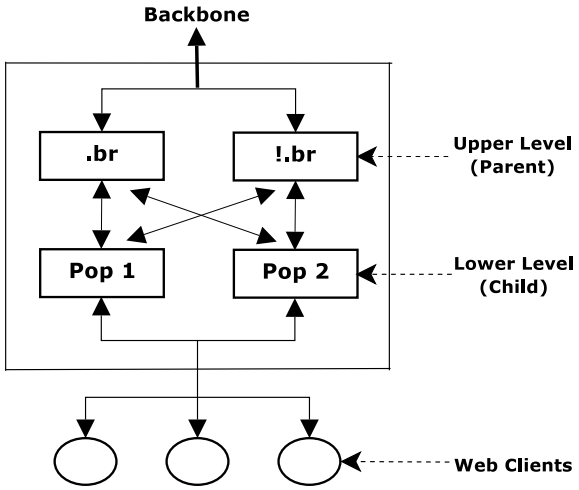


Figure 4: POP-MG ISP Scheme

Item	Pop-1	Pop-2
Period	10/18 - 10/19/01	10/18 - 10/19/01
Total requests	902,998	919,541
Image file requests	542,022	529,041
Image file requests (% of total)	60%	57%
Unique documents	238,880	237,290
Unique documents (% of requests)	26%	26%
One-timers	164,011	164,878
One-timers(% of unique documents)	18%	18%
Working set (Mbytes)	1,887	2,058
Working set of images (Mbytes)	806	865
Smallest file size (bytes)	0	0
Largest file size (bytes)	3,266,488	1,078,310
Mean file size (bytes)	8,283	9,095
Smallest image file size (bytes)	0	0
Largest image file size (bytes)	1,607,753	866,455
Mean image file size (bytes)	5,092	5,463
Normalized entropy	0.8563	0.8336
(Scaled) Normalized entropy	0.8426	0.7788

Table 1: Workload Features

Looking at the log features, we can see that, although the unique objects represent approximately 26% of the total of requests in both cases, the concentration of popularity is low, indicated by the value of H_n close to 1. In addition, the logs present 18% of one-timers objects. These are the objects that received only one reference during the period of time of the logs. Clearly, there is no benefit of caching one-timers documents.

The working sets have sizes 1,887 and 2,058 Mbytes and those represent the maximum size of the lower-level caches that stores all the objects present in the reference stream (when all objects are cacheable).

Figures 5 and 6 show the size distribution of the documents for both traces. We

can see that the small documents appear more frequently in both profiles. Figures 7 and 8 show the same size distribution for image files.

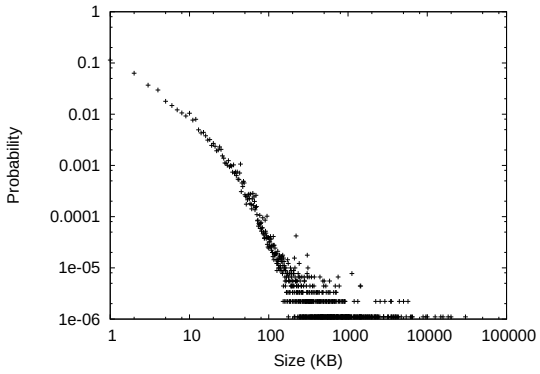


Figure 5: pop-1 Working Set

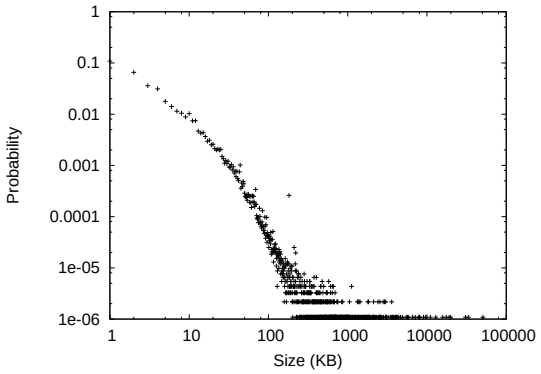


Figure 6: pop-2 Working Set

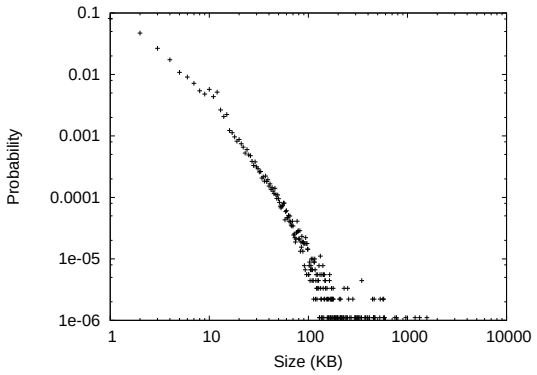


Figure 7: pop-1 Images Working Set

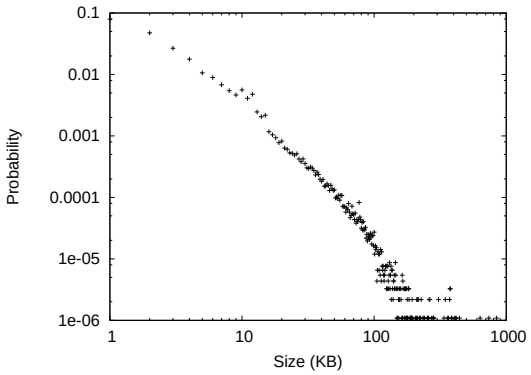


Figure 8: pop-2 Images Working Set

5.2. Experimental Methodology

This Section discusses the methodology used for the simulations we have performed. We have constructed a system of caches using cache simulators based on LRU replacement policy. These simulators are able to output metrics such as the document hit ratio of an execution. One problem that may arise is that our cache simulator execution includes *cold start* misses, that is, the cache is empty before the simulation starts. We have tried to diminish this effect by warming up the caches simulators with a preliminary execution, using the request streams.

The cache simulators are organized as shown in Figure 9. This Figure also shows the points where we submit the workload (request streams) to the system and collect the metrics. Points 1 and 2 represent the place where we submitted the *pop-1* and *pop-2* workloads. Points 3 and 4 are where we collected the aggregated entropy from the lower-level caches and the entropy of system's output respectively.

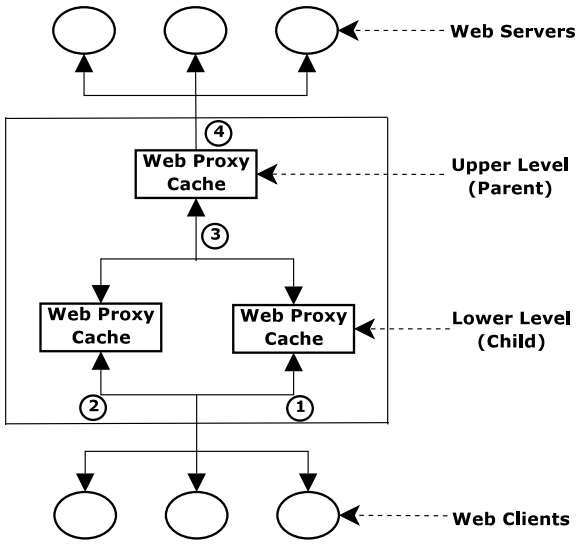


Figure 9: Entropy Measurements Points

The first experiment considers all documents as cacheable. Considering all static documents as cacheable is the most common management policy in practice. This experiment will be used as a reference for cache-layering experiment. We called this configuration, the *common policy*.

In addition, we develop the *file type-based-layering* experiment, which consists of treating images as non-cacheable in the lower-level caches.

In our experiments we have simulated these management policies for several cache sizes. We equally increase the cache sizes following a power of 2, starting from $1Mb$ to $1024Mb$. The maximum value represents more than the 50% of the size of the working set.

6. Experimental Results

This Section presents the results achieved by the simulation of the two management policies using the empirical traces.

Analyzing the internal profile of each system configuration of caches is useful for the understanding of system's behavior. For this purpose, our initial analysis consisted of collecting the *document hit ratio* of each cache of the system, for each management policy.

The behavior for the common policy is shown in Figure 10. Using this configuration, we can see that the first-level caches achieve higher document hit ratio than the second-level cache. The reason for that profile is that filtering absorbs part of the temporal locality of a request stream and generates a miss stream consisting of evenly distributed references to fairly popular objects. This means that first-level caches filter out the locality property of the stream presented to the parent cache. From this scenario, we can conclude that the second-level cache is not well exploited.

This behavior is inverted when analyzing the profile of *file-type-based-layering* management policy. This inversion is due to misses caused by the *non-cacheable* files

in the first level, and to the potential hits that the second level can achieve by caching image files. Furthermore, objects of the same type (images) are being aggregated in the input stream of the parent cache, coming from the miss streams of the first-level caches. Because of this, the overlapping of the two traces increases, decreasing the entropy of the input stream presented to the parent cache. This effect is shown in Figure 12, which shows the entropy of the aggregated stream formed by the miss streams of the lower-level caches. Looking at this plot, we can see that the entropy decreases when we simulated *file-type-based-layering* management policy rather than the common policy. Conversely, presenting a stream with lower Entropy to the parent cache leads the achievement of better performance, or higher document hit ratio, in this level, as shown by the curve for *file-type-based-layering* in Figure 13.

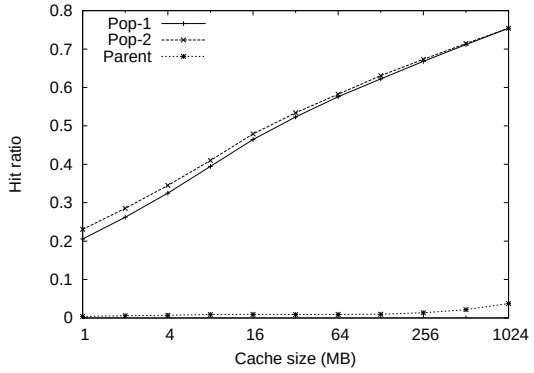


Figure 10: Hit Ratio of Each Cache Using the Common Policy

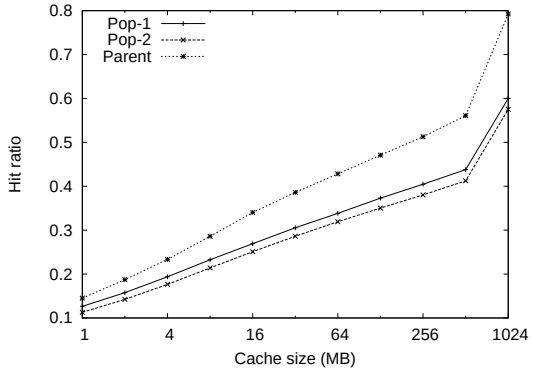


Figure 11: Hit Ratio of Each Cache Using File-Type-Based-Layering

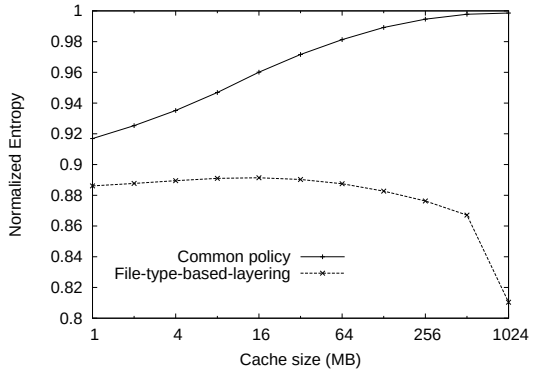


Figure 12: Aggregated File Entropy

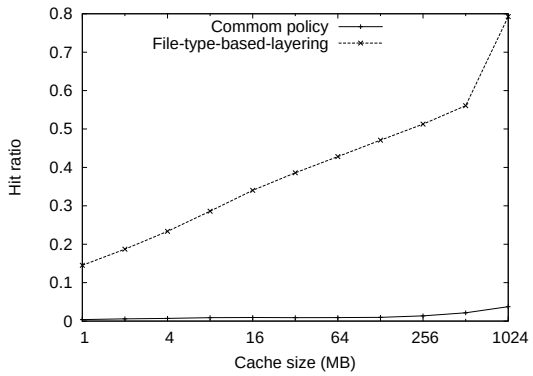


Figure 13: Second-Level Cache Hit Ratio

Restricting the first-level caches from store image files, reduces the number of copies of these files in whole system, since they will be stored only in the second level. Increasing storage means that more files can be cached by the system, therefore, increasing the *document hit ratio* of the entire system. Figure 14 shows the number of files stored in the system in the end of the execution of the experiment, as we vary cache size. We can see that *file-type-based-layering* management policy was able to store more files, for all cache sizes, rather than the common policy.

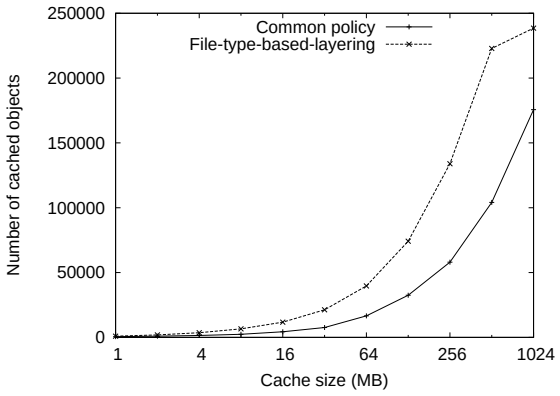


Figure 14: Storage at the end of simulation execution

Finally, the efficacy improvement of the system can be evaluated by two ways. First, we expect that the output stream of the system, which will be submitted to the backbone link, had the minimum number of requests as possible. This means that we want to avoid the same document to appear more than once in this stream, reducing the bandwidth needed from the backbone link. Thus, we can seize this effect by measuring the entropy of the system's output stream. We do this for the two management policies, for several sizes of caches. Figure 15 shows the entropy of the whole system's miss stream. We can see, looking at the plot, that *file-type-based-layering* produces higher entropy in the system's output stream than the common policy, for all sizes of caches considered. Similar results were found analyzing the total document hit ratio of the system. Figure 16 shows that the use of the cache-layering approach can improve the overall system hit ratio.

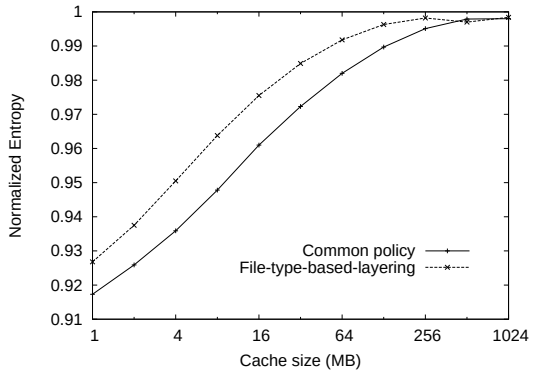


Figure 15: Entropy of the output stream of requests (backbone)

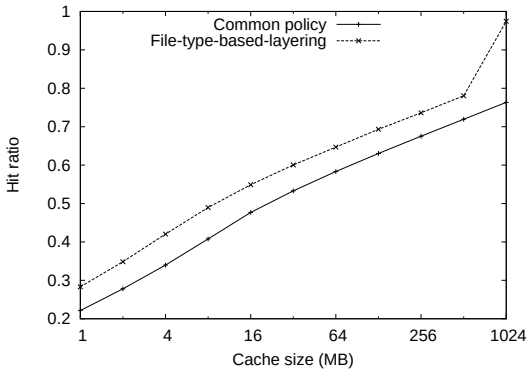


Figure 16: Whole System's Hit Ratio

7. Conclusion

An ISP (Internet Service Provider) configures a multi-level cache system in order to reduce the amount of bandwidth needed to serve its clients and to improve user's access performance. In this paper, we evaluate a cache system management policy within an ISP, which we call "cache-layering".

Using Entropy measurements and ADF, our simulations with actual proxy traces found that the use of ”cache-layering” can increase the efficacy of the parent caches, increase the total system storage and, therefore, increases the overall system efficacy.

The system's improvements were demonstrated by showing the increase of entropy of the output stream of the whole system and the increase of the hit ratio, using the cache-layering approach proposed. Since the infra-structure resources needed are unchanged, there is also cost reduction for the ISP point of view.

In future analysis, we intend to consider *file-size-based-layering* approach that consists of specifying a size threshold S and allow the lower level caches to store only files smaller than S, while the upper level cache is allowed to store files of all sizes. This approach provides a natural partitioning of the document space, using minimal information since document size information is available to Web servers and proxies in the HTTP response header.

The main contribution of this paper is to show how entropy and ADF can be used to evaluate the performance of Web caching system schemes.

8. Acknowledgements

The authors would like to thank Prof. Wagner Meira Jr. and Prof. Virgilio Almeida for their valuable contributions to this work.

References

POP-MG. Ponto de Presença da Rede Nacional de Pesquisa em Minas Gerais. `http://www.pop-mg.rnp.br`.

RNP. Rede Nacional de Ensino e Pesquisa. `http://www.rnp.br`.

Almeida, V., Bestavros, A., Crovella, M., and de Oliveira, A. (1996). Characterizing reference locality in the WWW. In *Proceedings of the Fourth International Conference on Parallel and Distributed Information Systems (PDIS96)*.

Bestavros, A. (1995). Using speculation to reduce server load and service time on the www. In *Proceedings of CIKM'95*, Baltimore, Maryland.

Breslau, L., Cao, P., Fan, L., Phillips, G., and Shenker, S. (1999). Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proceedings of IEEE Infocom*.

Cao, P. and Irani, S. (1997). Cost-aware WWW proxy caching algorithms. In *Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems (USITS-97)*, Monterey, CA.

Doyle, R., Chase, J., Gadde, S., and Vahdat, A. (2001). The trickle-down effect: Web caching and server request distribution. In *Proceedings of the 6th Web Caching Workshop*, pages 1–18.

Fan, L., Cao, P., Almeida, J., and Broder, A. Z. (2000). Summary cache: a scalable wide-area Web cache sharing protocol. *IEEE / ACM Transactions on Networking*, 8(3):281–293.

Fonseca, R., Almeida, V., Crovella, M., and Abrahao, B. (2003). On the intrinsic locality properties of web reference streams. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*.

Glassman, S. (1994). A caching relay for the World Wide Web. In *Proceedings of the First International World Wide Web Conference*, pages 69–76.

Jin, S. and Bestavros, A. (2000). Sources and Characteristics of Web Temporal Locality. In *Proc. of the 8th Int. Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. IEEE Computer Society Press.

Mahanti, A., Eager, D., and Williamson, C. (2000). Temporal locality and its impact on web proxy cache performance. *Performance Evaluation Journal: Special Issue on Internet Performance Modelling*, 42(2/3):187–203.

National Laboratory for Applied Network Research. Squid Internet Object Cache. http://squid.nlanr.net/Squid/.

Phalke, V. and Gopinath, B. (1995). An interreference gap model for temporal locality in program behavior. In *Proceedings of the 1995 ACM SIGMETRICS Conference*, pages 291–300.

Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656.

T. M. Cover and J. A. Thomas (1991). *Elements of Information Theory*. John Wiley and Sons.

Wang, J. (1999). A survey of Web caching schemes for the Internet. *ACM Computer Communication Review*, 25(9):36–46.

Weikle, D., S.Mckee, and W.Wulf (1998). Cache as filters: A new approach to cache analysis. In *6th Intl. Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS'98)*.

Williamson, C. (2002). On filter effects in web caching hierarchies. *ACM Transactions on Internet Technology*, 2(1):47–77.