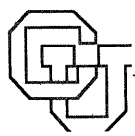
A Feasibility Study of Bandwidth Smoothing on the World-Wide Web Using Machine Learning

Carlos Maltzahn
Kathy J. Richardson
Dirk Grunwald
James Martin

CU-CS-879-99



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

A Feasibility Study of Bandwidth Smoothing on the World-Wide Web Using Machine Learning

Carlos Maltzahn and Kathy J. Richardson
Compaq Computer Corporation
Network Systems Laboratory
Palo Alto, CA
carlosm@cs.colorado.edu, kjr@pa.dec.com

Dirk Grunwald and James Martin
University of Colorado
Department of Computer Science
Boulder, CO
grunwald@cs.colorado.edu, martin@cs.colorado.edu

Abstract

The bandwidth usage due to HTTP traffic often varies considerably over the course of a day, requiring high network performance during peak periods while leaving network resources unused during off-peak periods. We show that using these extra network resources to prefetch web content during off-peak periods can significantly reduce peak bandwidth usage without compromising cache consistency. With large HTTP traffic variations it is therefore feasible to apply "bandwidth smoothing" to reduce the cost and the required capacity of a network infrastructure. In addition to reducing the peak network demand, bandwidth smoothing improves cache hit rates.

We calculate the potential reduction in bandwidth for a given bandwidth usage profile, and show that a simple hueristic has poor prefetch accuracy. We then apply machine learning techniques to automatically develop prefetch strategies that have high accuracy. Our results are based on web proxy traces generated at a large corporate Internet exchange point and data collected from recent scans of popular web sites.

Keywords: bandwidth smoothing, web proxy servers, machine learning

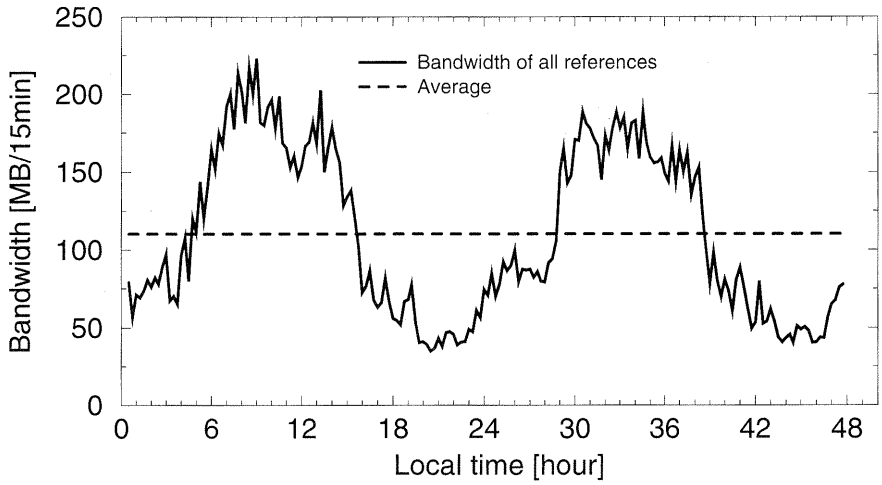


Figure 1: The typical bandwidth usage at the Palo Alto Gateway over the course of two weekdays. Each data point in the graph is the average Kbyte per second byte request rate of a 15 minute interval. Bandwidth usage varies dramatically each day but the fluctuations are similar each weekday with clearly discernible peak and off-peak periods. In this traffic profile the peak/off-peak boundaries lie at around 4:30 AM and 4:30 PM.

1 Introduction

The bandwidth usage due to web traffic can vary considerable over the course of a day. Figure 1 shows the web traffic bandwidth usage at the Palo Alto gateway of Digital Equipment Corporation. The figure shows that peak bandwidth can be significantly higher than the average bandwidth usage. Bandwidth usage varies dramatically each day but the fluctuations are similar each weekday with clearly discernible peak and off-peak periods. These diurnal access profiles are typical for enterprise-level web caches [22, 15, 17].

To perform well, a network needs to accommodate peak bandwidth usages. A reduction in peak bandwidth usage will therefore save network resources including lower demand for DNS, lower server and proxy loads, and smaller bandwidth design requirements.

A common approach to reducing bandwidth usage is to cache web objects as they are requested. Demand-based caching reduces both peak and off-peak bandwidth usage. The effectiveness of this form of caching is limited because of the high rate-of-change of large parts of the web content, the size of the web, the working set size, and the object reuse rates [10].

Peak bandwidth usage can also be reduced by shifting some bandwidth from peak periods to off-peak periods. We call this approach *bandwidth smoothing*. In contrast to caching, bandwidth smoothing does not necessarily reduce the daily average bandwidth usage – in fact, it will increase the total bandwidth usage. However, this approach uses unused resources during off-peak to reduce peak bandwidth usage.

Bandwidth smoothing can be accomplished by either appropriately changing user bandwidth usage behavior or by prefetching data during off-peak time. In this paper we will focus on the feasibility of the latter approach.

Web caching is performed by web proxies with caches (web caches) or by routers with attached caches (transparent caches). Both implementations are usually deployed at network traffic aggregation points and edge points between networks of multiple administrative domains. Aggregation points combine the web traffic from a large network user community. This larger user community increases the cache hit rate and reduces latency [11]. Edge points are an opportunity to reduce the bandwidth usage across domains because

inter-domain bandwidth is frequently more expensive than bandwidth within domains. Corporate gateways are usually both aggregation points and edge points. We account for this common configuration by basing our feasibility study on data collected from web proxies which are installed at a major Internet gateway of a large international corporation.

Network resources such as bandwidth are frequently purchased in quanta (*e.g.*, a T1 or T3 line). Reducing peak bandwidth usage by less than a quantum may not result in any cost savings. However, small reductions of peak bandwidth usage in many locations of a large organization can aggregate to savings that span bandwidth purchase quanta and can therefore lead to real cost savings. Reducing peak bandwidth requirements also extends the lifetime of existing network resources by delaying the need to purchase the next bandwidth quantum. For new networks, peak bandwidth reduction reduces the bandwidth capacity requirements which allows the purchase of fewer or smaller bandwidth quanta.

The rest of the paper is organized as follows: in the next section we lay out a framework for bandwidth smoothing, analyze the prefetchable bandwidth of the Palo Alto gateway, and show how to calculate the potential reduction in peak bandwidth usage for a given bandwidth usage profile. In section 3 we show how to measure prefetch performance and how to calculate the potential reduction in bandwidth usage for a given prefetch performance. In section 4 we demonstrate and discuss the performance of machine learning techniques to automatically develop prefetch strategies. Section 5 is an overview of related work. We conclude with a summary and future work.

2 Prefetchable Bandwidth

The goal of bandwidth smoothing is to shift some of the bandwidth usage from the peak usage periods to off-peak periods. Bandwidth smoothing is a technique that requires caching; prefetched items must be stored in the cache until they are referenced. Furthermore, we assume that items remain in the cache whether they are prefetched or demand fetched by a user. Obviously, cached items no longer need to be prefetched.

The effect of caching needs to be taken into account before smoothing techniques are applied to ensure the effects are additive. We are therefore only interested in "steady-state" bandwidth smoothing where we only study the effect of off-peak prefetching on the *directly following* peak period.

Figure 2 shows cache effects on bandwidth consumption. Caching somewhat smoothes the bandwidth consumption because it reduces the magnitude of the peak bandwidth usage more than the off-peak bandwidth usage. Our measurements also indicate that the hit rate during peak periods is higher than during off-peak periods.

One way to accomplish bandwidth smoothing is to predict peak period cache misses and prefetch the corresponding objects during the preceding off-peak period. The remainder of this section presents definitions and evaluates the prefetch potential and characteristics of prefetchable objects.

2.1 Definitions

We call an object *prefetchable* for a given peak period if it has the following properties: the object

- is referenced during the peak period and was not found in the cache,
- exists during the preceding off-peak time,
- is cacheable, and
- is unaltered between the beginning of the preceding off-peak period and the time it is requested.

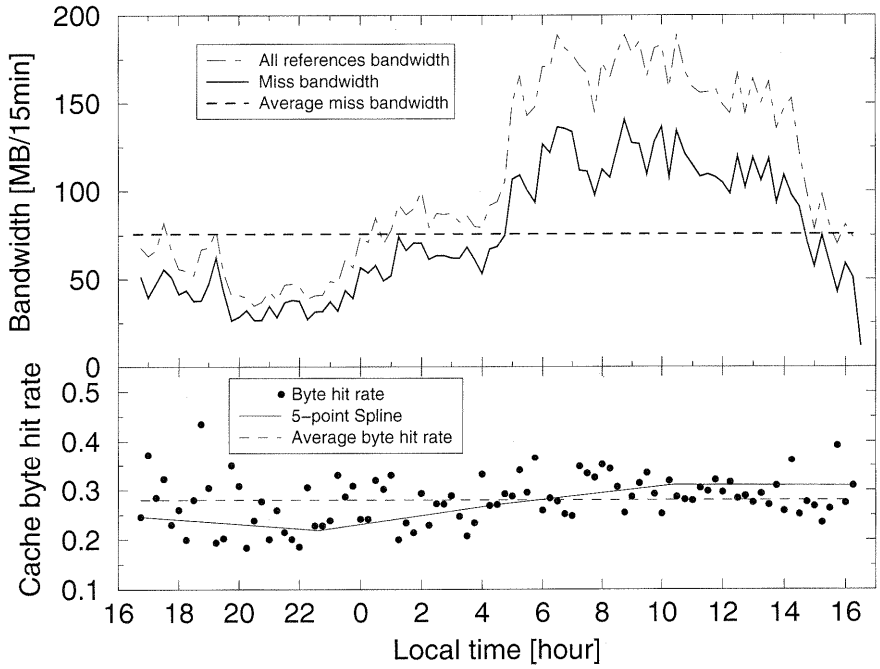


Figure 2: The smoothing effect of traditional caching on bandwidth usage. The cache byte hit rate is higher during the peak period because of more sharing opportunities of a larger network user community.

If an object fails to meet any of these conditions for a given peak period we call it *non-prefetchable* for that peak period. Because of the first condition a non-prefetchable object during one peak period can be a prefetchable object during another peak period. Non-prefetchable objects which meet all prefetchability conditions except the first are called *no-miss-prefetchable* objects.

The combined size of all prefetchable objects of a given peak period is the *prefetchable bandwidth*. There are three disjoint kinds of prefetchable objects, depending on the web access history of the aggregated network user community:

- *Seen prefetchable* objects have names which were previously referenced. These are either *revalidation misses* (caused by stale data) or *capacity misses* (caused by finite-capacity caches).
- *Seen-server prefetchable* objects are referenced for the first time, but they are served from previously accessed web servers. These are *compulsory misses* because they have not been seen before.
- The names and servers of *unseen-server prefetchable* objects were unknown prior to the current reference. Neither the object nor the server were previously accessed. These are also *compulsory misses* because the data has not been seen before

We distinguish *seen-server prefetchable* and *unseen-server prefetchable* objects because predicting the latter kind of objects is more difficult: the proxy access history offers no information about the existence of unseen servers.

2.2 Experimental Measurement and Evaluation Environment

In order to estimate the prefetchable bandwidth for bandwidth smoothing we analyzed the Digital WRL HTTP proxy request traces [16]. The instrumented HTTP proxies were installed at a gateway that connects

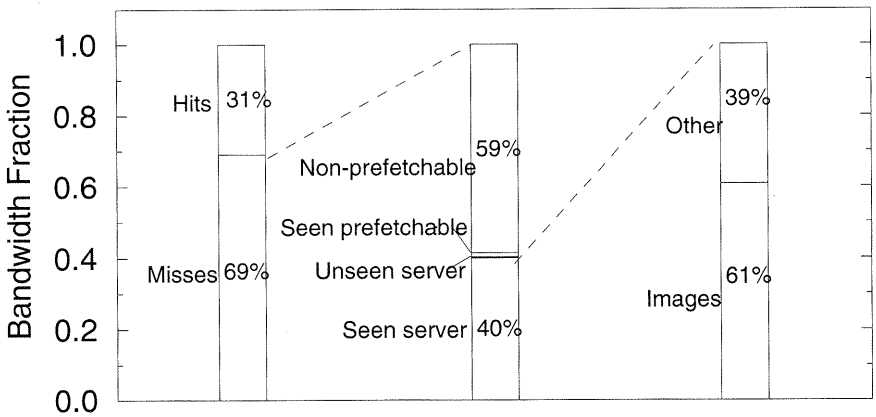


Figure 3: The components of the reference bandwidth. The miss bandwidth was measured for one peak period after a cache warm-up of over two days. The hit rate of the warm cache is 31%. The contribution of seen prefetchables and *unseen-server prefetchables* to the miss bandwidth is negligible.

a large international corporate intranet to the Internet. The traces were "sanitized" to protect individual privacy and other sensitive information such as individual web site access counts. As a consequence, each web server host, path, query, client, and URL (combination of host, path, and query) are encoded by identity preserving, unique numbers. The traces cover the days from August 29th through September 22nd, 1996 and consist of over 22 million requests.

For the sake of simplicity we divided bandwidth usage into off-peak periods starting at 4:30 PM and ending at 4:30 AM each weekday, and peak periods starting at 4:30 AM and ending at 4:30 PM each weekday (see figure 1). We analyzed the HTTP traffic to obtain object age information in order to identify prefetchable objects.

To identify misses in the Digital WRL trace (which was generated by a cacheless web proxy), we assumed (1) a cache of infinite size, (2) a cache hit is represented by a re-reference of an object with the same modification date as the previous reference[1], and (3) the cache never serves a stale object. From this cache model we determine the list of objects and the miss bandwidth for peak and off-peak periods.

To preclude any cache cold-start effects on our measurements we applied this model for at least two days worth of trace data before taking any bandwidth measurements.

2.3 Prefetchable Bandwidth Analysis

Figure 3 shows the composition of the miss bandwidth during a typical weekday peak period of the Digital trace out of an infinite cache. About 40% of the miss bandwidth is prefetchable.

Almost all the prefetchable bandwidth consists of *seen-server prefetchable* objects. The other prefetchable components (*seen prefetchable* and *unseen-server prefetchable* objects) are negligible. With a fixed size cache the number of *seen prefetchable* objects would increase through capacity misses. *Unseen-server prefetchable* objects are entirely workload dependent and independent of cache configurations. For a bandwidth smoothing prefetching strategy to work, it must rely on web server information in order to discover the names of *seen-server prefetchable* objects. Thus, prefetching involves two subproblems: predicting what to look for (the object selection problem) and predicting where to look (the server selection problem).

[1]The modification date must be non-zero. By convention a modification date of zero is used for dynamic and non-cacheable objects, *i.e.* these objects always miss in the cache. Some researchers use modification date and size to differentiate objects [18]. However, we are not aware of any web caches which do not determine object staleness solely based on object age.

Figure 4: The distribution of prefetchable bandwidth over all servers that served a miss during a single peak period, in reverse prefetchable service order. The measurement was taken after a cache warm-up of over two days. The fact that about 4% of servers serve 50% of all prefetchables simplifies the server selection problem.

Before analyzing the server selection problem we introduce a few definitions that proved to be convenient: We call the prefetchable bandwidth served by a server the *prefetchable service* of this server. A *top prefetchable service group* is a subset of all servers such that the subset consists of servers where each server serves a higher amount of prefetchable bandwidth than any server in the subset's complement. Servers in a *prefetchable service order* are ordered by their prefetchable service.

The server selection problem is simplified by the fact that a small number of servers provide most of the prefetchable items. According to our data, 10% of all servers serve 70% of all prefetchable bandwidth. Figure 4 shows the cumulative prefetchable service distribution over servers in reverse prefetchable service order. These results, however, only show the existence of top prefetchable service groups for a given day. The diurnal bandwidth usage suggests there may be a day-to-day stability of top prefetchable service groups. To verify this conjecture, we establish the following heuristic: *if* a server serves prefetchable bandwidth during a given peak period *then* the same server serves prefetchable bandwidth during the following peak period.

We use the "predictive value of a positive test" (PVP) to evaluate this conjecture. The PVP represents the probability that a positive prediction is correct ([1] pages 24-34). We call S the event of prefetchable service of a server on a given peak period (*i.e.*, the heuristic's condition applies), and D the event of prefetchable service of the same server on the following peak period (*i.e.*, the heuristic's consequence holds). Then $PVP = P[S \cap D]/P[S]$, where $P[S \cap D]$ is the probability of cases where event S and D holds, that is the probability that a server serves prefetchable bandwidth on two consecutive peak periods. $P[S]$ is the probability of event S, which is the probability of prefetchable service of a server on any given day.

Thus, the heuristic's PVP is the probability that prefetchable service during the first peak period is a positive indicator for prefetchable service during the following peak period. For a given prefetchable service group a high PVP of this heuristic means the group's day-to-day stability is high and the same servers always serve the majority of prefetchable bandwidth.

Figure 5 shows the heuristic's PVP average and quartiles for "top prefetchable service group" sizes between 100 and 21,000 servers of the entire Digital WRL trace data (which contains 17 peak periods). The average PVP is high for a small top prefetchable service group size, but drops below 0.5 for a group

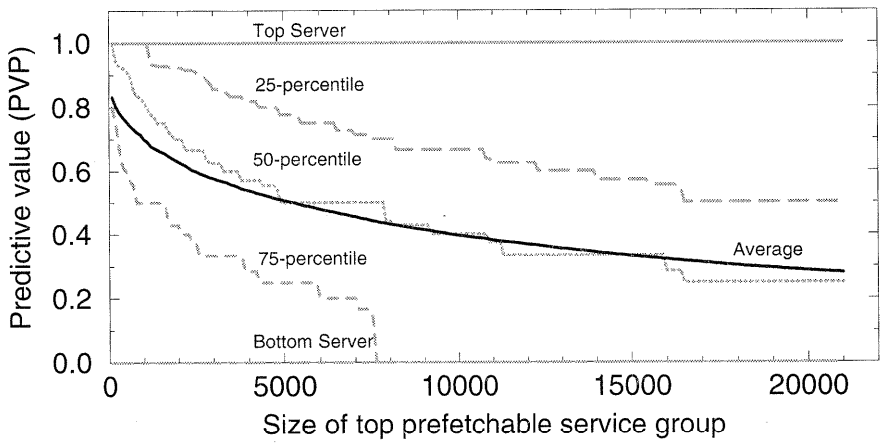


Figure 5: The top prefetchable service group quartiles of the fraction at which a positive prediction of the heuristic is correct. The fast decline of the heuristic's reliability as the number of servers gets larger motivates the use of more complex prefetching strategies and the use of machine learning techniques to automatically find those strategies.

size of over 5,000 servers. The top servers are included in each group. Since they serve prefetchable bandwidth every day, the heuristic's PVP for them is 1.0 (the 100-percentile). However, even the "top 100 prefetchable service" group contains servers which never serve on consecutive days. The quartile curves show the distribution of PVP throughout each group size.

Unless the top prefetchable service group consists only of a few hundred servers, this simple heuristic fails. The large spread of predictability (the difference between the first and third quartile) in larger groups suggests that keeping track of individual server behavior would be beneficial. However, this would be a time-consuming processes for humans, and should be automated to be practical. In section 4 we investigate the performance of server selection mechanism that are automatically derived from using standard machine learning techniques.

2.4 Bandwidth Smoothing Potential

For a given bandwidth usage profile the *bandwidth smoothing potential* Δ_{smooth} is the largest possible reduction of bandwidth cost. Various common bandwidth cost models exist. For the sake of simplicity we will assume a single tariff flat rate cost model (*e.g.*, a flat rate per month for a T1 line). The goal is therefore to keep peak bandwidth usage always below a *target level* L_t. In this case the bandwidth smoothing potential is the difference between the peak bandwidth usage L_m, and the lowest possible target level L_t that can be achieved by prefetching bandwidth (see figure 6), or $\Delta_{smooth} = L_m - L_t$.

In the following, we start symbols with L when they represent a bandwidth usage *level* (*e.g.*, MBytes/15min), and we start symbols with B if they represent the bandwidth usage of an entire period (*e.g.*, MBytes/12h). Thus, L_m and L_t are bandwidth usage levels.

If we assume perfect prefetching and a uniform prefetchable fraction of the miss bandwidth during the peak period, we get a first approximation L_0 of the target level by:

$$L_0 = (1 - \frac{B_{pre}}{B_{peak}})L_m \tag{1}$$

where B_{pre} is the prefetchable bandwidth and B_{peak} the total bandwidth usage during the peak period. L_0 is thus the non-prefetchable component of the peak bandwidth usage.

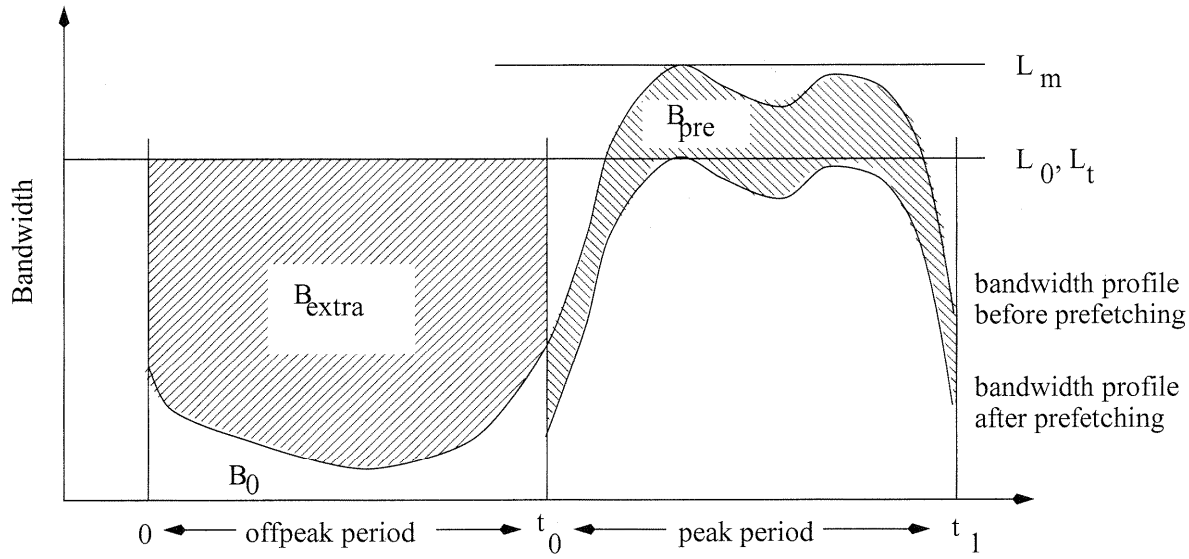


Figure 6: The bandwidth smoothing potential depends on the prefetchable bandwidth (difference between bandwidth profile before and after prefetching) and the extra bandwidth available during off-peak time.

This first approximation does not account for the amount of bandwidth available for prefetching during the previous off-peak period. This extra bandwidth might not suffice to prefetch all of B_{pre}. Using L_0 we compute the extra bandwidth B_{extra} as:

$$B_{extra} = \int_0^{t_0} L_0 dL_0 - B_o \tag{2}$$

where B_o is the total bandwidth usage during the off-peak period. B_{extra} is thus the difference between the integral bandwidth at bandwidth usage level L_0 during the off-peak period and the total off-peak bandwidth usage. We can now compute the target level L_t:

$$L_t = \begin{cases} L_0 & \text{if } B_{extra} \geq B_{pre} \\ \text{fix } f & \text{otherwise} \end{cases} \tag{3}$$

where fix f is the fixed point of the following recursive function f:

$$f(B_{pre}) = \begin{cases} B_{pre} & \text{if } B_{pre} = B_{extra}(B_{pre}) \\ f(\frac{B_{pre}+B_{extra}(B_{pre})}{2}) & \text{otherwise} \end{cases} \tag{4}$$

where the function B_{extra} is based on equation 2 expanded by equation 1:

$$\begin{aligned} B_{extra}(B_{pre}) &= (1 - \frac{B_{pre}}{B_{peak}}) \int_0^{t_0} L_m dL_m - B_o \\ &= -\frac{B_{pre}}{B_{peak}} \int_0^{t_0} L_m dL_m + \int_0^{t_0} L_m dL_m - B_o \end{aligned} \tag{5}$$

It is easy to see that a fixed point for f exists because of the recursive equation $f((B_{pre} + B_{extra}(B_{pre}))/2)$ in f and the fact that B_{extra} is a linear function with a negative slope (see equation 5).

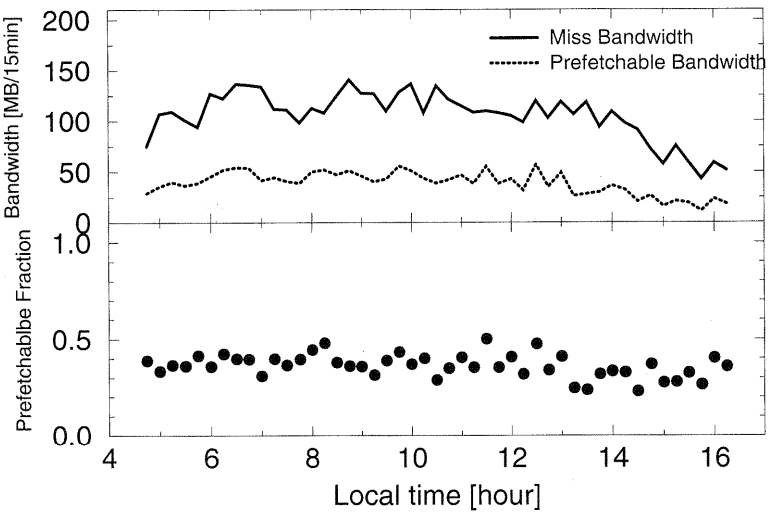


Figure 7: Validation of the assumption that the prefetchable fraction of the miss bandwidth is uniform. The mean prefetchable fraction is 0.36 with a standard deviation of 0.06. If this assumption were 100% correct, the lower part of the graph would be a straight horizontal line. The assumption is accurate enough to estimate the target level.

Recall that these calculations assume that the fraction of prefetchable bandwidth is constant for every measured interval during the peak period [2]. We validate this assumption by comparing the prefetchable bandwidth profile with the miss bandwidth profile (figure 7). The lower part of the figure shows that prefetchable fraction of each measurement. The assumption is accurate enough to estimate the target level.

3 Prefetch Performance

The bandwidth smoothing potential calculation assumed perfect prefetching, *i.e.* all prefetchable objects were prefetched and no extra off-peak bandwidth was wasted for prefetching incorrectly predicted objects. In practice, a certain percentage of the prefetched bandwidth consists of objects fetched because of false positive predictions and a certain amount of prefetchable bandwidth is not prefetched because of false negative predictions. Prefetch performance is defined by three measures:

Accuracy $P_a = B_{++}/B_+$ where B_{++} is the prefetchable component of the prefetched bandwidth and B_+ is the entire prefetched bandwidth. P_a represents the fraction of prefetched bandwidth that is prefetchable (also called *prefetch hit rate*).

Coverage $P_c = B_{++}/B_{pre}$ represents the fraction of the prefetchable bandwidth that has been prefetched.

Timeliness P_t represents the fraction of the prefetched prefetchable bandwidth that will not be modified before its miss time. Timeliness is part of the definition of the prefetchability of an object and therefore an implied property of prefetchable bandwidth.

To clarify the interdependence of accuracy and coverage we draw a 2×2 matrix listing the frequency for each outcome of a prefetch prediction:

[2]Note that we do not assume that this is true for any interval. All our bandwidth usage level measurements are taken in 15 minute intervals

Prefetchable \ Prefetched	$\bar{F}$	F
$\bar{P}$	A	B
P	C	D

where P is "prefetchable", $\bar{P}$ "no-miss-prefetchable", F "prefetched", and $\bar{F}$ "not prefetched"; thus $A = \bar{P} \cap \bar{F}$, $B = \bar{P} \cap F$, $C = P \cap \bar{F}$, and $D = P \cap F$. Expressing accuracy and coverage in terms of this frequency matrix we get $P_a = B_{++}/B_+ = D/(B + D)$ and $P_c = B_{++}/B_{pre} = D/(C + D)$.

In practice, $B_+ = B + D$ is fixed because of limited extra bandwidth during off-peak periods and $B_{pre} = C + D$ is determined by the access history of the corresponding peak period. Hence, a lower accuracy will result in a lower coverage and vice versa.

The equations for estimating the bandwidth smoothing potential in section 2.4 assume 100% accuracy and 100% coverage: according to equation 4 the fixed point is found if the prefetchable bandwidth equals the extra bandwidth available during off-peak periods,

$$B_{pre} = B_{extra}(B_{pre}) \tag{6}$$

If the accuracy is less than 100% we would however need extra off-peak bandwidth to prefetch all prefetchable bandwidth. On the other hand, a coverage of less than 100% reduces the amount of prefetchable bandwidth which we are able to prefetch and therefore requires less extra off-peak bandwidth. This is reflected in the following equation derived from the definition of accuracy and coverage:

$$P_a B_+ = B_{++} = P_c B_{pre} \tag{7}$$

By assuming that we are using all extra off-peak bandwidth for prefetching, *i.e.* $B_+ = B_{extra}$ we can now express equation 6 in terms of accuracy and coverage:

$$B_{pre} = \frac{P_a B_{extra}(P_c B_{pre})}{P_c} \tag{8}$$

We now have the more general calculation of the target level L_t:

$$L_t = \begin{cases} (1 - \frac{P_c B_{pre}}{B_{peak}}) L_m & \text{if } B_{extra} \geq \frac{P_c B_{pre}}{P_a} \\ \text{fix } f & \text{otherwise} \end{cases} \tag{9}$$

where fix f is the fixed point of

$$f(B_{pre}) = \begin{cases} B_{pre} & \text{if } B_{pre} = \frac{P_a B_{extra}(P_c B_{pre})}{P_c} \\ f(\frac{P_c B_{pre} + P_a B_{extra}(P_c B_{pre})}{2}) & \text{otherwise} \end{cases} \tag{10}$$

Figure 8 shows possible target levels of the bandwidth usage profile depicted in figure 2 depending on the prefetchable bandwidth and different accuracy and coverage levels. This figure quantifies our earlier qualitative measures: higher accuracy reduces the needed bandwidth for prefetching during off-peak periods while higher coverage increases the bandwidth savings during peak periods. Our later experiments will show that we can automatically develop prefetch strategies with high accuracy and medium coverage. We developed these tests using a machine learning tool.

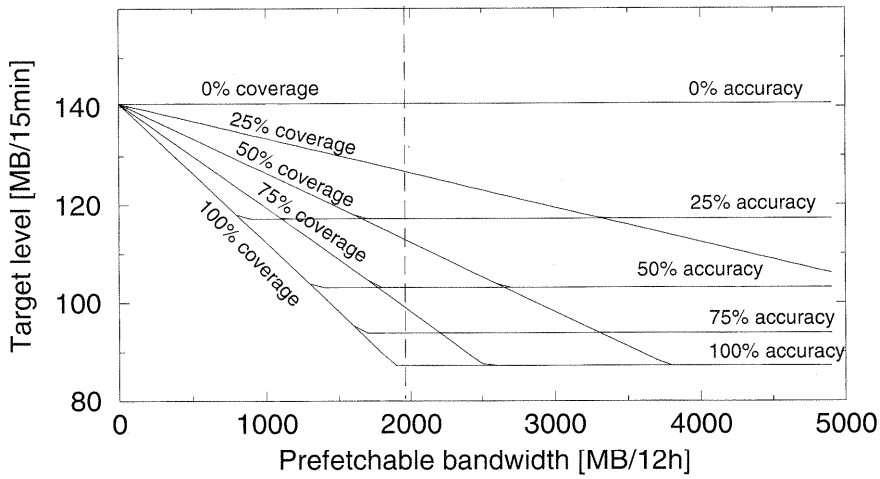


Figure 8: This shows Equation 9 as a function of B_{pre} for a given bandwidth usage profile at different prefetch performance levels ($B_{extra} = 2296.78$ MByte/12h, $B_{peak} = 4968.19$ MByte/12h, $L_m = 140.667$ MByte/15min). The x-axis represents the total prefetchable bandwidth of a peak period. The vertical dashed line indicates $B_{pre} = 1965.93$ MByte/12h, as it was measured in the given bandwidth usage profile. The y-axis represents the achievable target level L_t of bandwidth usage during the peark period. With a prefetch performance of zero accuracy and zero coverage, the target level is the same as the peak bandwidth usage level L_m. With 100% accuracy and coverage, and a prefetchable bandwidth of at least 1900 MByte/12h the best target level is 87.2 MByte/15min. The graph illustrates that coverage is the fraction of prefetched bandwidth that actually lowers the target level, and that accuracy is the maximum amount of prefetchable bandwidth that can be prefetched for a given B_{extra}.

4 The Use of Machine Learning for Finding Prefetch Strategies

The content as well as the traffic characteristics of the World-Wide Web is complex, heterogeneous and changes over time as new services become available and new protocol standards replace old ones. To perform well, prefetch algorithms have to be able to adapt to these changing patterns. Other areas of computer systems research with similar problem complexities and similar requirements for adaptability have successfully applied machine learning techniques (see for example [4]). We therefore investigate the use of machine learning techniques to automatically generate prefetching strategies. Automatic generation allows us to adapt prefetch strategies as often as every day.

4.1 Machine Learning

Machine learning can generally be divided into one-shot decision tasks (classification, prediction) and sequential decision tasks (control, optimization, planning). Sequential decision tasks are usually formulated as reinforcement learning tasks, where the learning algorithm is part of an agent that interacts with an "external environment." At each point, the agent observes the current state of the environment (or some aspects of the environment). It then selects and executes some action, which usually causes the environment to change state. The environment then provides some feedback to the agent. The goal of the learning process is to learn an action-selection policy that will maximize the long-term rewards received by the agent [8] (see [12] for an overview). One-shot decision tasks are usually formulated as supervised learning tasks, where the learning algorithm is given a set of input-output pairs (labeled data). The input describes the information available for making the decision and the output describes the correct decision [8]. As we demonstrate below, trace

data of web proxy servers and content on the web provide a wealth of labeled data. We will therefore focus on supervised learning.

The result of a learning task is a classification model (also called a *classifier*) which allows the classification of unseen data below a certain error rate. There are multiple classification formalisms available (see [21] for a short overview). Two common formalisms are decision trees and propositional rule sets.

The approach that we use for generating prefetch strategies is implemented as a tool called RIPPER [5] which efficiently produces and evaluates a classifier in form of a propositional rule set and which is freely available for non-commercial use at [6].

One of the most active areas of research in supervised learning has been the study of methods for constructing good *ensembles of classifiers*. An ensemble of classifiers is a set of classifiers whose individual decisions are combined in some way, usually by weighted voting (see [9] for an overview). Our results include performance data using an ensemble construction algorithm called ADABOOST [14, 13]. The technique is also called *boosting* and works roughly like this: Each classifier is constructed using a "weak learner" such as RIPPER. The difference between the individual classifiers is that they are trained on increasingly more difficult learning problems. The first classifier is learned by the original training data. The next learning problem is constructed by adding weight to the examples which are misclassified by the first classifier. This more difficult learning problem is used to train the next classifier. The examples misclassified by the second classifier receive additional weight in the next learning problem, and so on.

4.2 Training

In order to find a prefetch strategy, one has to find training data consisting of *positive evidence*, *i.e.* examples of objects that should be prefetched, and *negative evidence*, *i.e.* examples of objects that should not be prefetched. The question we are addressing in this section is to identify sources of training data.

Access log data from web proxy servers provides information about bandwidth usage and traffic characteristics, and defines the set of prefetchable objects by providing their reference and last modification time stamps. The set of prefetchable objects also defines the set of servers which provide prefetchable service. Thus, access logs provide positive evidence for learning server and object selection strategies.

Negative evidence is by definition not available in access logs. However, for the server selection problem we can approximate negative evidence based on the positive evidence in access logs. This is because over time the set of servers which serve prefetchables grows very slowly. This is reflected in figure 3 by the very small fraction of unseen-server-prefetchables. For each peak period, we can therefore approximate negative evidence for server selection with the set of all known servers minus the set of servers which served prefetchables.

Figure 3 also shows that most prefetchables are seen-server prefetchables, *i.e.* prefetchable objects that are referenced for the first time. This means for object selection that it is impossible to find a good approximation of negative evidence solely based on positive evidence because the set of known prefetchable objects grows relatively fast. Thus, we need to collect negative evidence from somewhere else.

There are multiple ways to acquire this information which differ in their impact on bandwidth consumption and their requirements on local services. For example, if the local site also runs a large search engine, the negative evidence can be derived from the search engine index as long as the search index contains information about textual as well as non-textual objects (see for example [3]). This approach has very little impact on bandwidth consumption. If no local search engine is available, a remote search engine could offer a service that allows querying for a very compact representation of the names and attributes of objects with certain properties. Finally, servers themselves could provide such a querying service in some well-known manner.

4.3 Training and Testing Methodology

Having none of the above services available, we built a "web robot" that scans a given set of servers and collects the name, size, MIME type, and age of each object. This approach has the greatest impact on bandwidth consumption because it requires the individual request and transmission of object headers and entire HTML objects.

To generate the training data we need to be able to compare object names in log data with object names found on scanned servers. Since Digital WRL traces are sanitized and do not provide this information, we collected our own data at the same site as the Digital WRL data was collected. Our access log data covers 14 days of full gateway traffic (Monday, 5/18/1998 - Sunday, 5/31/1998).

Near the end of the 14 day period, during Friday, 5/29/1998, we identified the top 320 prefetchable service group which serves about 45% of the total prefetchable bandwidth during the 11 day period prior to Friday. We then scanned these 320 servers during the weekend of 5/30-31/1998. The resulting scan contains information on 1,935,086 objects. Limited resources prevented us from scanning more than 320 servers and because of time constraints we were unable to completely scan these 320 servers. To estimate the relative size of the scan data sample, we assume that prefetchable objects are uniformly distributed in any scan data. Since we know the amount of prefetchable bandwidth from the access log data we can then approximate the scan data sample size by the fraction of the prefetchable bandwidth contained in the scan data. According to this approximation our scan data sample size is about 22% of the size of the entire content of the scanned servers.

The training data consists of 12 files, each of them covering one day. We now describe how we generated the training data. We start with the positive evidence: By using the same methodology as described in section 2.2 we identified prefetchable objects for each day of access log data after allowing two days to warm up the cache (thus the $14 - 2 = 12$ days of training data). We then encoded each prefetchable object by six attributes: (1) age, (2) MIME Type, (3) size, (4) server name, (5) top level domain, (6) the label that marks this entry as prefetchable, and (7) a weight proportional to the size. The first three attributes train object selection, the fourth and fifth attribute train server selection, and the weight represents the relative significance of an entry to the overall bandwidth consumption.

For the negative evidence we collected each day's no-miss-prefetchable objects from the scan data. Recall that no-miss-prefetchable objects are objects that meet all prefetchablility conditions as described in section 2.1 except the first condition, *i.e.* the object is not missed during the peak period of the current day. Each no-miss-prefetchable object is encoded and weighted in the same way as prefetchable objects except that the entry is labeled as non-prefetchable.

Note that we used the same scan data for each day of training data. The scan data is a (incomplete) snapshot of the server content during the last weekend of the measurement period. Thus we probably missed some no-miss-prefetchable objects that disappeared during the measurement period prior to the scan period. However, prefetchable and no-miss-prefetchable objects belong by definition to the more stable content of servers. Also, the generated training data consists mostly of negative evidence (see Figure 9). We are therefore confident that the quality of our training data suffers more under the incompleteness of the scan than under the omission of no-miss-prefetchable objects.

Another important aspect is the difference between the average size of negative and positive evidence. Figure 9 shows a larger gap between the size of positive and negative evidence than between the number of positive and negative examples. We found that the average size of objects in positive examples is 20,320 Bytes while the average size of in negative examples is 103,183 Bytes.

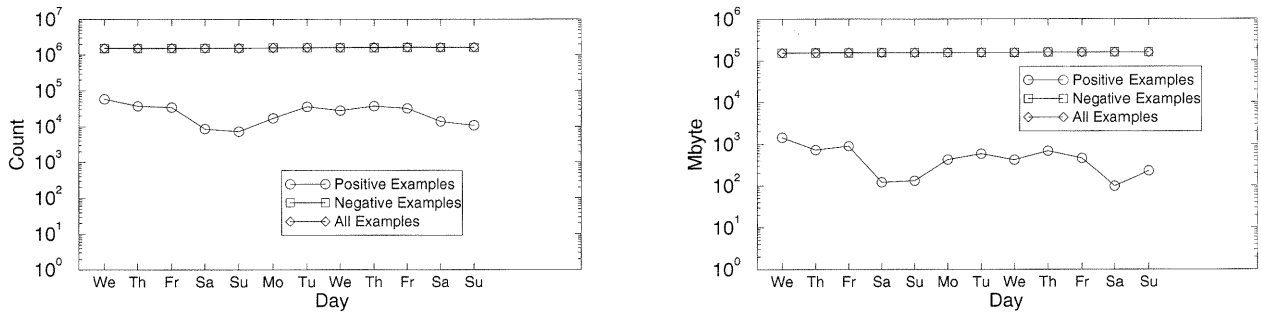


Figure 9: The positive and negative evidence components of each day's training data measured in number of examples and number of Bytes. The y-axis is log-scaled. The vast majority of training examples are negative. Over time the number and size of negative evidence remains almost unchanged. This indicates that most of the no-miss-prefetchable objects are older than the measurement period.

4.4 Experiments

We conducted two experiments: In the first experiment we applied RIPPER to the training data of the days 5/20/1998 - 5/30/1998. This produced eleven sets of rules. We tested each rule set against the training data of the next day (5/21/1998 - 5/31/1998). This results in prefetch performance data for 11 consecutive days, using a different prefetch strategy each day.

In the second experiment we used boosting to construct a 10 classifier ensemble using RIPPER as a weak learner. Since boosting takes significantly longer than the generation of a single classifier, we only generated one prefetch strategy based on the data of 5/20/1998 and tested its performance on the training data of the 11 remaining days.

To better distinguish between the results of these experiments we refer to the 11 prefetch strategies created by the first experiment as the "non-boosted prefetch strategies", and we call the prefetch strategy from the second experiment the "boosted prefetch strategy".

4.5 Results

Figure 10 shows the performance of machine learned prefetching strategies in terms of bandwidth and in terms of accuracy and coverage. For non-boosted prefetch strategies accuracy is particularly high. This means that the bandwidth used for prefetching is well invested. However coverage is generally low. One reason for this is that the prefetch strategies pick out smaller objects since the average size of positive examples is smaller than the size of negative examples: if the performance is measured based on number of examples instead of number of Bytes, the coverage is significantly higher.

Notice that the first weekend is Memorial Day weekend. HTTP traffic pattern during weekends and holidays are different from traffic patterns during work days. Therefore, Friday provides poor training data for Saturday, and Memorial Monday provides poor training data for Tuesday. This is reflected by relatively low accuracy on Saturday and Tuesday. A better strategy would be to use the previous Sunday as training day for a Saturday, and to use previous Friday as a training day for the first week day, in this case Tuesday.

The performance of the boosted prefetch strategy is particularly high on the first Thursday and Friday because of the proximity to the day of training. In terms of saved bandwidth the boosted prefetch strategy out-performs all non-boosted strategies but shows lower accuracy, particularly during the weekends. Since we only use one strategy which was learned on a week day the boosted strategy performs much better on Tuesday than the corresponding non-boosted strategy which was learned on a holiday.

Figure 11 shows the bandwidth smoothing effect of the learned prefetching strategies relative to the

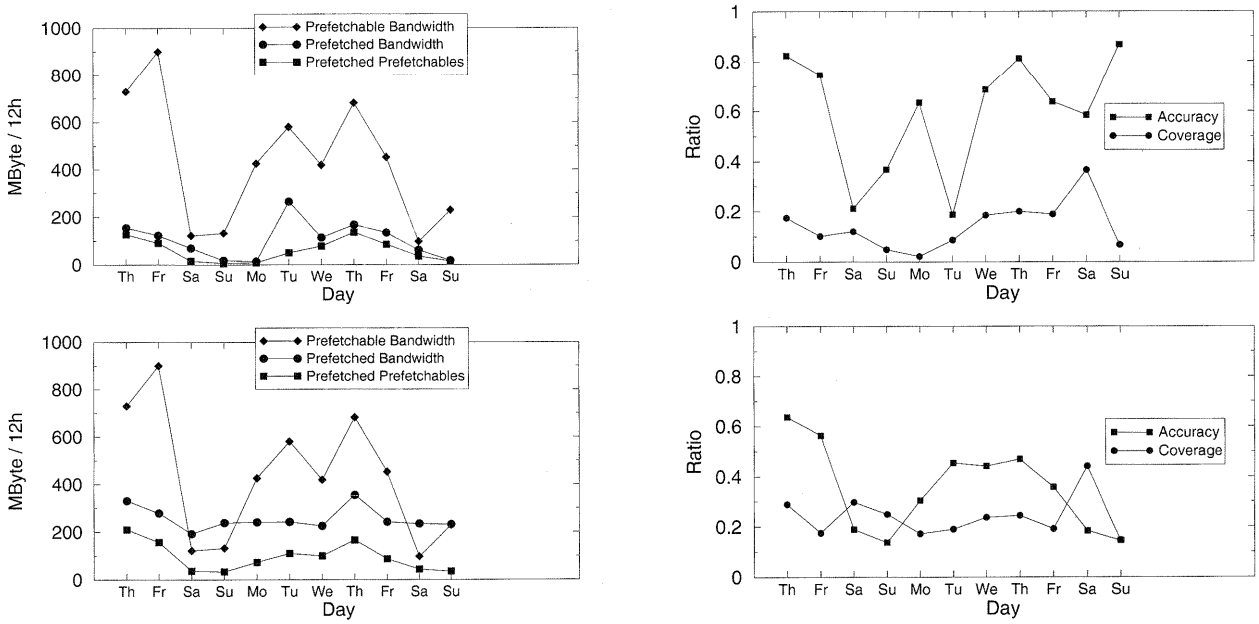


Figure 10: The upper two graphs show the performance of non-boosted prefetch strategies, each one trained on the day previous to the indicated day. The lower two graphs show the performance of the boosted prefetch strategy which was learned on the Wednesday prior to the first Thursday. "Prefetchable bandwidth" represents the total prefetchable bandwidth during the peak period of the indicated day. "Prefetched bandwidth" shows the total bandwidth prefetched during the off-peak period prior to the peak period of the indicated day. "Prefetched prefetchables" shows the total amount of bandwidth saved due to prefetching. The performance on Saturday and Tuesday (Monday was Memorial Day) is weak because of the different traffic pattern of week days and weekend/holidays. In terms of saved bandwidth the boosted prefetch strategy out-performs all non-boosted strategies but shows lower accuracy, particularly during the weekends.

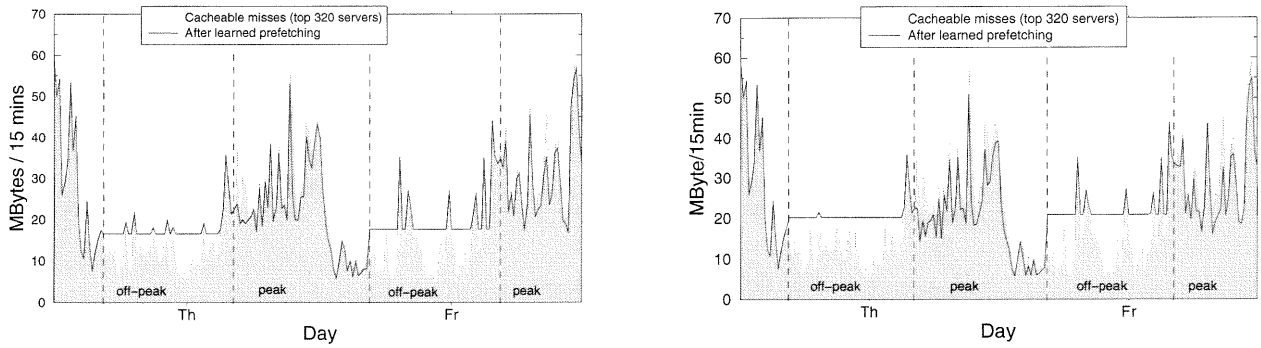


Figure 11: The impact of the learned prefetch strategies on bandwidth consumption (left without boosting, right with boosting). The x-axis marks the beginning of each day. The gray curve shows the bandwidth consumption profile of the top 320 servers. The black curve shows bandwidth consumption after prefetching. To show the impact of prefetching during off-peak hours we raised the off-peak bandwidth usage to a minimum level at which the difference between miss bandwidth and the raised level equals the prefetched bandwidth. White areas below the black curve show the extra bandwidth used for prefetching during off-peak periods. Grey areas above the black curve show the saved bandwidth during peak periods. The very left section of each graph is Wednesday's peak period which was only used for training and not for testing (*i.e.* no prefetching for Wednesday).

cacheable service of the top 320 servers on Thursday and Friday (for clarity we left out the other days but the results are similar). To simulate the effect of prefetching during off-peak hours we raised the off-peak bandwidth usage to a minimum level at which the difference between miss bandwidth and the raised level equals the prefetched bandwidth. The resulting increase in bandwidth usage is represented by white areas below the black curve. During peak periods the bandwidth savings are shown by grey areas above the black curve. No prefetching has been done for the left-most period of the graph (peak period of Wednesday).

The smoothing of peak bandwidth usage varies. At the beginning of the peak period of Thursday, two miss bandwidth peaks are completely cut off. The highest peak on Thursday is reduced by 10.5% using a non-boosted strategy (left graph) and nearly 15% using the boosted strategy (right graph). Table 1 shows that the observed smoothing effect matches the prediction of the model in section 3. The peak bandwidth of misses to all servers during Thursday is around 200M Byte/15 mins. Thus the reduction of the overall peak bandwidth usage level is around 3% to 4.5%. While the overall performance does not seem very impressive, the peak load reduction for the selected 320 servers is significant.

The generated prefetch strategies are complex. Nevertheless, we would like to understand them. What follows is a number of common properties across all non-boosted strategies. We will not analyze the boosted strategy because as of writing this paper only one strategy is available to us.

Each non-boosted strategy consists of a set of propositional rules which specify positive prefetching conditions. If none of these rules match, the default is to not prefetch. Figure 12 shows a comparison of the strategies. The number of rules in a strategy does not correlate to prefetch performance. Most rules in each strategy are server specific rules, *i.e.* they only apply to one server. Most notably, all prefetch strategies refer to only 39 hosts. Almost a third of all server-specific rules of all prefetch strategies refer to one and the same server which is a major Internet portal. 14 server-specific rules refer to the same major news service. It is also interesting that a significant number of rules are specific to JPEG images. All other objects types, including GIF images and HTML objects are only rarely part of rules. In fact, a large fraction of rules are not specific to any object type.

B_{peak}	=	1159.9	MByte / 12h
B_o	=	638.5	MByte / 12h
B_{pre}	=	728.9	MByte / 12h
L_m	=	59.1	MByte / 15min

prefetch strategy	L_t [M Byte / 15min] model	actual
not boosted		
$P_c = 0.18$, $P_a = 0.82$	52.6	52.9
boosted		
$P_c = 0.29$, $P_a = 0.64$	48.3	50.9

Table 1: Validation of the prefetch performance model introduced in secion 3. The profile parameters and the actual values are taken from the first Thursday of the measurement period.

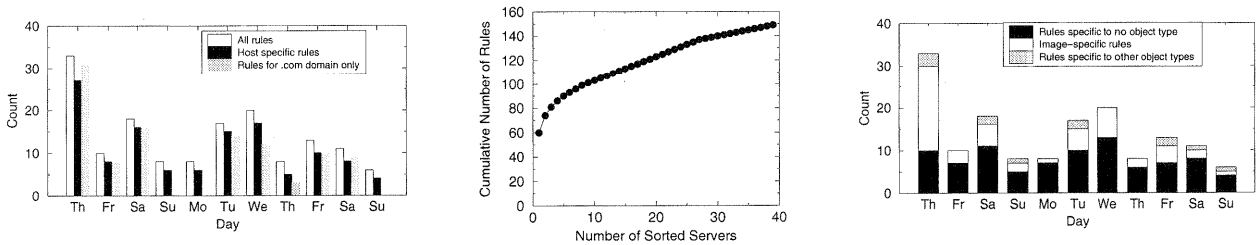


Figure 12: The left graph shows each strategy's total number of rules, the number of server-specific rules, and the number of rules specific to the ".com" top-level domain. All strategies consist mostly of rules that are specific to servers. There is a large overlap of host-specific rules and rules specific to the ".com" domain. The middle graph shows the cumulative distribution of all server-specific rules over all servers that occur in least one rule. The distribution is very skewed - about 40% of all server-specific rules refer to the same server. The right graph shows the number of rules specific to object types, in particular to images (almost all JPEG, very few GIF). There are a significant number of rules in each strategy that are not specific to any object type.

4.6 Discussion

The results show (a) and that there is considerable "prefetchable" data available for bandwidth smoothing and (b) that automated machine learning is a viable method to rapidly and automatically develop prefetch strategies.

The training of these strategies, require information obtained directly from web sites. Web site scanning consumes considerable bandwidth since at least the entire HTML content has to be down-loaded plus header information of images and other objects. Clearly, this time and resource consuming process would make our approach infeasible. However, a well-known query interface that allows clients to issue a query to a server or a search engine for information about objects that match certain properties would significantly reduce the overhead (see for example [3]). There are also a number of promising projects in the Web metadata community [19] which are interested in a compact representation of web content.

5 Related Work

We are not aware of any work that investigated real web traffic work loads in terms of shifting peak bandwidth usage to off-peak periods through prefetching; most work on prefetching focuses on immediate prefetching to reduce interaction latency.

In [7] Crovella and Barford show that bandwidth smoothing can lead to an overall reduction of queuing delays in a network and therefore to an improvement of network latency.

A server-initiated prefetching approach based on the structure of web objects and user access heuristics as well as statistical data is presented in [23]. Padmanabhan and Mogul present an evaluation of a server-initiated approach in which the server sends replies to clients together with "hints" indicating which objects are likely to be referenced next [20]. Their trace-driven simulation showed that their technique could reduce significantly latency at the cost of an increase in bandwidth consumption by a similar fraction. Padmanabhan and Mogul's approach is based on small extensions to the existing HTTP protocol. A similar study but with an idealized protocol was performed by Bestavros [2] in which the author proposes "speculative service" in which a server sends replies to clients together with a number of entire objects. This method achieved up to 50% reduction in perceived network latency.

The greatest challenge in prefetching is to achieve efficient prefetching. In [24], Wang and Crowcroft define prefetching efficiency as the ratio of prefetch hit rate (the probability of a correct prefetch) and the relative increase of bandwidth consumption to achieve that hit rate. Assuming a simplifying queuing model (M/M/1) they show an exponential relationship between the bandwidth utilization of the network link and the required prefetching efficiency to ensure network latency reduction. Crovella and Barford propose "rate controlled prefetching" in which traffic due to prefetching is treated as a lower priority than traffic due to actual client requests [7].

6 Summary and Future Work

We showed that using extra network resources to prefetch web content during off-peak periods can significantly reduce peak bandwidth usage and that these effects are additive to effects of traditional demand-based caching. We presented a mathematical model on how to calculate the benefit of bandwidth-smoothing a particular bandwidth usage profile.

We also showed that machine learning is a promising method to automatically generate prefetch strategies. These strategies were able to prefetch up to 40% of the prefetchable bandwidth and do so without wasting significant bandwidth.

We are in the process of verifying and refining the presented models. We are also experimenting with various machine learning techniques to increase learning performance as well as reduce the duration of the learning process. We are currently working on a technique that increases prefetch performance by finding the optimal balance between accuracy and coverage. Our results above show that even with prefetching there is still a large amount of extra bandwidth available during off-peak periods. Until this extra bandwidth is used we can accept lower accuracy to increase coverage. The above results are based on a machine learning algorithm that assumes equal penalty for false positives and false negatives (the *loss ratio* is 1). By increasing the penalty for false negatives and reducing the penalty for false positives one reduces the accuracy in favor of coverage. We are therefore interested in the relationship between the loss ratio and prefetch performance.

While non-boosted learning of one prefetch strategy on a dedicated workstation takes only a couple of hours, boosted learning took in our case seven days (generating 10 rule sets). One way to speed up the learning process is to increase the abstraction of training data without loosing relevant pattern information. For example, size and age of our training examples are real values. Grouping these values in a well chosen manner can significantly speed up learning.

The current framework assumes *a priori* definition of peak and off-peak periods. Figure 10 and 11 show that the actual bandwidth profile does not always match these fixed periods and that a more flexible notion of peak and off-peak periods would increase the amount of extra bandwidth available for prefetching.

References

[1] Arnold O. Allen. *Probability, Statistics, and Queueing Theory: with Computer Applications (2nd edition)*. Academic Press, San Diego, CA, 1990.

[2] Azer Bestavros. Speculative Data Dissemination and Service to Reduce Server Load, Network Traffic and Service Time for Distributed Information Systems. In *ICDE'96: The 1996 International Conference on Data Engineering*, New Orleans, Louisiana, March 1996.

[3] Krishna Bharat, Andrei Broder, Monika Henzinger, Puneet Kumar, and Suresh Venkatasubramanian. The connectivity server: fast access to linkage information on the web. *Computer Networks and ISDN Systems*, 30(1-7), April 1998. Special Issue on Seventh International World-Wide Web Conference, April 14-18, 1998, Brisbane, Australia.

[4] Brad Calder, Dirk Grunwald, Michael Jones, Donald Lindsay, James Martin, Michael Mozer, and Benjamin Zorn. Evidence-based static branch prediction using machine learning. *ACM Transactions on Programming Languages and Systems*, 19(1):188–223, January 1997.

[5] William W. Cohen. Fast Effective Rule Induction. In *Machine Learning: Proceedings of the Twelfth International Conference (ML95)*, 1995.

[6] William W. Cohen. The RIPPER Rule Learner. Available on the World-Wide Web at `http://www.research.att.com/~wcohen/ripperd.html`, November 25 1996.

[7] Mark Crovella and Paul Barford. The Network Effects of Prefetching. Technical Report 97-002, Boston University, CS Dept., February 7 1997.

[8] Thomas G. Dietterich. Machine Learning. *ACM Computing Surveys*, 28(4es), December 1996. Available on the World-Wide Web at `http://www.acm.org/pubs/citations/journals/surveys/1996-28-4es/a3-dietterich/`.

[9] Thomas G. Dietterich. Machine-Learning Research: Four Current Directions. *AI Magazine*, 18(4):97–136, Winter 1997.

[10] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, and Jeffrey Mogul. Rate of Change and other Metrics: a Live Study of the World-Wide Web. In *Usenix Symposium on Internet Technologies and Systems (USITS)*, Monterey, CA, USA, December 8-11 1997. Usenix.

[11] Brad Duska, David Marwood, and Michael J. Feeley. The Measured Access Characteristics of World-Wide Web Client Proxy Caches. In *Usenix Symposium on Internet Technologies and Systems (USITS)*, Monterey, CA, USA, December 8-11 1997. Usenix.

[12] D. H. Fisher, M. J. Pazzani, and P. Langley. *Concept Formation: Knowledge and Experience in Unsupervised Learning*. Morgan Kaufmann, San Mateo, CA, 1991.

[13] Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, Berlin, 1995. Springer Verlag.

[14] Y. Freund and R. E. Schapire. Experiments with a New Boosting Algorithm. In L. Saitta, editor, *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, San Francisco, CA, 1996. Morgan Kaufmann.

[15] Steven D. Gribble and Eric A. Brewer. System design issues for Internet Middleware Services: Deductions from a large client trace. In *Usenix Symposium on Internet Technologies and Systems (USITS)*, Monterey, CA, USA, December 8-11 1997. Usenix.

[16] Thomas Kroeger, Jeffrey Mogul, and Carlos Maltzahn. Digital's Web Proxy Traces. Available on the World-Wide Web at `ftp://ftp.digital.com/pub/DEC/traces/proxy/webtraces.html`, October 1996.

[17] Carlos Maltzahn, Kathy Richardson, and Dirk Grunwald. Performance Issues of Enterprise Level Proxies. In *SIGMETRICS '97*, pages 13–23, Seattle, WA, June 15-18 1997. ACM.

[18] Jeffrey C. Mogul, Fred Douglis, Anja Feldman, and Balachander Krishnamurthy. Potential benefits of delta-encoding and data compression for HTTP. In *SIGCOMM '97*, Cannes, France, September 1997. ACM.

[19] International Federation of Library Associations and Institutions (IFLA). Digital Libraries: Metadata resources page. Available on the World Wide Web at `http://www.nlc-bnc.ca/ifla/II/metadata.htm`, August 31 1998.

[20] Venkata N. Padmanabhan and Jeffrey C. Mogul. Using Predictive Prefetching to Improve World-Wide Web Latency. In *SIGCOMM'96*, pages 22–36. ACM, July 1996.

[21] J. Ross Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[22] Alex Rousskov. On Performance of Caching Proxies. Available on the World-Wide Web at `http://www.cs.ndsu.nodak.edu/~rousskov/research/cache/squid/profiling/papers/sigmetrics.html`, 1997.

[23] Stuart Wachsberg, Thomas Kunz, and Johnny Wong. Fast World-Wide Web Browsing Over Low-Bandwidth Links. Available on the World-Wide Web at `http://ccnga.uwaterloo.ca/~sbwachsb/paper.html`, 1996.

[24] Zheng Wang and Jon Crowcroft. Prefetching in World-Wide Web. Available on the World-Wide Web at `http://www.cs.ucl.ac.uk/staff/zwang/papers/prefetch/Overview.html`, January 30 1996.