

The First IRCache Web Cache Bake-off
The Official Report

Alex Rousskov

Duane Wessels

Glenn Chisholm

<http://bakeoff.ircache.net/>

polyteam@ircache.net

April 2, 1999

Abstract

A three-day benchmarking “bake-off” for Web proxy caches was held during the third week of March, 1999. Our group tested nine proxy caches from six different organizations. In this report, we summarize performance data collected during these tests and analyse the results.

Contents

1	Introduction	4
1.1	Motivation	4
1.2	Bake-off Participation	5
1.3	Terminology	5
1.4	How (not) to Read this Report	6
1.5	Where to find more information	6
2	The Rules	7
2.1	Last-minute Changes	8
3	Web Polygraph	10
3.1	The Bake-off Workload	10
3.2	PolyMix #1 Workload Summary	14
4	Testing Environment	15
4.1	Location	15
4.2	Schedule	15
4.3	Polygraph Machines	15
4.4	Monitoring Machines	16
4.5	Time Synchronization	16
4.6	Network Configurations	16
4.7	Filling the Caches	17
4.8	Running Polygraph	17
5	Comparison of Results	18
5.1	Throughput	18
5.2	Response time	18
5.3	Hit Ratio	20
5.4	Price	22

5.5	Price/Performance	23
5.6	Failed Runs	25
5.7	Summary	26
6	Future Work	27
7	Conclusions	28
A	IBM	30
A.1	Equipment	30
A.2	Results	30
B	InfoLibria-L	31
B.1	Equipment	31
B.2	Configuration	31
B.3	Results	32
B.4	Polyteam Comments	33
B.5	Participant Comments	33
C	InfoLibria-S	35
C.1	Equipment	35
C.2	Configuration	35
C.3	Results	36
C.4	Polyteam Comments	37
C.5	Participant Comments	37
D	Network Appliance-L	39
D.1	Equipment	39
D.2	Results	39
E	Network Appliance-S	40
E.1	Equipment	40
E.2	Results	40
F	Novell/Dell-L	41
F.1	Equipment	41
F.2	Configuration	41
F.3	Results	42
F.4	Polyteam Comments	44

F.5	Participant Comments	44
G	Novell/Dell-S	45
G.1	Equipment	45
G.2	Configuration	45
G.3	Results	46
G.4	Polyteam Comments	47
G.5	Participant Comments	47
H	Peregrine	49
H.1	Equipment	49
H.2	Configuration	49
H.3	Results	50
H.4	Polyteam Comments	52
H.5	Participant Comments	52
I	Squid	53
I.1	Equipment	53
I.2	Configuration	53
I.3	Results	54
I.4	Polyteam Comments	56
I.5	Participant Comments	56
J	Verifying Benchmarking Environment	57
J.1	Running <i>netperf</i>	57
J.2	No-proxy Tests	57
K	FreeBSD Tweaks	59

Chapter 1

Introduction

1.1 Motivation

A number of Web cache¹ products have entered the marketplace within the last couple of years. One way that vendors distinguish their products from their competition is with claims of performance. Unfortunately, these performance claims are difficult, if not impossible to reproduce and verify.

Many of the companies developed their own benchmarking software. While these benchmarking programs are generally available to potential customers, the software was developed internally to the company, without outside influence. Thus, a given benchmark suite may be designed to show off the features of one product, but not another. It is generally deemed unacceptable to compare company A's cache with company B's cache by benchmarking with A's benchmarking software. The quality of some existing benchmarks is also questionable as many of them are simply "load generators."

To alleviate this problem, we have developed a freely-available high-performance cache benchmarking package called Web Polygraph [1]. Polygraph can generate about 1000 requests per second between a client-server pair on a 100baseT network. Furthermore, Polygraph allows you to specify a number of important workload parameters such as hit ratio, cachability, response sizes, and server-side delays.

Using a single tool to benchmark multiple products does not necessarily allow for legitimate comparison of results. A number of other factors may affect benchmarking results. In particular, the operating system's TCP stack needs to be tuned to support a high connection load. When comparing results, we feel that it is very important to minimize differences in the testing environment. A seemingly minor difference in configuration may cause a very significant change in performance. Hardware (processors, RAM, network cards) should be identical if at all possible. Software versions and configurations should be identical as well. We strive for a lot of detail regarding our benchmarking environment and configuration so that the results of our tests can be reliably reproduced by others.

In order to compare the performance of different caching products under identical conditions, we proposed to hold a "bake-off." The basic idea is that everyone comes together for a few days in one location and puts their products through a series of tests. Because all tests occur at the same time and place, under identical conditions, comparisons can be made between the participant's results. Another less obvious, but very important reason for having the bake-off is to bring the caching

¹For the purpose of this report, we use terms "Web cache," "proxy," "proxy cache," etc. interchangeably.

vendors together to agree on a widely-accepted methodology for benchmarking Web caches.

1.2 Bake-off Participation

The Bake-off was first announced during a Web caching BOF at the 14th NANOG² meeting held November 8–10, 1998, in Atlanta, Georgia. A mailing list was soon established, and by January we knew that CacheFlow, Cisco, Entera, InfoLibria, Inktomi, Network Appliance, and Novell were all interested in participating. In February, we held a planning meeting with representatives from each of these companies. Later on in the process, IBM and the University of Wisconsin also became bake-off participants.

As the schedule progressed toward the bake-off date, these companies were required to make a commitment to have their products benchmarked. For them, the decision may have been difficult. If they come to the bake-off, but do not perform well, their company stands to lose revenue and potential customers. Recall that, until now, verifying and comparing vendor's performance claims has been very difficult. Also note that, as we will explain later, vendors may come to the bake-off and later choose to not disclose their results.

As we learned all too well, much of the caching industry is driven by the forces of marketing. Thus, in addition to worrying about their own performance, vendors seem to spend even more time worrying about how they will stack up against their competitors. Some companies were concerned that publication of their results is a competitive disadvantage when the competition's performance is unknown, or undisclosed.

In the end, CacheFlow, Cisco, Entera, and Inktomi decided to not bring their products to the bake-off. Certainly we are disappointed by their choice. We feel that the benchmarking results are more useful when there are more results to compare. At the same time, however, we take it as a compliment that our benchmark was taken very seriously, and perhaps even feared, by these companies. We understand that in most cases, the decision to participate was made by the highest levels of management. It seems we certainly have their attention.

IBM, InfoLibria, Network Appliance, Novell, University of Wisconsin, and NLANR/IRCACHE brought their products to the bake-off in Redwood City. There we all had 72 sleepless hours of work, surprises, frustrations, and, most importantly fun!

Unfortunately, after the bake-off was over, IBM and Network Appliance decided not to disclose their results, as allowed by the bail-out provision described in Chapter 2. The results of the remaining four participants (six caches total) are presented in this report.

1.3 Terminology

Throughout this report we use a few terms that have specific meaning for the bake-off. A *vendor* is an organization that has a caching product. To simplify the terminology, all commercial, non-profit, virtual, etc. organizations are labeled as “vendors.” A vendor is allowed to bring more than one product to the bake-off. Each product that a vendor brings counts as one *participant*. We have one benchmarking *harness* (bench) for every participant.

²<http://www.nanog.org/>

1.4 How (not) to Read this Report

We strongly caution against drawing hasty conclusions from these benchmarking results. Our report contains a lot of performance numbers and configuration information; take advantage of it. Since the tested caches differ a lot, it is tempting to draw conclusions about participants based on a single performance graph or pricing table. We believe such conclusions will virtually always be wrong. Here are a few recommendations to prevent misinterpretation of the results.

1. Always read Polyteam and Participant Comments sections in the Appendices.
2. Compare several performance factors: throughput, response time, and hit ratio. Weight each factor based on your preferences.
3. Do not overlook pricing information in Section 5.4 and price/performance analysis in Section 5.5.

Our benchmark addresses only the performance aspects of Web cache products. Any given cache will have numerous features that are not addressed here. For example, we think that manageability, reliability, and correctness are very important attributes that should be considered in any buying decisions.

The remainder of this report is organized as follows. In the following chapter we will discuss the rules of the bake-off. We talk about Web Polygraph in chapter 3, and in particular, the workload model that we used for running the experiments. Chapter 4 has details about the bake-off environment, including hardware and software configurations for the Polygraph clients and servers. The best part—the results—are presented in chapter 5. We close with thoughts for the future and conclusions.

1.5 Where to find more information

IRCache maintains the Official Bake-off Site, `bakeoff.ircache.net`, where this report and detailed Polygraph log files from the bake-off are stored. All bake-off information at the Official Site is freely available.

There are no other official sources of bake-off results.

Documentation and other information about Web Polygraph is available at `http://polygraph.ircache.net/`.

Chapter 2

The Rules

The bake-off rules were defined and agreed upon one month prior to the first day of benchmarking. We will not include the full set of rules here. Please refer to <http://bakeoff.ircache.net/bakeoff-01/doc/rules.html> for the original rules document. Here, we will highlight some of the more interesting and controversial provisions.

Availability

Following examples from other types of benchmarking, we adopted a rule that the cache products (for which results are published) must be available to the public within six months from the end of the bake-off. This means that the products described here should be available to consumers by at least mid September, 1999. Furthermore, the products are supposed to be available for the prices given in this report. Of course, both of these rules are difficult to actually enforce. If you are aware of a tested cache that is not available within this time period, we would like to hear from you.

TIME_WAIT

For busy TCP servers, the “time wait” value for recently closed sockets can adversely affect performance. For example, TCP generally has 64,000 port numbers available on a single system. A server that handles 1,000 connections per second will cycle through all port numbers in 64 seconds. If the time wait value is larger than 64 seconds, new connections will be blocked until the old port numbers time out. For the bake-off, everyone agreed to use a 60 second time wait value on the caches. This corresponds to a 30 second Maximum Segment Lifetime (MSL). It is difficult for us to detect non-compliance of this rule. In our experience, some TCP implementations allow port reuse of sockets in the TIME_WAIT state under certain conditions.

Fill Time

We require that the cache must be entirely fillable within eight hours. For the real benchmarking runs, we require full caches because we want to test steady-state conditions that occur in practice. Due to time constraints, we can not spend too much time filling up the cache. This rule also limits the size of a cache that may participate in the bake-off. For example, a cache that fills at a rate of 100 objects per second will store 2.8 million objects over an eight hour period. Given a mean object size of 13 KB, this corresponds to a cache size of 36.4 GB. For more information, see Section 4.7.

Repeats

We allow any benchmark run to be repeated, if time permits. Our primary motivation for this rule is to allow participants to complete failed experiments. However, the rule does not prohibit them from also repeating successful runs. Given two or more successful runs with identical parameters, the participant can choose which result to report.

Costs

We include the price of networking equipment in the participants' costs. Initially we wanted all participants to use identical networking hardware and configurations. We quickly discovered that no single configuration would work for all participants. Some want layer-four connection hijacking. Some want to use WCCP, and thus require a router. Powerful networking equipment may be able to shift some load from the cache, onto the network, thus giving a participant an unfair advantage. We added the networking equipment to the participant cost primarily to discourage the vendors from bringing ultra-high-end switches and routers.

Bail-out

The bail-out provision gives a participant the option to not disclose their results. Thus, if a vendor comes to the bake-off and experiences some unexpected problems, they are not forced to share their results. We place no restrictions on their right to take this option, and do not comment on their decision.

Participant Comments

Participant who disclose their results are allowed to include their own comments in this report. We impose no restrictions on the content or nature of these comments.

Referencing Bake-off Results

Finally, we are concerned that some bake-off results may be quoted out-of-context without conveying the whole picture. Thus, any work that is derived from, or uses any of the bake-off results, or this report, must include the following reference to our official site:

Wessels, Rousskov, Chisholm, The First IRCACHE Web Caching Bake-off. In Proceedings of the Fourth Web Caching Workshop, San Diego, California, USA, April 1999. Raw data and independent analysis at <http://bakeoff.ircache.net/>

2.1 Last-minute Changes

A number of issues and questions came up during the bake-off itself.

Launch Window

One participant was worried that the start of a run caused a sudden, strong increase in load. The request rate goes from zero to “full blast” in a very short time period, and caused the participant’s cache to fail. By adding `--launch_win 60sec` to the *polycft* command line, the load increases gradually, over the first 60 seconds of the run. With this option, the participant’s cache handled the load better. All bake-off participants agreed to add this option to all Polygraph runs.

Configuration Changes

An interesting issue arose regarding whether or not we should allow participants to change their configuration during the bake-off. Our rules clearly state that “participants are not allowed to change the proxy configuration or tuning parameters during the test phase (from the start of the first test till the end of the last test).” However, after completing the full set of tests, one participant wanted to change the configuration of a switch port and re-run all the tests to get better results. A case was made that a participant should be allowed to do anything within the allotted time to achieve maximum performance.

We could not agree on what to do if the new experiment fails, or if the results are worse than before. Should both results be presented? Only the best results? After meeting, the group decided to not allow any changes at all to the cache or the networking equipment, unless the participant was willing to throw out all previous results and start over from the very beginning.

Increasing Maximum Request Rates

Every participant was required to tell us the maximum request rate that their cache can handle. After successfully achieving this rate, one vendor wanted to increase their stated maximum. We took issue with that because if a participant is allowed to continually increase their stated maximum, they may be able to cautiously approach it at small increments. That approach is much less risky than choosing a single maximum and sticking to it. After discussion, the bake-off participants agreed to allow each participant one more opportunity to change their maximum stated request rate.

Chapter 3

Web Polygraph

Web Polygraph is a high-performance proxy benchmark. Polygraph is capable of generating various Web proxy workloads that either approximate real-world traffic patterns, or are designed to stress a particular proxy component. Designed with the bake-off needs in mind, Polygraph is able to generate complex, high request rate workloads with negligible overhead. Web Polygraph has been successfully used to debug, tune, and benchmark many caching products.

The Polygraph distribution includes two programs: *polyclt* and *polysrv*. Poly-client (-server) emits a stream of HTTP requests (responses) with given properties. The requested resources are called *objects*. URLs generated by Poly-client are build around *object identifiers* or *oids*. In short, oids determine many properties of the corresponding response, including response content length and cachability. These properties are usually preserved for a given object. In other words, a response for an object with a given oid will have the same content length and cachability status regardless of the number of earlier requests for that object.

A Polygraph run has three distinct phases. The first (*warm-up*) phase lasts for 10% of the duration of a run. Most measurements collected during the warm-up phase are usually ignored. The data from the second (*measurement*) phase is used for the results. Finally, an optional cool-down phase of a given length completes the run. Cool-down phase measurements are usually ignored.

Polygraph logs a lot of statistics, including: response rate, response time and size histograms, achieved hit ratio, and number of transaction errors. Some measurements are collected at one second intervals, while others are collected every five minutes or just once per phase.

For the bake-off tests, we use version 1.0p5 of Web Polygraph.

3.1 The Bake-off Workload

One difficult part of this benchmark (and indeed most) was to develop the proper workload. Real-world Web traffic is incredibly complicated, both to understand and to simulate. Many workload attributes are well understood by themselves, but not when combined with each other. For example, we have a clear idea of real-world object size distributions. But how does object size combine with popularity? Are popular objects larger, smaller, or the same as unpopular ones?

Web Polygraph models several key aspects of real Web traffic. We realize that the model we used is not complete, but rather than wasting months or years on developing a comprehensive model, we feel it is better to model the important parameters now and further extend the model later.

Our model addresses the following characteristics of Web traffic:

- Reply sizes
- A mixture of cachable and uncachable responses
- Server-side latencies
- A mixture of cache hits and cache misses
- Object popularity
- Request rates and interarrival times

Noticeably absent from Polygraph traffic model are:

- Persistent connections
- DNS-lookup latencies
- Real content (HTML, images, etc)
- Aborted requests
- Cache validation (IMS requests)
- Forced cache validations (reloads)
- Distributions of Last-modified and Expires timestamps.

Reply Sizes

We use a relatively simple distribution for object reply sizes. For a given object, the reply size is chosen from an exponential distribution with a mean of 13 KBytes. We selected 13 KBytes as the mean because that is the number that we observe on real Web caches.

The exponential distribution with a mean of 13,312 has a median of 9,227 bytes. Real traffic distributions have medians closer to 4,000 bytes. Hence, Polygraph workload creates somewhat heavier traffic than proxies may be subjected to in the real world.

The reply size depends only on the oid. Thus, the same oid always has the same reply size.

Cachable and Uncachable Replies

Real Web traffic obviously has a mix of cachable and uncachable responses. For the bake-off, we used a cachability factor of 80%. That is, 80% of the Polygraph client requests are for cachable objects, while the remaining 20% are for uncachable objects. Because the cache (hopefully) stores the cachable objects, the Polygraph *server* ends up serving more than 20% uncachable objects over time.

The real world cachability varies from location to location. We have chosen 80% as a typical value that represents many common environments.

A cachable response includes the following HTTP reply header:

```
Cache-Control: public,max-age=31536000
```

An uncacheable response includes the following HTTP reply header:

```
Cache-Control: private,no-cache
```

Note that Polygraph 1.0p5 does not generate a *Last-Modified* reply header. Some caches may not store responses unless this header is present.

Cachability depends only on the oid. The same oid is always cacheable, or always uncacheable.

Server-side Latencies

We use server-side delays to simulate wide-area Internet delays. When the Polygraph server receives a request, it waits some amount of time before writing any part of the reply message. For each request, this “think time” is selected from a normal distribution with a 3 second mean, and a 1.5 second standard deviation. There are no artificial delays between consecutive socket writes, only this initial “think” delay.

Our simulation is useful for several reasons. First of all it utilizes more of the cache’s resources for the same request rate. The server delay increases the total time for the request, so the cache has more total TCP connections opened at one time.

The server-side delays also allow us to observe the cache’s hit ratio in its response time numbers. Usually, a cache hit will have a fast response time, and a cache miss will have a slow response time. If a cache’s hit ratio decreases as the load increases, the response time may increase. The converse is not necessarily true, however. Just because response time increases, that does not mean the hit ratio decreases.

The server think time does not depend on the oid. Instead, it is randomly chosen for every request.

Cache Hits and Misses

For the bake-off workload, we told Polygraph to give us a constant hit ratio of 55%. This number was agreed upon in February by a group of potential bake-off participants as representative of a typical cache’s hit ratio.

Polygraph enforces the desired hit ratio by requesting objects that have been requested before, and *should* have been cached. There is no guarantee, however, that the object is in the cache. Thus, our parameter (55%) is an upper limit. The hit ratio achieved by a proxy may be lower if a proxy does not cache some cacheable objects, or purges previously cached objects before the run completes.

Object Popularity

Applying known object popularity models to the bake-off workload turned out to be more difficult than we expected. Recent research [2], [3] has shown that object popularity in proxy cache logs follows a Zipf-like distribution. Analysis of our own log files confirms this.

When we used a Zipf popularity model in Polygraph, however, the influence of other parameters (such as hit ratio) resulted in too many memory hits¹ on the cache. In other words, the “hot set”

¹A memory hit is a cache hit for an object that was in the cache’s RAM. Because it is in memory, it is much, much faster than if it needed to be read from disk

was too small and could easily be held in the cache’s memory. A large percentage of memory hits is bad because a cache’s performance really depends upon its disk system, not the memory. The Zipf model did not sufficiently exercise a cache’s disk system for cache hits.

Instead of Zipf, we ended up using a “uniform” distribution. Specifically, when Polygraph needs to generate a cache hit, the next object is randomly selected from a uniform distribution of previously requested objects. Thus, we still have some notion of popularity because the objects selected at the beginning of the run will be requested more often than later objects. Figure 3.1 compares popularity characteristics of three different traces. Each trace consists of 1.5 million URLs. The *real* trace is taken from a real Web cache (sv.cache.nlanr.net) operated by us. The *poly-zipf* trace is generated from Polygraph using the `--pop_model zipf` command line option. The *poly-unif* trace is also generated from Polygraph, but with the `--pop_model unif` option.

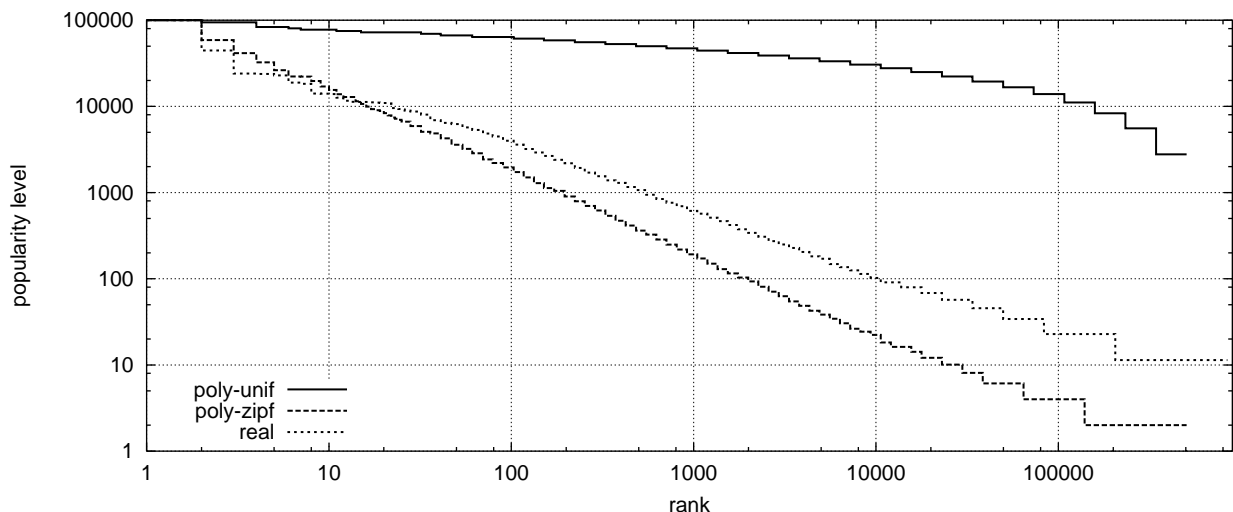


Figure 3.1: Comparison of popularity models

Note that the shape of the *real* and *poly-zipf* traces are very similar, perhaps differing only in their slope, or rather their exponent. The *poly-unif* trace, on the other hand, is much different. It lies above the Zipf traces, and therefore has a much broader popularity. The *poly-unif* trace has 100,000 objects that are requested 10,000 times or more. The Zipf traces, however, have only 10 objects that are requested 10,000 times or more.

Request Submission

Web Polygraph supports two different request submission modes. For the bake-off, we used the constant request rate model to approximate real-world conditions. In this mode, Polygraph clients emit a Poisson request stream at a given average rate. The request rate is, of course, not truly constant at small time scales, but has a constant average over large time periods (tens of seconds). Some fluctuations in the rate do exist, as in the real world traffic.

The request submission rate does not depend on the reply rate. If a cache is able to handle the offered rate, the average reply rate will be the same as the request rate. If the offered rate is too high, either the cache or Polygraph machines will run out of resources, and the corresponding experiment will be reported as “failed.”

3.2 PolyMix #1 Workload Summary

We named the bake-off workload “PolyMix #1.” It is characterized as follows:

- Reply sizes exponentially distributed with 13 KByte mean
- 20% uncacheable responses
- Server-side “think-time” delays normally distributed with a 3 second mean and 1.5 second standard deviation
- 55% document hit ratio
- “Uniform” popularity model
- “Constant Request Rate” model

We specify this model to Polygraph with the following command line arguments:

```
% polysrv --rep_sz exp:13KB --xact_think norm:3sec,1.5sec
```

```
% polyclt --rep_cacheable 80p --dhr 55p --pop_model unif --robots 1 --req_rate N/sec
```

Chapter 4

Testing Environment

4.1 Location

The bake-off was held at the offices of Vixie Enterprises. Paul Vixie was kind enough to lend us the use of his large warehouse area for approximately one week. In this warehouse we had plenty of room to set up all the necessary equipment.

This location was partly chosen because it is in the Bay Area, where a number of caching vendors are also located. Ironically, no participants that disclosed their results came from the area. We are considering holding the next bake-off in Antarctica.

4.2 Schedule

The duration of the bake-off was three days. The first day was used for testing the networking equipment and cluster configurations. By the end of the first day, most participants were filling up their caches. The filling process was allowed to complete overnight.

On the second day, we started PolyMix #1 runs. Theoretically, there was enough time to finish all required PolyMix #1 runs on the second day. If all tests run sequentially without problems, it takes just under 12 hours.

The third day was our safety net, in case some runs were not completed previously. Participants also had the option of repeating some runs if needed.

4.3 Polygraph Machines

We rented 80 PC's for use as Polygraph clients and servers. These 80 machines were Compaq Deskpro EN systems, each with a 233 MHz Pentium II CPU, 128 MB of RAM, an Intel Etherexpress PRO/100+ fast ethernet card, and an IDE disk.

We used FreeBSD-2.2.8 as the operating system for the Polygraph clients and servers. Thanks to FreeBSD's customizable boot floppy and ability to install over a network, we developed a procedure to easily and quickly load FreeBSD on all of the 80 PC's. To support high request rate workloads, FreeBSD requires some customizations that were also automated. Appendix K details our changes to the standard FreeBSD system for use during the bake-off.

The number of Polygraph machines varies for each participant’s cache. Peak request rates vary a lot among caching products. Thus, each participant informed us how many Polygraph client-server pairs were needed to drive their cache at its maximum capacity.

Each client-server pair is tightly coupled such that a Poly-client sends requests to only one Poly-server. We found that a single client-server pair can maintain about 350 requests per second for the PolyMix #1 workload. During the bake-off, we never used more than 250 requests per second for reported tests.

4.4 Monitoring Machines

Each cluster had an extra PC connected to the harness network. That monitoring PC was used to start Polygraph runs, display run-time statistics, and collect logs after the completion of a run. Monitoring stations proved to be handy for monitoring and diagnostic purposes, and most participants found run-time statistics very useful.

4.5 Time Synchronization

We ran a *xntpd* time server on each monitoring PC. The monitoring PC’s were synchronized periodically with a designated server maintained by Polyteam. Polygraph servers and clients were synchronized with monitoring machines before each test. The procedure allowed for a zero-overhead (as far as Polygraph tests were concerned) synchronization of all machines. Accurate time synchronization was not necessary for the bake-off workload, but deemed to be essential for later log analysis.

4.6 Network Configurations

Each test harness, or cluster, consists of Polygraph machines, the participants proxy cache(s), and a network to tie them together. For this bake-off, the networking equipment falls under the participant’s domain. That is, each participant is responsible for providing the networking equipment need to connect the Polygraph machines to the caches. Furthermore, the networking equipment that the participant brings contributes to their costs in the price/performance results.

Some participants could bring “smart” (and usually expensive) network equipment to either offload some of the proxy work from the cache box or to perform load balancing and similar optimizations (e.g., Cisco would probably bring a router that supports WCCP). Other participants would not rely on any costly network features to manage the traffic (e.g., Squid box can use a humble 100Mbit hub). The requirement to include the price of the networking gear into the total cost equalizes both types of participants.

Each Polygraph machine requires a fast ethernet port, so the participant must have enough ports to connect all of Polygraph machines within participant’s cluster.

We ran *netperf* (Appendix J.1) to test raw TCP throughput, and then did Polygraph “no-proxy” runs (Appendix J.2) to ensure that each cluster of clients and servers can generate enough load on its network to sufficiently drive the cache under test. With switches brought by Polyteam, all *netperf* tests showed raw bandwidth of at least 93 Mbps, and no-proxy runs sustained throughput

of at least 300 req/sec. These numbers convinced us that the clusters are configured correctly. When vendors later installed their own networking gear, we repeated the netperf and no-proxy tests. The results of those tests for each participant are reported in the Appendices. In short, all participants were satisfied with the results.

4.7 Filling the Caches

The bake-off rules require that caches be full for the PolyMix #1 tests. Measuring performance for empty or partially-full caches during the bake-off is wrong because those are not steady-state conditions. The rules define a “full cache” as one that does not continue to allocate disk space despite a steady stream of cachable traffic. When a cache is full, object deletion rate is equal to object addition rate, providing for steady-state conditions. Note that due to cache configuration or aggressive garbage collection, disk space utilization of a “full” cache is not necessarily 100%. The duration of (total number of requests for) the filling-the-cache run can be calculated given the cache size and average request rate (reply size).

As mentioned previously, caches must be fillable within eight hours. All participants met this requirement. Performance results from this phase of the benchmark are not reported.

The “Full cache” requirement (along with a limited time for the bake-off) has various implications for restarting failed runs, and dealing with crashes. A participant with a cache that has a slower filling rate (or with a larger cache) would require more time to refill the cache than a participant with a faster (or smaller) cache. Hence, the former participant would have less time to rerun or finish a given number of experiments. This inequality might penalize products that are, performance-wise, comparable with or superior to their competitors.

4.8 Running Polygraph

The bake-off test suite consists of a series of Polygraph runs. All runs have the same workload except for the request rate.

Each run is about one-hour in duration (`--goal 1hr`). There is a 10 minute idle period between runs. This idle time was introduced to allow the cache to settle down and timeout its pending connections, if any. Because participants were able to rerun some of the tests, the gaps between tests could be significantly longer than 10 minutes if a participant was not sure which test to rerun next.

A Perl script running on a monitoring PC started *polyclt* and *polysrv* processes using *ssh*. Each Poly-client is generating request stream with a rate equal to the total required request rate divided by the number of Polygraph client-server pairs.

Initially request rates were “mixed” in a low/high sequence. However, our rules allow any individual run to be restarted. The only requirement is that the cache be full. Participants are allowed to flush, refill the cache, then start a run. Our filling-the-cache workload probably leaves the cache disks/filesystem in a state that may give better performance for the next run, than would be achieved after a series of PolyMix #1 tests. The Rules for the next bake-off eliminate this potential problem (see Section 6).

Chapter 5

Comparison of Results

This section compares performance of all participating proxies. We also compare price/performance ratios. Detailed performance data for each participant can be found in the Appendices.

Good “performance” means different things to different people. Some value throughput, while others look mostly at bandwidth or response time savings. We have tried to show the results from various angles. We expect the reader to weight various performance aspects according to her preferences.

Participating caches vary in price, maximum throughput, cache capacity, and other sizing parameters and design decisions. Special care should be taken when comparing performance results. The reader should consult Appendices for details about proxy configurations and match those configurations with his needs, making price and performance adjustments if necessary.

The two traditional performance metrics are *throughput* and *response time* (Sections 5.1 and 5.2). Caching applications add another orthogonal characteristic — *hit ratio* (Section 5.3). Price/performance analysis compares response time measurements adjusted for the price of the equipment (Section 5.5).

As mentioned previously, we have results from six participants and four vendors. Both InfoLibria and Novell/Dell brought two products to the bake-off. We distinguish their entries by appending “-L” for large, and “-S” for the smaller entries.

5.1 Throughput

The bake-off workload is based on constant request rate. A proxy that cannot handle a given request rate either creates a lot of transaction errors or overloads the Polygraph machines with pending transactions. Both situations result in a failed experiment according to the rules of the bake-off. Consequently, throughput (i.e., reply rate) for successful experiments equals to the offered request rate and is not reported. Instead, we often use throughput for the x -axis on the graphs.

Table 5.3 in the end of this Chapter gives exact peak throughput values for each participant.

5.2 Response time

There are several ways to measure response time. No single measurement shows the entire picture. The most commonly used *mean* response time is easily skewed by a few extreme measurements.

Response time *histograms* depend a lot on the bin sizes. *Percentiles* are affected by clustering of the results.¹We present several measurements rather than making a single imperfect choice.

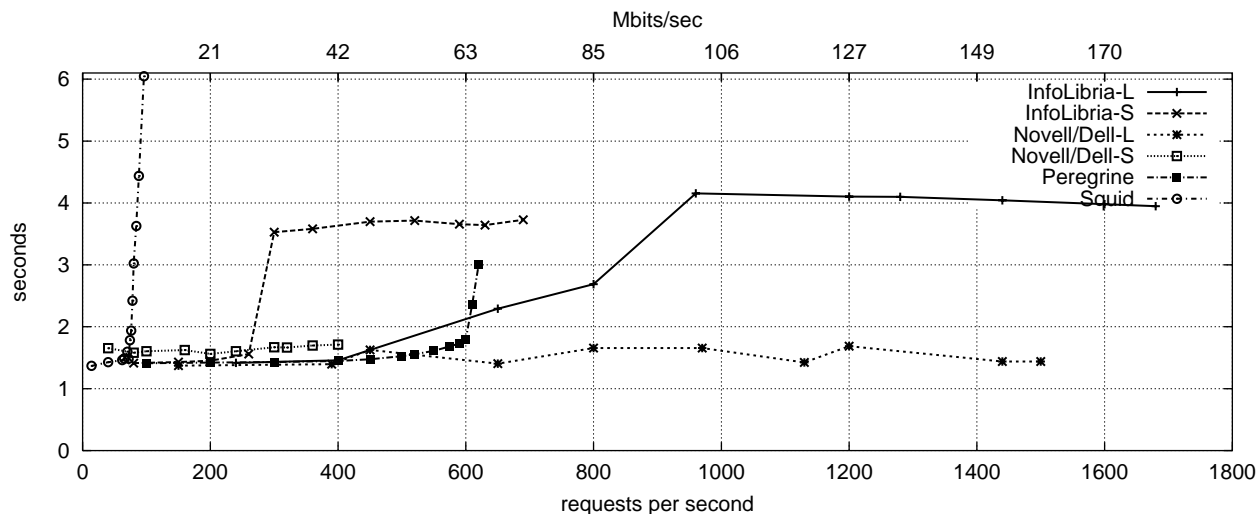


Figure 5.1: Response Time: Mean

Figure 5.1 shows mean response time for all participants. Response time curves turned out much different than expected. Only two proxies, Squid and Peregrine, exhibit a “theoretical” exponential growth of response time when they approach 100% utilization levels (about 100 req/sec and 650 req/sec).

Novell/Dell proxies handle load increase without response time degradation up to their corresponding maximum request rates (400 and 1500 requests per second). Beyond 1500 req/sec, a sharp degradation of proxy performance does not allow for a successful Polygraph run. We speculate Novell/Dell’s bottleneck is CPU.

When the load increases, DynaCache components of InfoLibria boxes accept only some connections and let the rest pass through DynaLink without bothering the cache. InfoLibria calls this patent-pending feature “connection pass-through.” The mechanism prevents response time growth until the DynaLink network utilization becomes too high. An instant change of behavior is especially visible on the InfoLibria-S cache. The presence of this resource saving mode allows InfoLibria to achieve higher request rates than other participants in the same throughput range (690 req/sec and 1680 req/sec), but at a price of significantly higher response time.

Figure 5.2 depicts median (i.e. 50-*th* percentile) response times. Complete response time distributions as well as mean, median, and 75% plots for each participant can be found in the Appendices.

Response time percentiles are similar in shape to the mean response time curves (Figure 5.1). However, the clear differences in absolute values give insight on proxies design.

We define cache *utilization* as the percentage of maximum throughput supported by the box. Hence, utilization ranges from 0% at 0 req/sec to 100% at the corresponding maximum request rate supported by the cache.

At low utilization levels, all proxies are able to send 50% of replies in 500 msec or less. Most of

¹For example, with 55% hit ratio, 50-*th* percentile (i.e. median response time) is usually very small while 60-*th* percentile is already large.

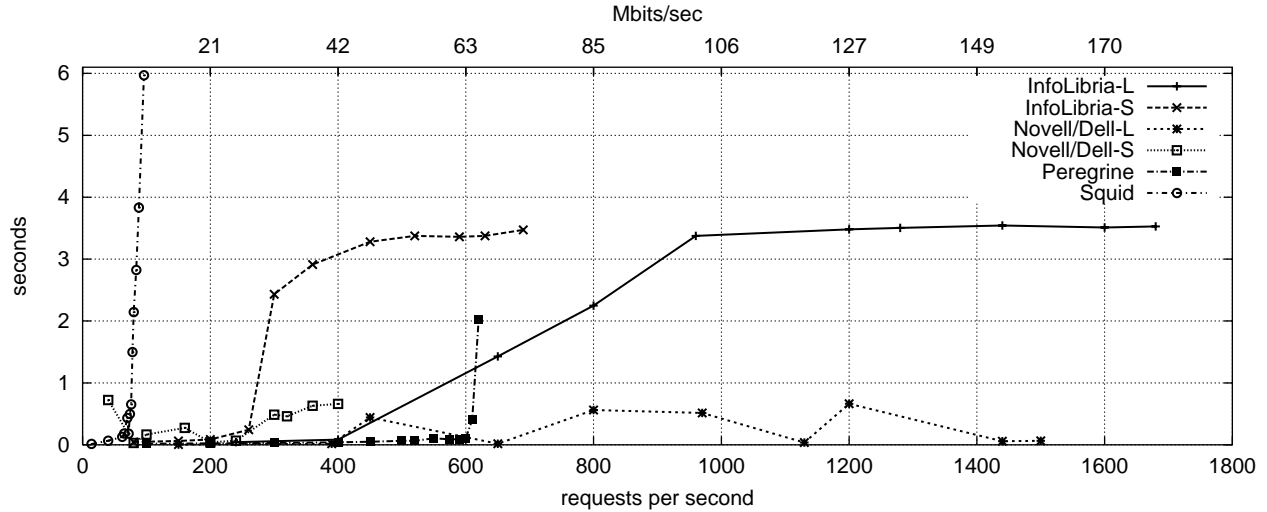


Figure 5.2: Response Time: Median

those fast replies are apparently hits because misses are slowed down by an artificial 3 sec delay on the server side.

When utilization is high, the optimizations depend on the product under test. For example, Novell/Dell boxes still have at least 50% of fast replies and 75% of replies below 3.5 seconds. On the other hand, similar 50% and 75% measurements for both InfoLibria products are about 3.5 and 5 seconds, approaching the no-proxy case. Hit ratio analysis (Section 5.3) further illustrates the difference between Novell/Dell and InfoLibria approaches.

Both Peregrine and Squid are able to reply fast until their utilization is close to 100% when all replies are slowed down a lot.

5.3 Hit Ratio

Maximizing cache hit ratio (HR) is essential for many proxy installations, especially for non-US markets where bandwidth is scarce and very expensive. Recall that during the bake-off, Web Polygraph was configured to offer a “constant hit ratio” of 55%. Thus, a proxy could not achieve higher ratios, and HR maximization problem was reduced to keeping HR at 55% level. This section studies the actual hit ratios achieved by proxies.

Measured HR versus load is depicted on Figure 5.3. All products except Novell/Dell-S are capable of preserving Hit Ratio at 55% at relatively low loads. Novell/Dell-S consistently shows ratios in 46-50% range regardless of the load.

As proxy utilization increases, Squid and InfoLibria proxies start “missing hits.” Squid temporarily stops opening disk files when the number of open disk files is above a certain threshold value (200 open files). Connection pass-through mode employed by InfoLibria is similar to Squid’s algorithm except Squid does not have the luxury of bypassing the cache on the network level. Similar techniques are known to be employed by many other caching vendors and are sometimes called “healing mode.”

Both InfoLibria boxes are capable of supporting the highest request rates (690 req/sec and 1680

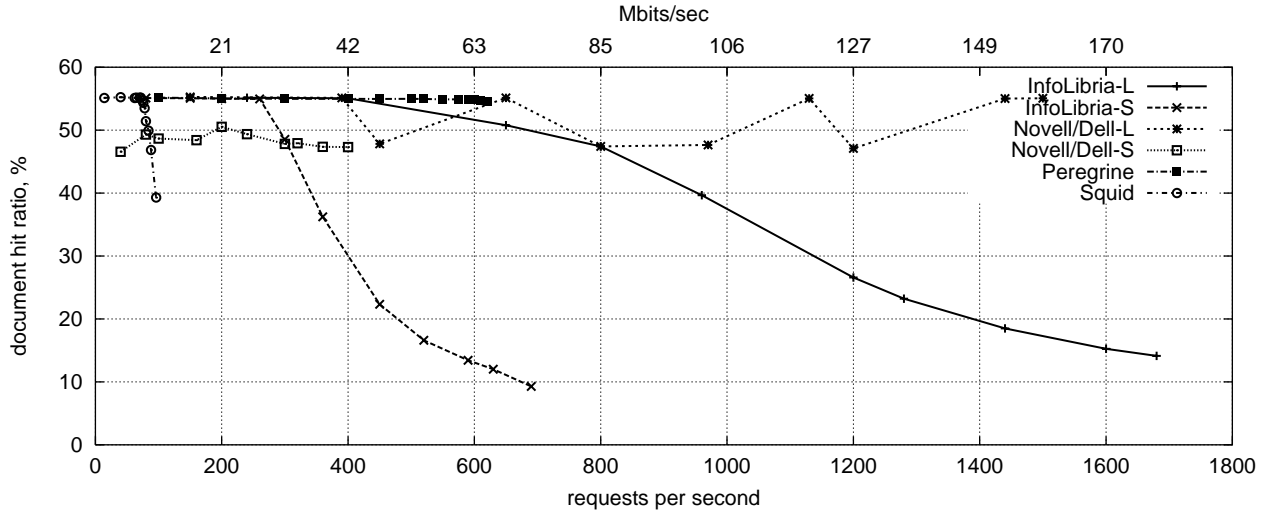


Figure 5.3: Document Hit Ratio

req/sec) at a cost of significantly reduced hit ratios (9% and 14%). No other participant has demonstrated similar trade-off abilities.

The Peregrine proxy does not have a “healing mode” and shows a steady Hit Ratio around 55%.

The Novell/Dell-L hit ratio fluctuates between 47% and 55%. See Appendix F.5 for an explanation from Novell/Dell.

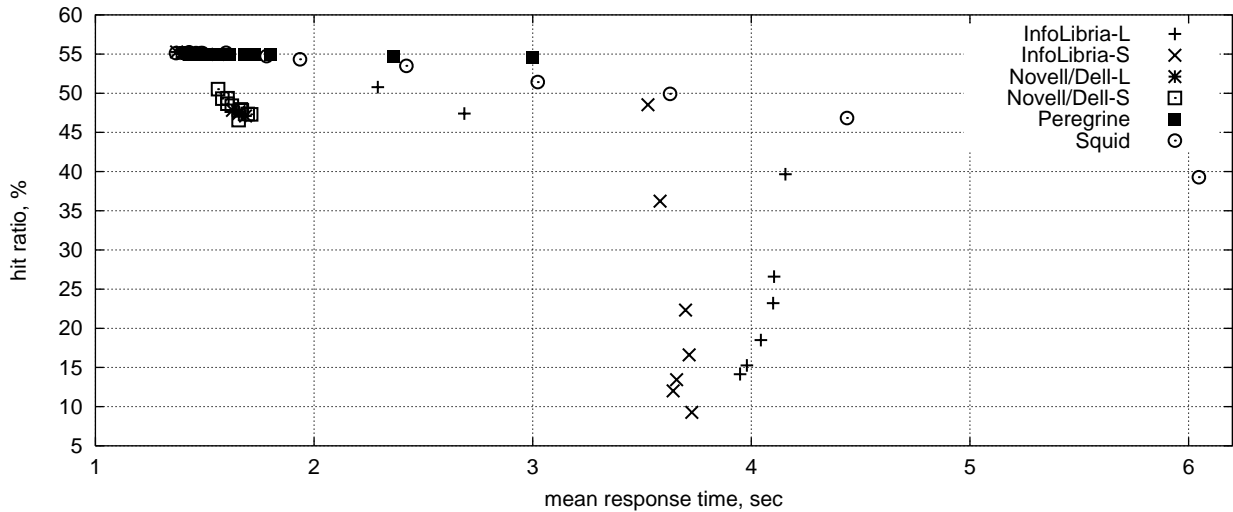


Figure 5.4: Document Hit Ratio versus Mean Response Time

Cache hits are not delayed by the 3 second server side “think time.” Thus, it is reasonable to expect that hits are significantly faster than misses. Consequently, Hit Ratio should affect response time distribution and averages. However, the actual dependence may be quite complex.

Figure 5.4 illustrates the measured relationship between Hit Ratio and mean response time. Note that this Figure is just a combination of Figure 5.1 and Figure 5.3 to emphasize the differences in

cache behavior.

Two clusters with mean response time below two seconds represent fast replies of underutilized proxies. Note that both Novell/Dell proxies have all their data points in these clusters. For the Novell/Dell, caches, utilization does not affect response time or hit ratio much.

Peregrine proxy also does not sacrifice hit ratio to high request rates. Longer response delays are exclusively due to the increased amount of work on the proxy side rather than more misses in the reply stream.

The relationship between hit ratio and response time for InfoLibria caches depends on the request rate. At high request rates, hit ratio is decreasing while response time remains stable. The latter implies that the InfoLibria caches are capable of measuring the response time (or some similar characteristic) run-time. The proxies are keeping response time below a certain level by allowing more and more requests to bypass the cache. Stable, albeit high, response time at highest request rates comes at a price of more misses.

Squid response time degrades smoothly as the hit ratio is decreasing because of healing mode.

5.4 Price

Sheer performance is probably important only for product developers and customers that have money to burn. Most customers also have to consider price in their decision making process. We summarize pricing of participating products in Table 5.1. Detailed product configurations are listed in the Appendices.

Table 5.1 lists two prices per proxy: with and without network gear. In some cases, a customer already has the necessary network gear in place and does not have to pay twice for it, or a cheaper alternative is available. The network gear portion of the total cost varies from less than 1% for InfoLibria-S to 52% for Peregrine.

Box	Price	
	Cache alone	Cache with Network
InfoLibria-L	\$199,980	\$202,880
InfoLibria-S	129,995	130,745
Novell/Dell-L	44,999	49,995
Novell/Dell-S	9,999	13,499
Peregrine	7,296	15,291
Squid	3,800	3,960

Table 5.1: Prices

Note that these costs represent *list prices* of the equipment only. In reality, there are many additional costs of owning and operating a Web cache. These may include: software/technical support, power and cooling requirements, rack/floor space, etc. A more thorough cost analysis might try to determine something like a two-year cost of ownership figure.

5.5 Price/Performance

This section presents performance data adjusted for the price of participants' boxes. We use total prices, including networking gear. The choice was approved by vendors attending the preparation meeting in February. See Section 5.4 and Appendices for pricing details.

Price/performance analysis is probably the most controversial part of this report. Knowledgeable readers are rarely misled by pure performance figures regardless of the presentation. With price/performance comparisons, it is much easier to influence reader's opinion by the choice of presentation format.

The discussion in this section is augmented with examples. Our examples try to illustrate how price/performance data can be used in purchasing process. We must however caution the reader that we cannot account for all the factors that must be considered when buying or comparing two products. The reader must make the necessary real-world adjustments.

How much is one reply per second?

Figure 5.5 helps to answer the following question: "Which single product can support a given request rate for less money?" In other words, the lower the curve, the more throughput one can get for the money (using a single proxy).

Example:

Suppose you expect your Web traffic to be in the 200 – 400 req/sec range and you want to buy a single box that can sustain the given throughput. Looking at Figure 5.5, one can immediately say that Novell/Dell-S and Peregrine are the most cost effective candidates. You will end up paying about 32 - 65 dollars for each request per second your system should support.

For request rates around 1400 req/sec, Novell/Dell-L and InfoLibria-L should be considered with price/performance ratios around 35 and 130 dollars per req/sec.

Of course and again, the throughput cost is only one of the factors to be considered.

Regardless of the performance, a single proxy may not be able to support the required request rate. Figure 5.5 can also be used to decide if it makes sense to buy *several* identical boxes to support required traffic. Here is how to get the price for a "cluster" of identical boxes P that must support a given total request rate R :

1. Using performance graphs for participant P (see Appendices), choose the request rate x that a proxy can support without sacrificing other performance factors that are important to you.²
2. On Figure 5.5, find y — the throughput price that corresponds to the request rate x .
3. $R * y$ is the cluster price.

Example:

²This choice is highly dependent on customer requirements. The latter vary a lot. That explains why we could not make the outlined computation ourselves.

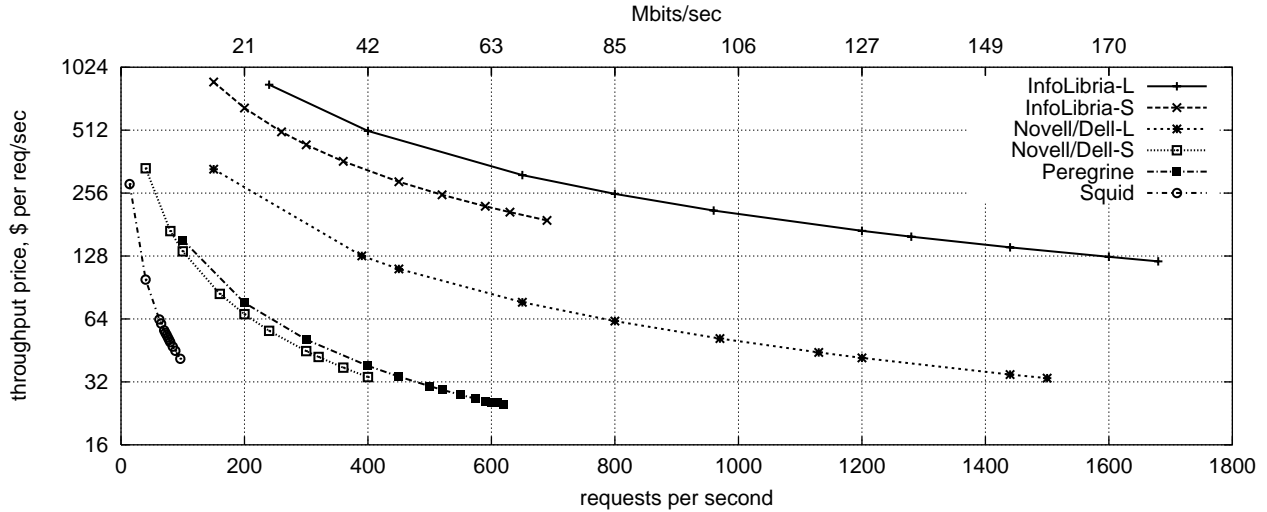


Figure 5.5: Throughput Cost

You want a single-vendor solution that supports throughput around 100-150 req/sec. Clearly, there are several choices. Suppose your final candidates are Squid and Novell/Dell. You can build a cluster using several Squid boxes, or you can go with a single Novell/Dell-S box.

Looking at Appendix I, you decide that (performance-wise) two Squid boxes will do the job just fine (each box will handle 50-75 req/sec with 55% hit ratio). The throughput price for Squid at 50 req/sec is about \$64. Thus, your Squid cluster will cost you at most \$64 dollars per throughput unit to support 100-150 req/sec rates. Comparing that with Novell/Dell-S price/performance at 100-150 req/sec range (\$128 - \$75) you decide that Squid cluster is more cost efficient solution for you, all other factors being equal.

Note that to build a cluster you may need additional network components (e.g., an L4 switch). However, since the price/performance analysis is based on price that includes network equipment, the total price of all network components will often be close to the price you will have to pay for the network gear required for your cluster. Otherwise, adjustments are necessary.

Figure 5.6 shows the same information as Figure 5.5, except the x -axis represents proxy utilization or percent of the maximum supported request rate. Figure 5.6 is not very useful for making buying decisions because the latter are usually based on desired request rate. However, the Figure depicts a certain *efficiency* of a product. Here, “efficiency” means how well a proxy is spending your money at a given utilization level (rather than at a given request rate).

Example:

Interestingly, Novell/Dell products support very different request rates but have the same efficiency. Apparently the latter indicates that software, hardware, and pricing were scaled proportionally to the supported throughput. On the other hand, InfoLibria’s two caches have different price/utilization curves, perhaps indicating an alternative scaling strategy.

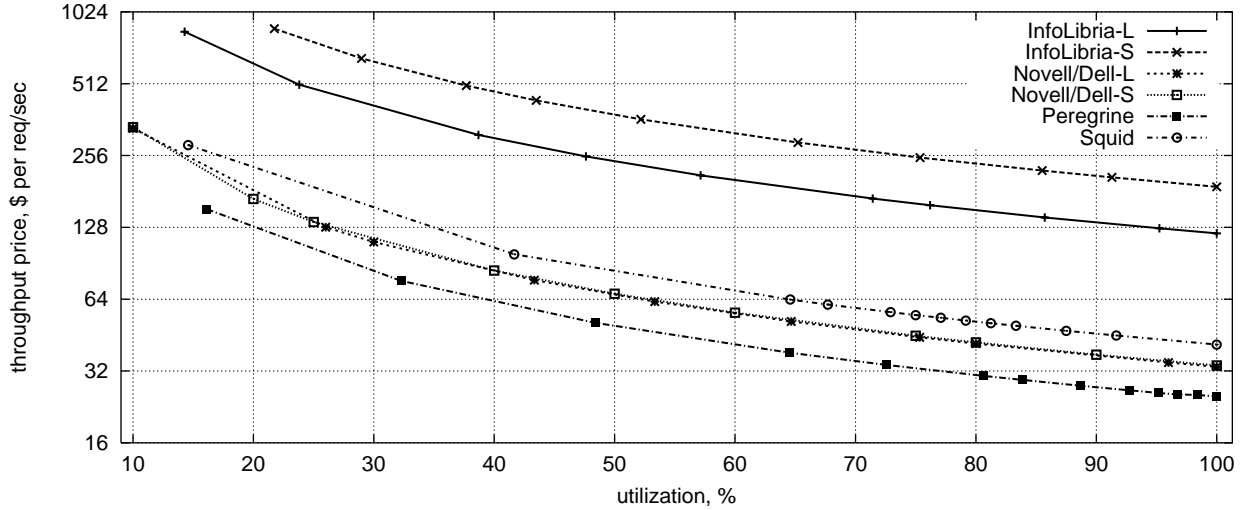


Figure 5.6: Throughput Cost versus Utilization

5.6 Failed Runs

Six participants have chosen to disclose their results. Interestingly enough, four out of those six overestimated the peak performance of their boxes (see Table 5.2). Participant entries in the Appendices have details about peak and other request rates for each participant.

Overestimation can be attributed, in part, to the limited experience the participants had with Polygraph. However, there were other important reasons. The reader must consult with corresponding Participant Comments sections in the Appendices before making any conclusions.

Box	Peak Request Rate	
	Achieved	Claimed
InfoLibria-L	1680	1680
InfoLibria-S	690	720
Novell/Dell-L	1500	1600
Novell/Dell-S	400	400
Peregrine	620	660
Squid	96	140

Table 5.2: Claimed and Achieved Peak Throughput

Bake-off Rules disqualify runs with more than 3% of failed transactions. Three percent was an arbitrary chosen number. In practice, an experiment would either fail completely or have less than 0.1% of transaction errors. Hence, all reported experiments have negligible number of errors unless stated otherwise.

5.7 Summary

Table 5.3 summarizes information presented in this chapter. The table shows peak values and ranges (min-max) of most important measurements discussed in this report as well as price and a pointer to the appendix where more information (including Polyteam and Participant Comments) can be found.

Box	Total Price (\$)	Peak Throughput (req/sec)	Mean Response Time (sec)	Hit Ratio (%)	Appendix
InfoLibria-L	202,880	1680	1.4-4.2	14-55	B
InfoLibria-S	130,745	690	1.4-3.7	9-55	C
Novell/Dell-L	49,995	1500	1.4-1.7	47-55	F
Novell/Dell-S	13,499	400	1.6-1.7	47-51	G
Peregrine	15,291	620	1.4-3.0	55-55	H
Squid	3,960	96	1.4-6.0	39-55	I

Table 5.3: Price and Performance Summary

Once again, we note that summaries are often misleading and should be used or interpreted only after reading the detailed analysis. See section 1.4 for hints on how to read this report.

Chapter 6

Future Work

IRCache intends to hold another benchmarking event around September–October, 1999. In August, Polyteam intends to publish a revised addition of The Rules and release the next generation of Web Polygraph software. Below is a list of major modifications and additions planned by Polyteam. All items are, of course, subject to change.

The Rules: The following major changes are proposed to allow for more and better tests:

- Increase the duration from 3 to 5 days
- Becoming a participant should be easier, bailing out should be harder
- Test-sets (see below)
- Mandatory proxy interface to report disk space utilization
- More experiment variety
- Rigid requirements on product availability

Polygraph: Polygraph additions will include:

- Support for persistent connections and, perhaps, pipelining
- Support for “pages” with inlined images and, perhaps, hyper-references
- Better object popularity and temporal locality models
- A client talking to many servers
- More, and higher quality statistics

Test-sets: Each participant must pass several *sets* of tests. A set includes a filling-the-cache experiment and several performance tests. Participants are allowed to complete the tests within one set in any order but cannot interleave experiments from different sets. A set has a limit on the total number of attempts a participant can make to complete all the tests (e.g, at most 10 attempts to complete 6 tests). A participant chooses which attempts to report, but the results for all tests must be reported.

Chapter 7

Conclusions

The Web caching industry's thirst for a benchmarking standard led to the creation of Web Polygraph and the launch of a series of IRCache bake-offs. The first bake-off was a success. Despite the absence of several big players, we have collected a representative set of interesting performance data. This report is the first industry document that provides a fair performance comparison of a variety of caching proxies. We hope the performance numbers and our analysis will be used by buyers and developers of caching products.

Polyteam applauds the vendors who came to the bake-off and disclosed their results. We regret that other caching vendors did not show their leadership. We certainly hope that more companies will participate in future benchmarking events and will have the courage to disclose their results.

We envision a lot of *a posteriori* discussions about the bake-off and its results, including, perhaps, attempts to denounce bake-offs in general. In fact, some anti-bake-off material appeared even before the event was finished. Our answer is simple: We believe that first bake-off rules and workload give knowledgeable customers a lot of valid, useful, and unique performance data. Future bake-offs will further improve the quality and variety of our tests. Polyteam does not know a better substitute for a fair same-rules, same-time competition.

Finally, we expect some companies will try to mimic bake-off experiments in private labs, and we certainly welcome such activities. We trust the reader will be able to separate unsubstantiated speculations and semi-correct bake-off clones from true performance analysis. If unsure about the validity of vendor tests, consult this report and Polyteam members directly.

Bibliography

- [1] Polyteam, “Web polygraph site.” <http://polygraph.ircache.net/>.
- [2] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and zipf-like distributions: Evidence and implications,” in *Proceedings of Infocom'99*, March 1999.
- [3] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, “Characterizing Reference Locality in the WWW,” in *Proceedings of PDIS'96: The IEEE Conference on Parallel and Distributed Information Systems*, (Miami Beach, Florida), December 1996.

Appendix A

IBM

A.1 Equipment

Description	Price
IBM 34C00 Web Cache Manager This unit includes: (8) 9-GB IBM SSA disks (2) ?-MHz RISC processors (2) 100 MB/s ethernet interfaces 1-GB RAM	—
(1) IBM 8271 12-port 10/100 ethernet switch	—

A.2 Results

IBM declines to publicly release any of their results. All data related to the tests of IBM's Web Cache Manager has been removed from the official bake-off results based on IBM's request. The Rules of the bake-off prohibit the inclusion of any other comments regarding IBM's decision.

Appendix B

InfoLibria-L

<http://www.infolibria.com/products/f-dyna.htm>

B.1 Equipment

Description	Price
Cluster of (4) DynaCache IL-100-7. Each DynaCache unit includes: (7) Seagate 4-GB 10,000-RPM SCSI Disks (1) DynaLink 512-MB RAM	\$199,980
(4) Bay Networks Netgear FS104 4-port 10/100 ethernet switches	\$1,000
(1) Cisco Catalyst 2900 16-port 10/100 ethernet switch	\$1,900
Total	\$202,880

B.2 Configuration

Cache disks: Total cache size is 30-GB, spread across 28 disks.

Software: DynaCache 2.0.

Polygraph machines: This cluster uses eight (8) client-server pairs.

Network: The Polygraph clients are all connected to the 4-port Netgear switches; two clients are connected to each switch. Each of the four DynaLink's is also connected to each of the four switches. Each DynaLink is connected to a DynaCache with a pair of 100BaseT cables. Each DynaLink is also connected to the 16-port Cisco switch, along with all of the Polygraph servers. Figure B.1 shows the networking diagram for this cluster.

All IP addresses are in the 10.12.11/24 network. Polygraph clients send requests directly to the cache.

Request Rates: PolyMix #1 tests were executed for the following request rates: 240, 400, 650, 800, 960, 1200, 1280, 1440, 1600, 1680.

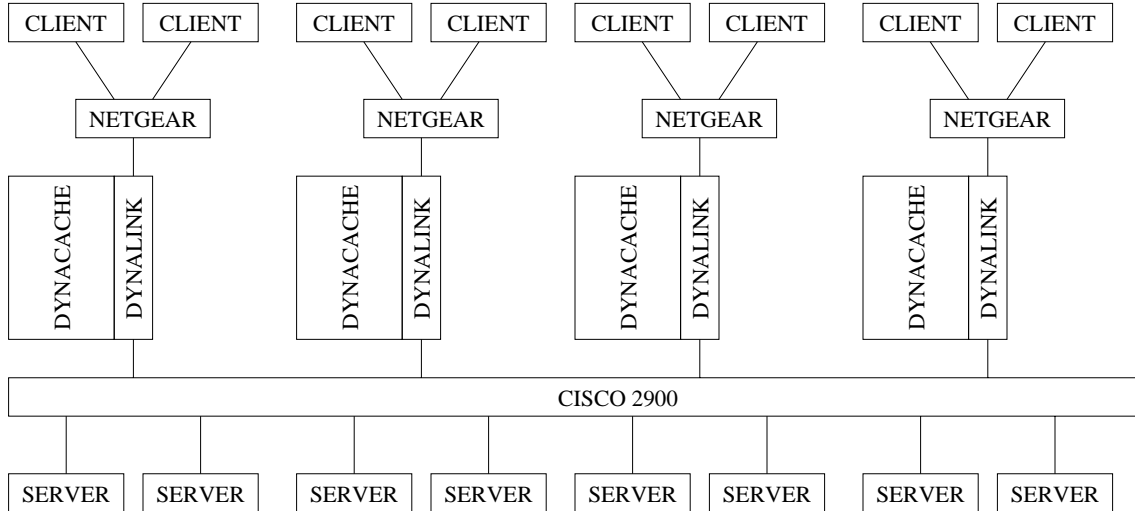


Figure B.1: InfoLibria-L network diagram

B.3 Results

Network throughput

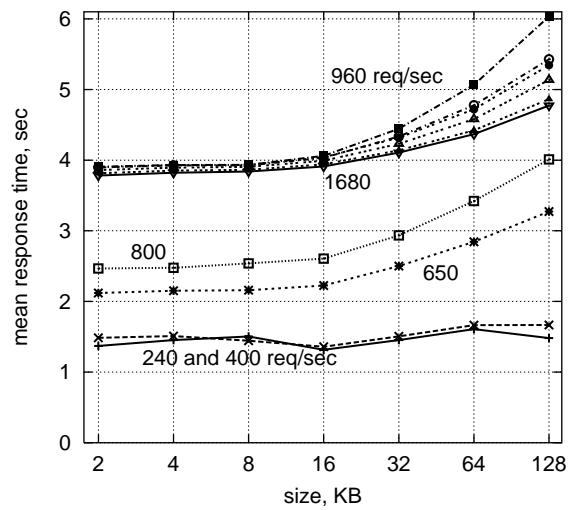
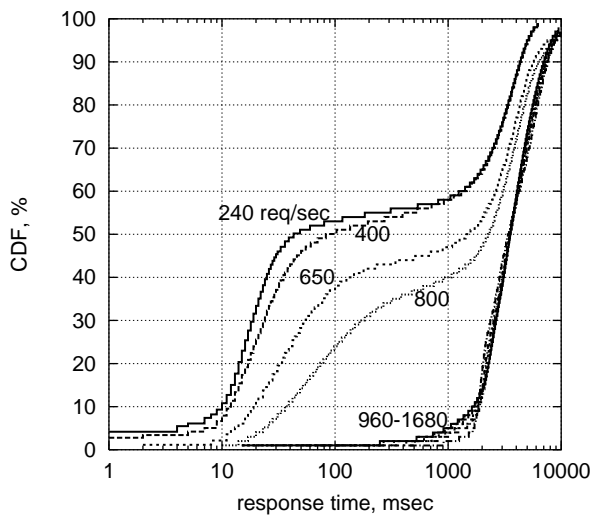
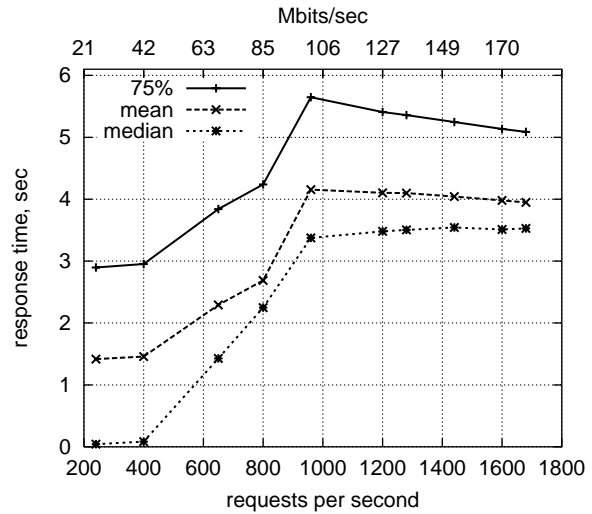
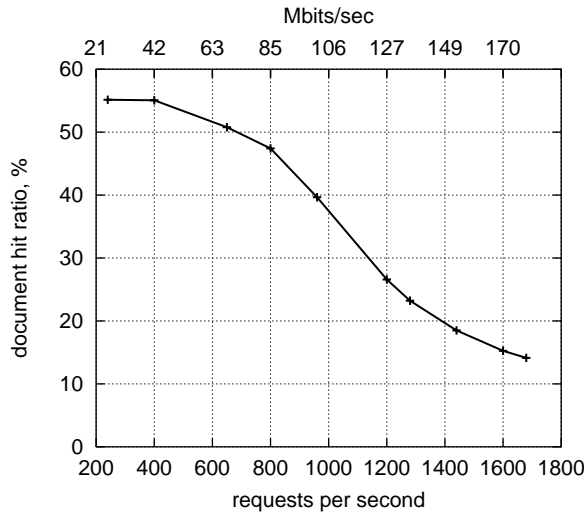
Our five-minute *netperf* test on this cluster showed the mean throughput between each client-server pair to be 46.5 Mbits/sec.

No-Proxy

Our 1-hour no-proxy test on this cluster showed that the network can handle at least 300 requests per second.

PolyMix #1

Request Rate (per second)	Mean Response Time (seconds)	Hit Ratio (percent)
240	1.42	55.1
400	1.46	55.1
650	2.29	50.8
800	2.69	47.4
960	4.16	39.7
1200	4.10	26.6
1280	4.10	23.2
1440	4.04	18.5
1600	3.98	15.3
1680	3.95	14.1



B.4 Polyteam Comments

During the test, one DynaCache unit (main processor) failed and was replaced. Of course, for a real customer, this replacement would be covered at no cost under InfoLibria's warranty.

B.5 Participant Comments

First, InfoLibria wishes to thank IRCache's PolyTeam for their admirable performance of a valuable service to the caching community of users and vendors. The bake-off set a high standard both for design and execution, and for the cache robustness required for completion. We hope that other cache vendors will join in future bake-offs.

Second, we remind the reader that the bake-off test focuses exclusively on caching performance and price, under certain standardized conditions. A caching product must also exhibit a number of other equally key attributes for it to be suitable for the intense demands of an Internet environment. An

Internet-grade cache must be fail-safe, highly manageable, able to maintain the integrity of content, and not impede e-commerce transactions or the collection of usage data. In addition, the price of a cache is only one component in the *total cost of ownership*. The cost of ownership also includes the cost of any human labor needed to maintain the cache in full operation. DynaCaches uniquely excel in all of these regards in ways that are not captured in the bake-off.

Appendix C

InfoLibria-S

<http://www.infolibria.com/products/f-dyna.htm>

C.1 Equipment

Description	Price
(1) DynaCache IL-200X-14. The DynaCache unit includes: (14) Seagate 4-GB 10,000-RPM SCSI Disks (1) DynaLink 512-MB RAM	\$129,995
(3) Bay Networks Netgear FS104 4-port 10/100 ethernet switches	\$750
Total	\$130,745

C.2 Configuration

Cache disks: Total cache size is 15-GB, spread across 14 disks.

Software: DynaCache 2.0.

Polygraph machines: This cluster uses three (3) client-server pairs.

Network: The three clients are connected to one 4-port switch. The fourth port on that switch is connected to the DynaLink. Similarly, the three servers and the other DynaLink port are connected to the 4-port switch. Figure C.1 shows the networking diagram for this cluster.

The third 4-port Netgear was needed only for our monitoring station and central router link.

All IP addresses are in the 10.13.11/24 network. Polygraph clients send requests directly to the cache.

Request Rates: PolyMix #1 tests were executed for the following request rates: 80, 150, 200, 260, 300, 360, 450, 520, 590, 630, 690, 720.

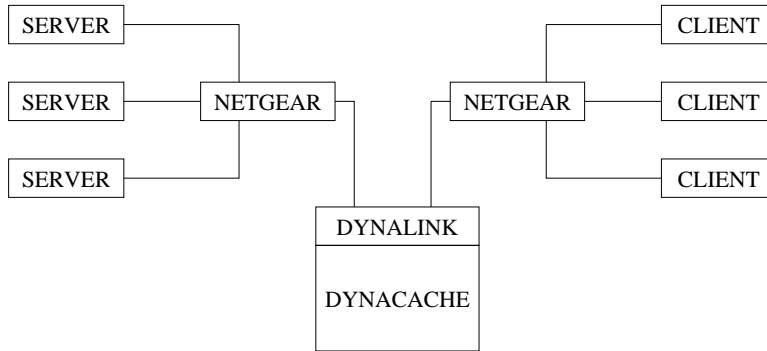


Figure C.1: InfoLibria-S network diagram

C.3 Results

Network throughput

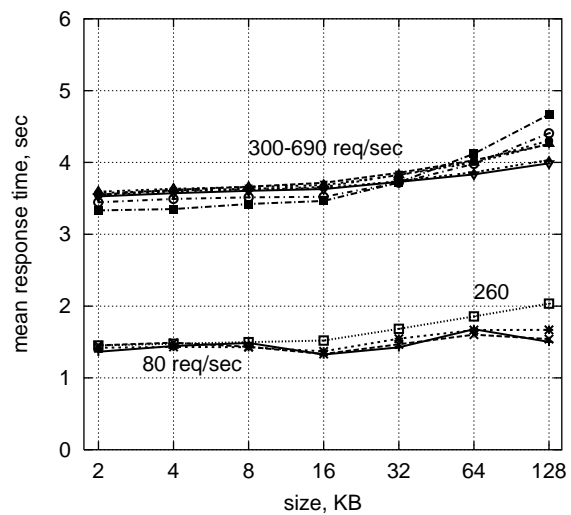
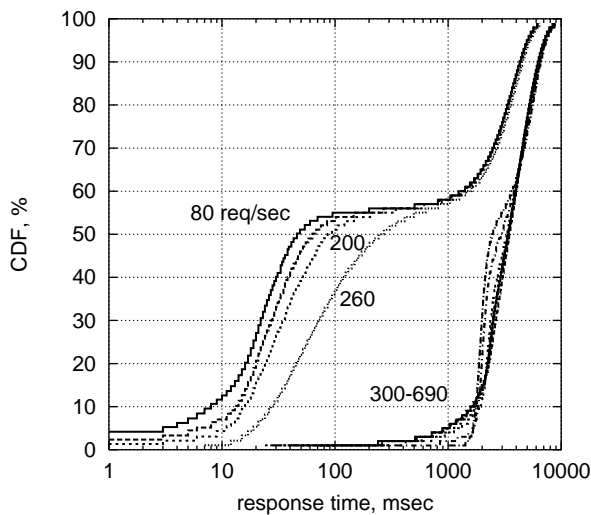
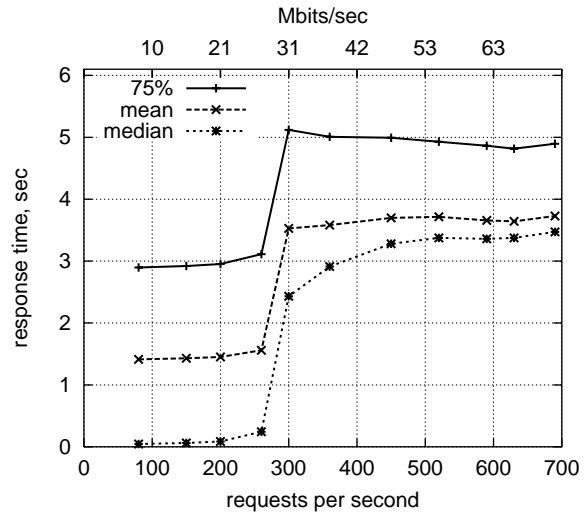
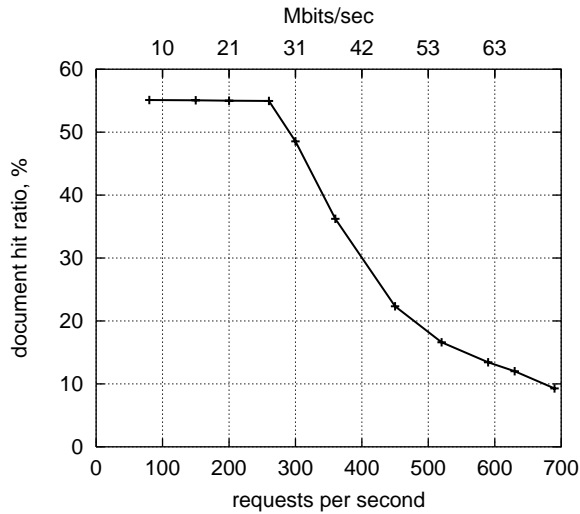
Our five-minute *netperf* test on this cluster showed the mean throughput between each client-server pair to be 31.0 Mbits/sec.

No-Proxy

Our 1-hour no-proxy test on this cluster showed that the network can handle at least 250 requests per second.

PolyMix #1

Request Rate (per second)	Mean Response Time (seconds)	Hit Ratio (percent)
80	1.41	55.1
150	1.43	55.1
200	1.45	55.0
260	1.56	55.0
300	3.53	48.5
360	3.58	36.2
450	3.70	22.3
520	3.72	16.6
590	3.66	13.4
630	3.64	12.0
690	3.73	9.3



C.4 Polyteam Comments

Failed runs This cache failed to handle 720 req/sec. Initially InfoLibria selected 630 req/sec as their maximum. Given some extra time on the third day, they chose to increase their maximum attempted request rate.

C.5 Participant Comments

First, InfoLibria wishes to thank IRCache's PolyTeam for their admirable performance of a valuable service to the caching community of users and vendors. The bake-off set a high standard both for design and execution, and for the cache robustness required for completion. We hope that other cache vendors will join in future bake-offs.

Second, we remind the reader that the bake-off test focuses exclusively on caching performance and price, under certain standardized conditions. A caching product must also exhibit a number of other

equally key attributes for it to be suitable for the intense demands of an Internet environment. An Internet-grade cache must be fail-safe, highly manageable, able to maintain the integrity of content, and not impede e-commerce transactions or the collection of usage data. In addition, the price of a cache is only one component in the *total cost of ownership*. The cost of ownership also includes the cost of any human labor needed to maintain the cache in full operation. DynaCaches uniquely excel in all of these regards in ways that are not captured in the bake-off.

Appendix D

Network Appliance-L

D.1 Equipment

Description	Price
Cluster of (3) NetCache C720S Appliances. Each NetCache unit includes: (4) 9-GB 10,000-RPM SCSI Disks 512-MB RAM	—
(1) Foundry Networks ServerIron 16-port 10/100 ethernet switch	—
Total	—

D.2 Results

Network Appliance declines to publicly release any of their results. All data related to the tests of Network Appliance's Cluster of three NetCache C720S Appliances has been removed from the official bake-off results based on Network Appliance's request. The Rules of the bake-off prohibit the inclusion of any other comments regarding Network Appliance's decision.

Appendix E

Network Appliance-S

E.1 Equipment

Description	Price
(1) NetCache C720S Appliance The NetCache unit includes: (4) 9-GB 10,000-RPM SCSI Disks 512-MB RAM	—
(1) Bay Networks BayStack 350T 16-port ethernet switch	—
Total	—

E.2 Results

Network Appliance declines to publicly release any of their results. All data related to the tests of Network Appliance's NetCache C720S Appliance has been removed from the official bake-off results based on Network Appliance's request. The Rules of the bake-off prohibit the inclusion of any other comments regarding Network Appliance's decision.

Appendix F

Novell/Dell-L

<http://www.novell.com/products/nics/>

F.1 Equipment

Description	Price
Novell Internet Caching System (Beta) powered by Dell This system includes: (1) Dell PowerEdge 6350 (4U rack-optimized) (1) 450-MHz Xeon CPU 2-GB RAM (17) Seagate Cheetah 9-GB Ultra-2/LVD 10,000-RPM SCSI Disks (3) on-board Adaptec AIC-7890 SCSI controllers (1) Adaptec 2940U2W PCI SCSI controller (1) Intel Pro/1000 Gigabit Server Adapter (2) PowerVault 200 rack-mount subsystems	\$44,999
Foundry Networks FastIron Workgroup (FWS24) with Gigabit uplink	\$5,000
Total	\$49,995

F.2 Configuration

Cache disks: Total cache size is 64-GB, spread across 16 disks. The remaining disk is used for system space.

Two of the on-board SCSI controllers are not used. One of the on-board controllers is used for half of the disks. The Adaptec 2940 is used for the other half.

Software: Novell Internet Caching System (Beta)

Polygraph machines: This cluster uses ten (10) client-server pairs.

Network: All devices are connected to the Foundry switch. Figure F.1 shows the networking diagram for this cluster.

All IP addresses are in the 10.17.11/24 network. Polygraph clients send requests directly to the cache.

Request Rates: PolyMix #1 tests were executed for the following request rates: 150, 390, 450, 650, 800, 970, 1130, 1200, 1440, 1500, 1600.

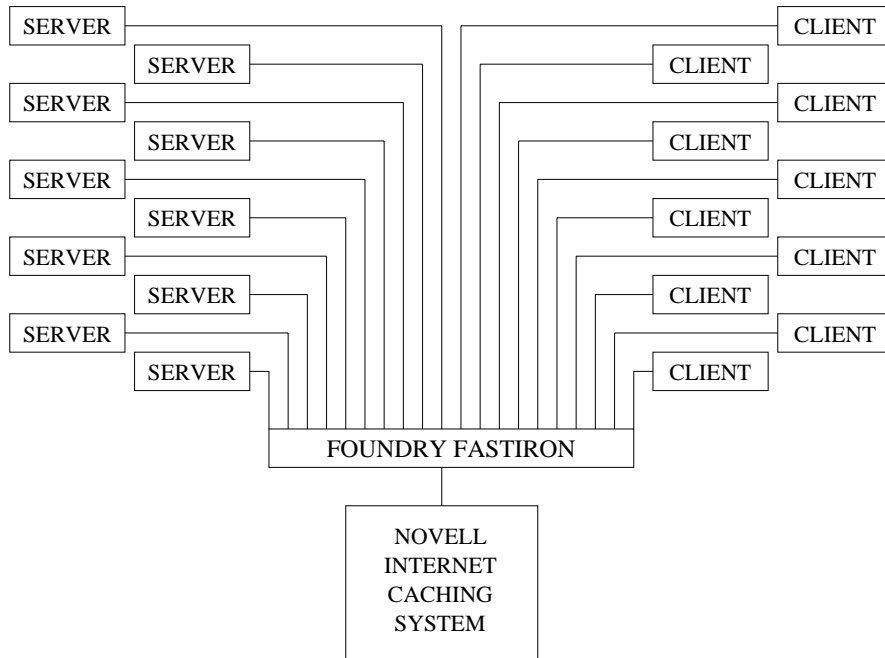


Figure F.1: Novell/Dell-L network diagram

F.3 Results

Network throughput

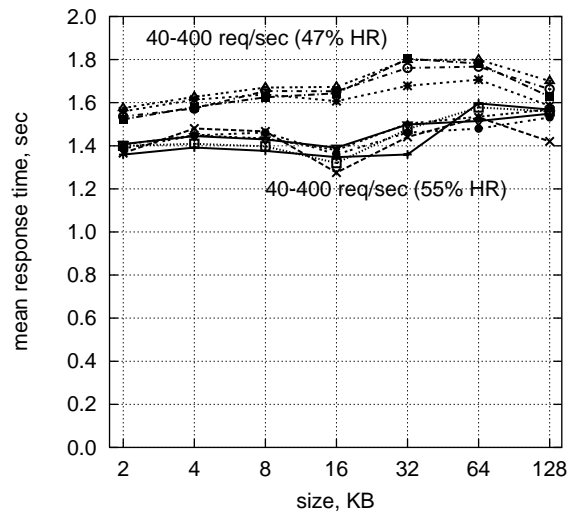
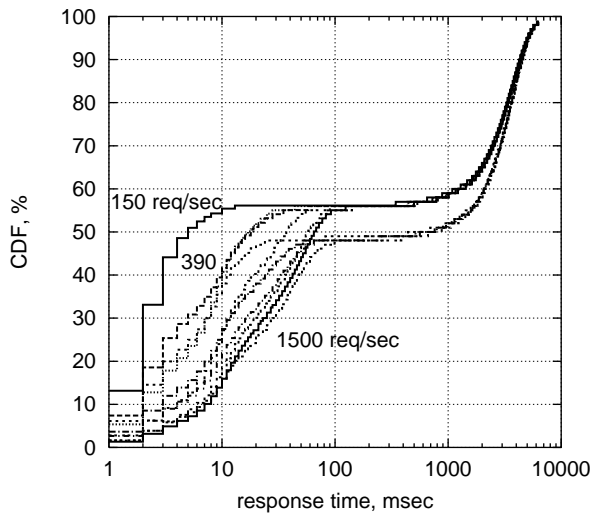
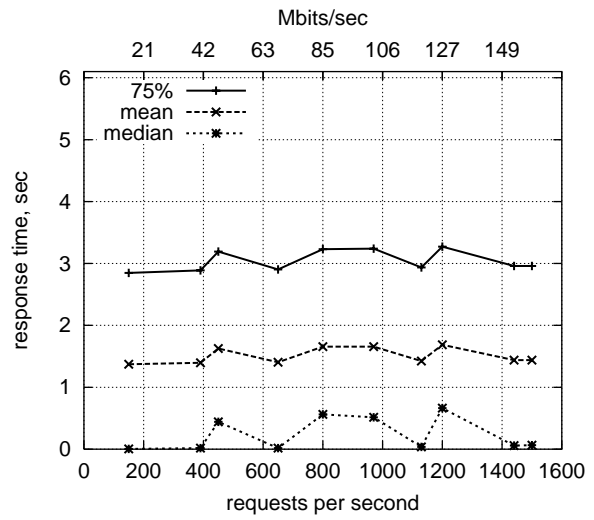
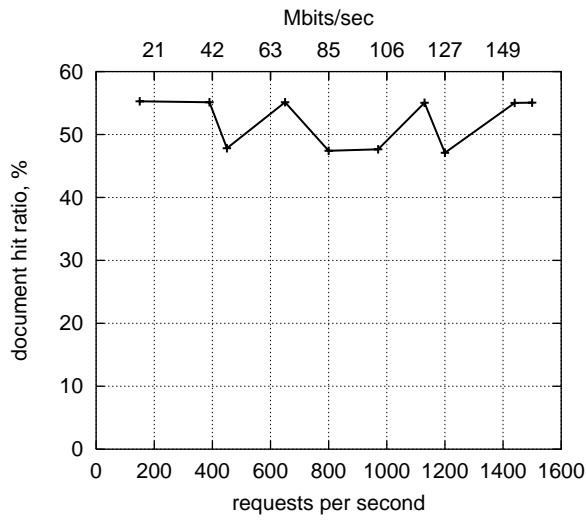
Our five-minute *netperf* test on this cluster showed the mean throughput between each client-server pair to be 93.5 Mbits/sec.

No-Proxy

Our 1-hour no-proxy test on this cluster showed that the network can handle at least 300 requests per second.

PolyMix #1

Request Rate (per second)	Mean Response Time (seconds)	Hit Ratio (percent)
150	1.37	55.3
390	1.40	55.1
450	1.63	47.8
650	1.40	55.1
800	1.66	47.4
970	1.66	47.6
1130	1.42	55.0
1200	1.69	47.1
1440	1.44	55.0
1500	1.44	55.1



F.4 Polyteam Comments

Failed runs This cache failed to handle 1600 req/sec.

A word of caution Participant Comments contain forward-looking statements concerning, among other things, future performance results. All such forward-looking statements are, by necessity, only estimates of future results and actual results achieved by the Participant may differ substantially from these statements.

F.5 Participant Comments

The Novell Internet Caching System (NICS) used in these tests will soon be available from Dell and other Novell OEM's. This new product is a headless appliance that blends high-performance and scalability with web-based remote management and a rack-mount form factor. These results were produced with Dell's beta version of Novell Internet Caching System Powered By Dell.

This solution sustained the highest single-system bake-off result—1500 requests per second—demonstrating Novell's and Dell's leadership in web cache performance and scalability. This solution will be capable of even higher results prior to its final release. The Novell Internet Caching System Powered By Dell also supports clustering which can scale its single system performance produced in the bake-off linearly without the significant hit rate degradation produced by other systems.

The Novell Internet Caching System Powered By Dell successfully maintained a hit rate above 45% for all Polygraph tests. Although this is very good, this product was capable of maintaining the 55% hit rate offered by the test. However, during the bake-off, we encountered a bug in our replacement algorithm that manifested itself after a specific amount of content had been replaced during the tests. This bug broke the replacement algorithm and forced our write process to run through a fairly heavy error path in the code. The effects of this increased workload included a decreased hit rate and higher CPU utilization, which in turn was the cause of our failed attempts to hit 1600. This bug was fixed during the bake-off but the fix was unused due to the bake-off rules. If you were to test our new code, you would experience a consistent 55% cache hit rate across all 10 hours of testing with mean response times remaining below 1.450 seconds all the way up to 1600 requests per second. These performance points, combined with the performance of the Novell/Dell-S system provide a welcome definition of scalability that our customers can afford.

Appendix G

Novell/Dell-S

<http://www.novell.com/products/nics/>

G.1 Equipment

Description	Price
Novell Internet Caching System (Beta) powered by Dell This system includes: (1) Dell PowerEdge 2300 with (1) 450-MHz Pentium II CPU 512-MB RAM (2) Seagate Cheetah 9-GB Ultra-2/LVD 10,000-RPM SCSI Disks (1) Intel Etherexpress PRO/100+ NIC	\$9,999
(1) Foundry Networks FastIron Workgroup (FWS24) 24-port 10/100 ethernet switch	\$3,500
Total	\$13,499

G.2 Configuration

Cache disks: Total cache size is 17-GB, spread across two disks. 0.5-GB of each of the two disks is used for system space.

Software: Novell Internet Caching System (Beta)

Polygraph machines: This cluster uses two (2) client-server pairs.

Network: All devices are connected to the Foundry switch. Figure G.1 shows the networking diagram for this cluster.

All IP addresses are in the 10.16.11/24 network. Polygraph clients send requests directly to the cache.

Request Rates: PolyMix #1 tests were executed for the following request rates: 40, 80, 100, 160, 200, 240, 300, 320, 360, 400.

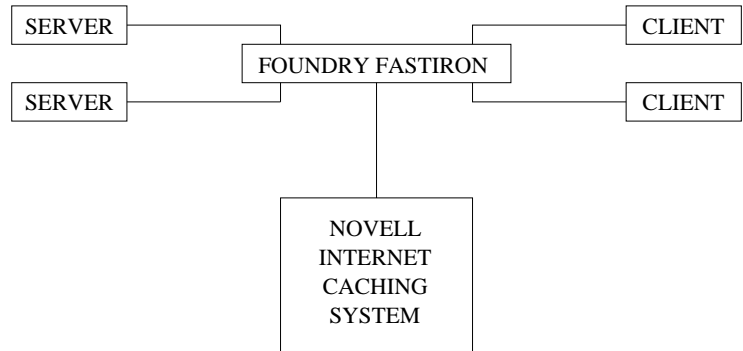


Figure G.1: Novell/Dell-S network diagram

G.3 Results

Network throughput

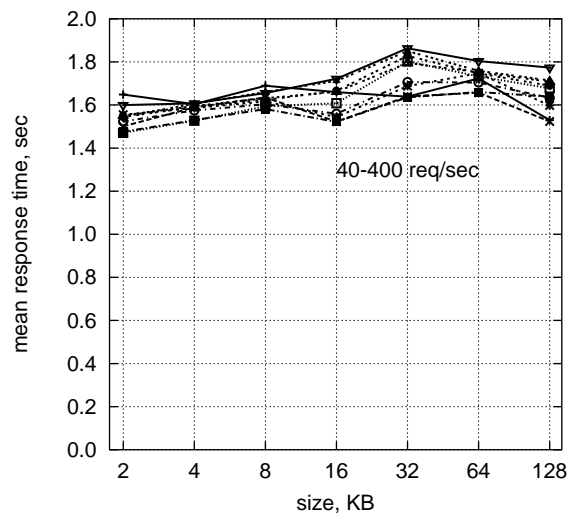
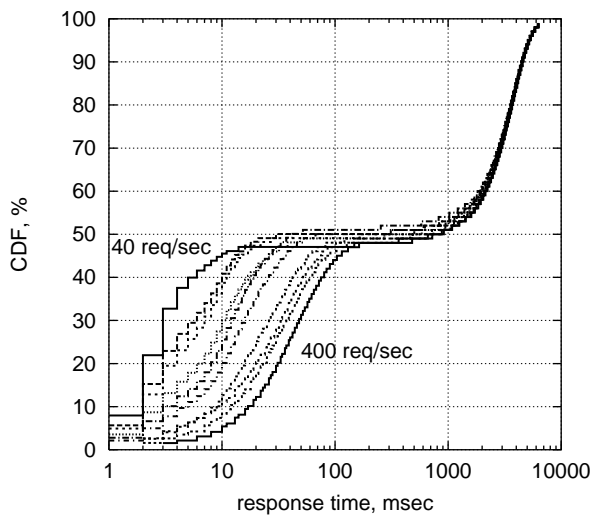
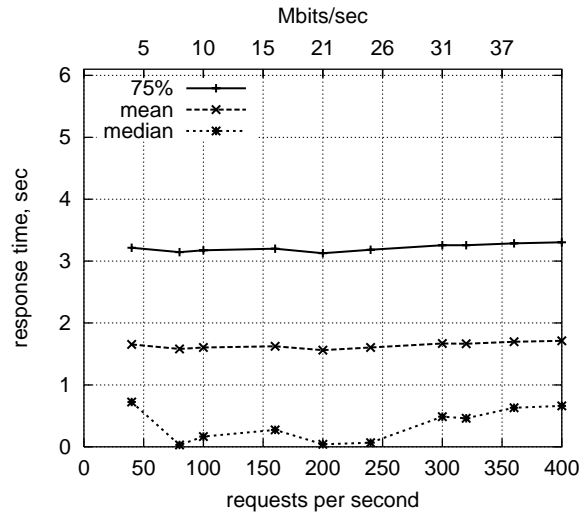
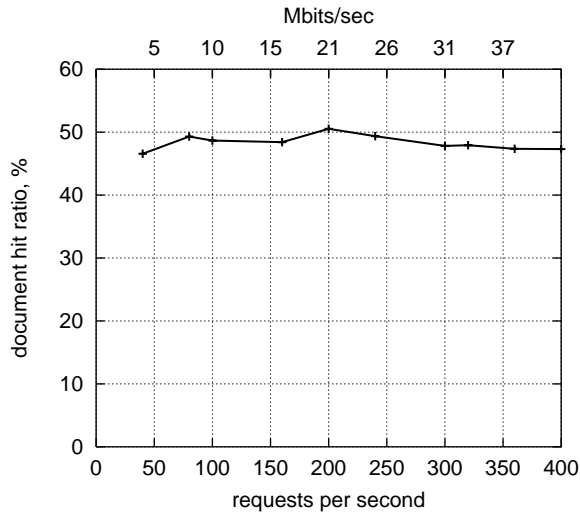
Our five-minute *netperf* test on this cluster showed the mean throughput between each client-server pair to be 93.6 Mbits/sec.

No-Proxy

Our 1-hour no-proxy test on this cluster showed that the network can handle at least 300 requests per second.

PolyMix #1

Request Rate (per second)	Mean Response Time (seconds)	Hit Ratio (percent)
40	1.66	46.6
80	1.58	49.3
100	1.60	48.7
160	1.62	48.4
200	1.56	50.5
240	1.60	49.4
300	1.67	47.8
320	1.67	47.9
360	1.70	47.4
400	1.71	47.3



G.4 Polyteam Comments

A word of caution Participant Comments contain forward-looking statements concerning, among other things, future performance results. All such forward-looking statements are, by necessity, only estimates of future results and actual results achieved by the Participant may differ substantially from these statements.

G.5 Participant Comments

The Novell Internet Caching System Powered By Dell used in these tests will soon be available from Dell. This new product is a headless appliance that blends high-performance and scalability with web-based remote management and a rack-mount form factor. The results in this report were produced with a beta version of that product.

This solution sustained 400 requests per second with a half-duplex network connection. The bot-

tleneck, at a peak throughput of 8.3MBps, was the LAN channel. When this system is configured with a full-duplex connection it sustains 600 PolyMix#1 requests per second, and moves the system bottleneck to the 2-spindle disk subsystem. With the addition of two drives to this configuration, combined with the full-duplex connection, this system can sustain 800 PolyMix#1 requests per second, a 55% cache hit rate, and a mean response time of 1.937 seconds, all while 91% of the hits come off disk. These performance points, combined with the performance of the Novell/Dell-L system provide a welcome definition of scalability that our customers can afford.

The Novell Internet Caching System Powered By Dell also supports clustering which allows you to scale its single system performance linearly without the significant hit rate degradation produced by competing systems.

At the bake-off, the Novell Internet Caching System Powered By Dell successfully maintained a hit rate above 46% for all Polygraph tests. Although this is very good, this product was capable of maintaining the 55% hit rate offered by the test. However, during the bake-off, we encountered a bug in our replacement algorithm that manifested itself after a specific amount of content had been replaced during the tests. This bug broke the replacement algorithm and forced our write process to run through a fairly heavy error path in the code. The effects of this increased workload included a decreased hit rate. This bug was fixed during the bake-off, but the fix was unused due to the bake-off rules. If you were to test our new code, you would experience a consistent 55% cache hit rate across all 10 hours of testing.

Appendix H

Peregrine

<http://www.cs.wisc.edu/~cao/>

H.1 Equipment

Description	Price
Peregrine v1.0	\$0 ¹
Running on a Dell PowerEdge 2300, including: (1) 450-MHz Pentium III CPU 512-MB RAM (1) Intel Etherexpress PRO/100+ (6) Quantum QM39100TD-SCA 9-GB 7200 RPM SCSI disks (1) Adaptec 2940UW SCSI adapter	\$7,296
Alteon ACEdirector 2 8-port 10/100 ethernet switch	\$7,995
Total	\$15,291

H.2 Configuration

Cache disks: Total cache size is 45-GB, spread across five disks. The remaining disk is used for system space and log files.

Software: Peregrine v1.0.

Polygraph machines: This cluster uses three (3) client-server pairs.

Network: All devices are connected to the Alteon switch. Figure H.1 shows the networking diagram for this cluster.

All IP addresses are in the 10.19.11/24 network. Polygraph clients send requests directly to the cache.

¹At this time, the developers of Peregrine are pushing for permission to distribute their software at no cost. There is a chance, however, that Peregrine will not be free. We are willing to assume free software in this cost analysis, especially since they could have lowered their total entry price by using a less expensive ethernet switch.

Request Rates: PolyMix #1 tests were executed for the following request rates: 100, 200, 300, 400, 450, 500, 520, 550, 575, 590, 600, 610, 620, 630, 640, 650, 660.

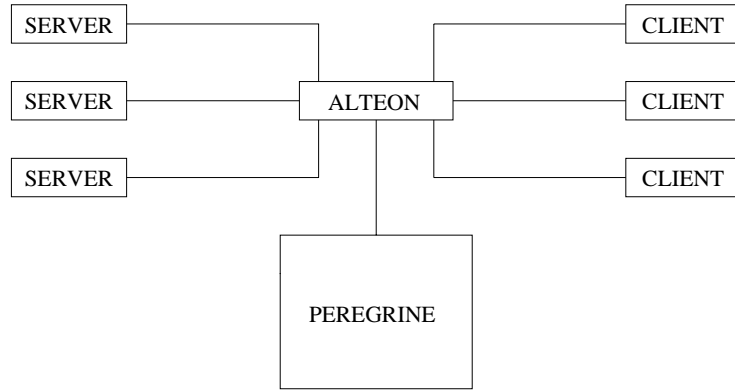


Figure H.1: Peregrine network diagram

H.3 Results

Network throughput

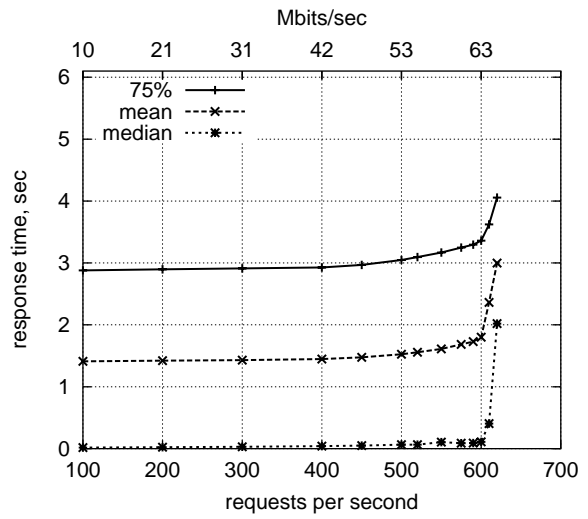
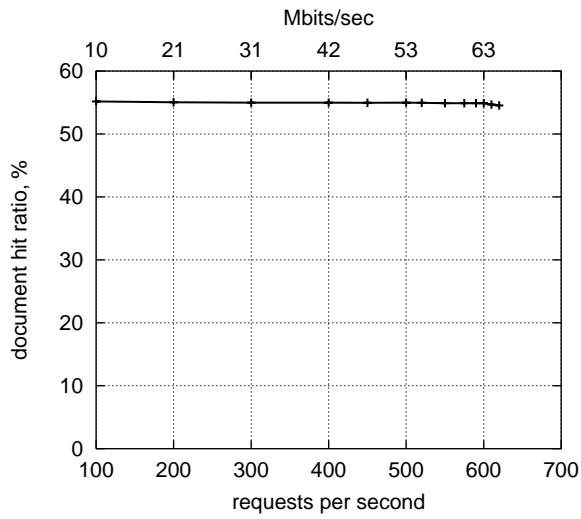
Our five-minute *netperf* test on this cluster showed the mean throughput between each client-server pair to be 93.5 Mbits/sec.

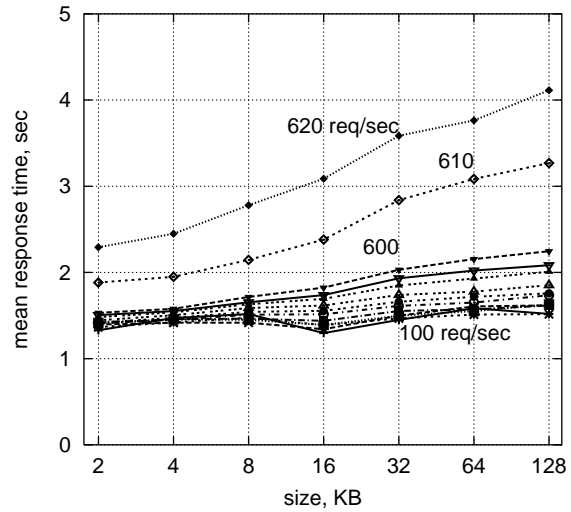
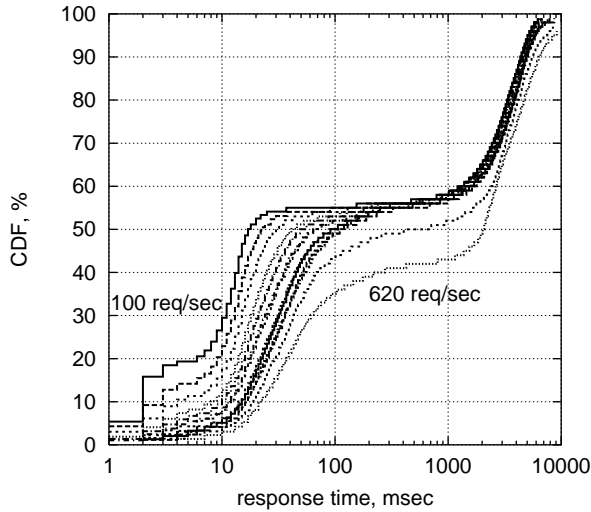
No-Proxy

Our 1-hour no-proxy test on this cluster showed that the network can handle at least 300 requests per second.

PolyMix #1

Request Rate (per second)	Mean Response Time (seconds)	Hit Ratio (percent)
100	1.41	55.2
200	1.42	55.0
300	1.43	55.0
400	1.45	55.0
450	1.48	55.0
500	1.52	55.0
520	1.56	55.0
550	1.61	54.9
575	1.68	54.9
590	1.73	54.9
600	1.80	54.9
610	2.36	54.7
620	3.00	54.5





H.4 Polyteam Comments

Failed runs This proxy failed to handle request rates above 620 req/sec.

Other One important limitation of Peregrine version tested during the bake-off is its inability to preserve cache contents between proxy restarts. Adding this essential feature may affect its performance.

A word of caution Participant Comments contain forward-looking statements concerning, among other things, future performance results. All such forward-looking statements are, by necessity, only estimates of future results and actual results achieved by the Participant may differ substantially from these statements.

H.5 Participant Comments

The Peregrine proxy is built as a part of the WisWeb research project funded by the National Science Foundation. The proxy is currently under evaluation by the University of Wisconsin-Madison for campus wide Web caching. The system will be available by September 1999. Though the system used in the bake-off employs an Alteon ACEdirector, the same performance can be achieved with a Netgear FS516 fast ethernet switch (price \$1000). The total cost of the cache plus network is \$8,296 in this case.

Performance of Peregrine systems can be scaled linearly via clustering. For example, four Pentium-based Peregrine systems connected with a load-balancing switch such as Alteon's CacheDirector or Foundry Networks' FastIron can offer potential throughput up to four times the throughput reported here. The designers of Peregrine avoided the "healing mode" in order to maximize the bandwidth savings of the cache, but the mode can be turned on via a configuration parameter. During the bakeoff, the system was able to handle experiments at request rate of 630-650 req/sec, though at such rates, about 2% of client connections were refused due to overload.

Appendix I

Squid

<http://squid.nlanr.net/>

I.1 Equipment

Description	Price
Squid, running on a generic Pentium-based PC, including:	\$0
(1) Pentium II 333 MHZ CPU	\$230
(1) Vextrec VTI-P6BIAK motherboard	\$140
(2) SD16X64-PC100 128MB SDRAM	\$620
(1) Trident T9680 VGA video	\$35
(1) Teac 1.44 floppy	\$25
(1) Western Digital WD-AC36400 6.4 GB IDE disk	\$140
(1) ATX desktop case	\$70
(1) Intel Etherexpress PRO/100+	\$55
(1) Adaptec 2940UW SCSI controller	\$290
(6) IBM DDRS-39130W 8.7-GB SCSI disks	\$1,950
(2) 4-bay disk enclosures	\$170
(2) SCSI cables	\$75
(1) 3COM TP800 8-port unmanaged fast ethernet hub	\$160
Total	\$3,960

I.2 Configuration

Cache disks: Total cache size is 30.6-GB, spread across six disks.

Software: Squid 2.2.DEVEL3.

The following configuration options were changed from their defaults:

```
max_disk_open_fd 200
memory_pools off
half_closed_clients off
cache_mem 128 MB
```

Polygraph machines: This cluster uses two (2) client-server pairs.

Network: All devices are connected to the 3COM repeater. Figure I.1 shows the networking diagram for this cluster.

All IP addresses are in the 10.18.11/24 network. Polygraph clients send requests directly to the cache.

Request Rates: PolyMix #1 tests were executed for the following request rates: 14, 40, 62, 65, 70, 72, 74, 75, 76, 80, 84, 88, 96, 100, 124, 132, 138.

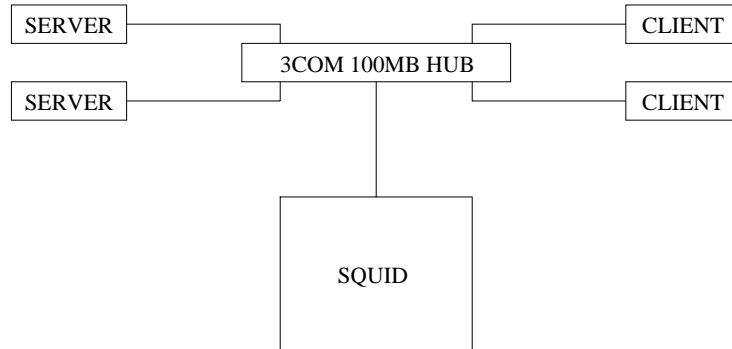


Figure I.1: Squid network diagram

I.3 Results

Network throughput

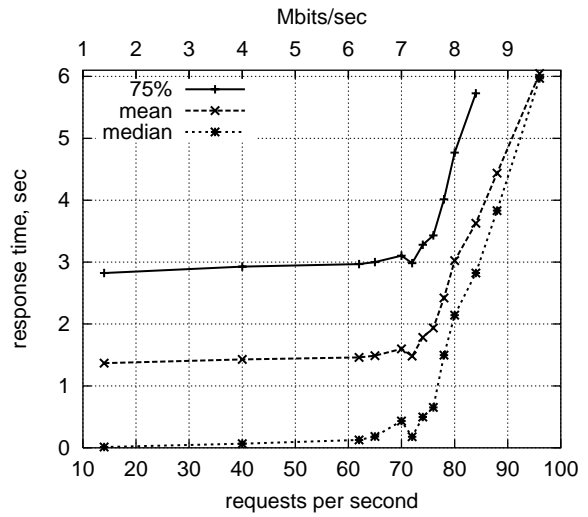
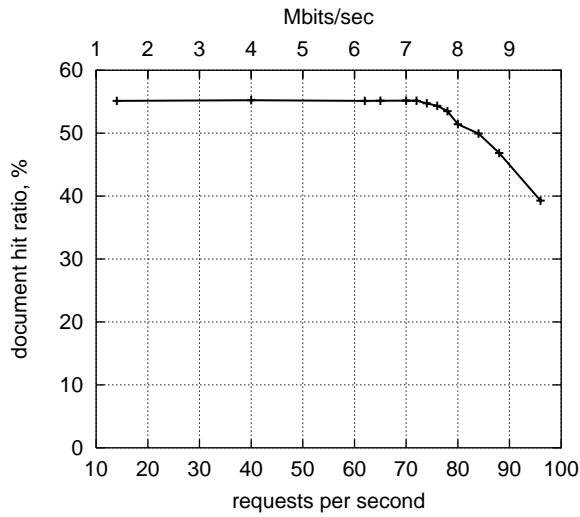
Our five-minute *netperf* test on this cluster showed the mean throughput between each client-server pair to be 37.0 Mbits/sec.

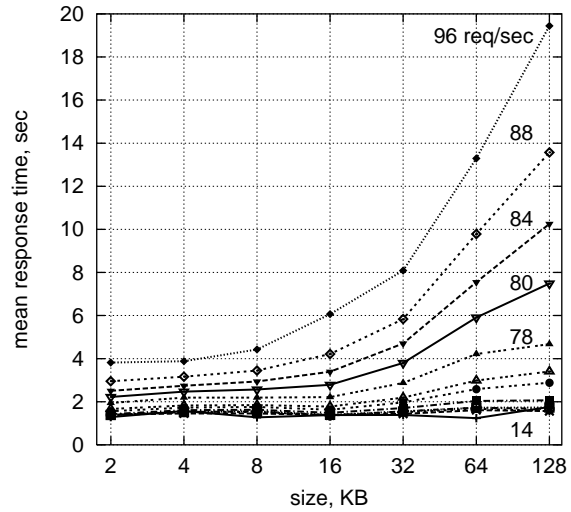
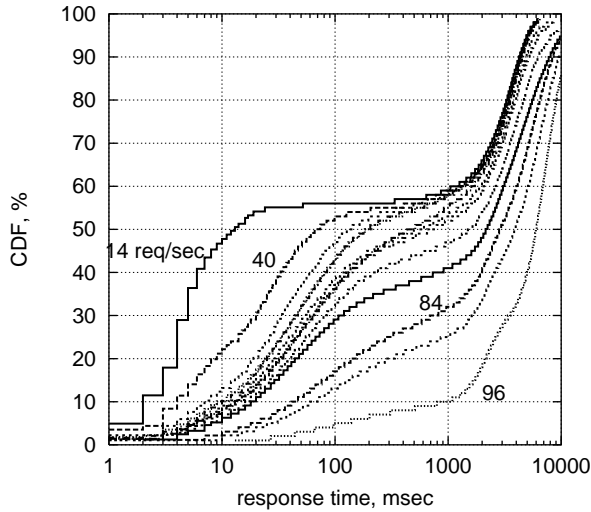
No-Proxy

Our 1-hour no-proxy test on this cluster showed that the network can handle at least 250 requests per second.

PolyMix #1

Request Rate (per second)	Mean Response Time (seconds)	Hit Ratio (percent)
14	1.37	55.1
40	1.43	55.2
62	1.46	55.1
65	1.49	55.1
70	1.60	55.2
72	1.48	55.1
74	1.78	54.8
75	3.64	51.1
76	1.94	54.3
78	2.42	53.5
80	3.02	51.4
84	3.63	49.9
88	4.44	46.8
96	6.05	39.3





I.4 Polyteam Comments

Failed runs This cache failed to handle request rates at 100/sec and higher.

Other Polyteam acknowledges the fact that Polyteam members are also Squid developers. To the best of our knowledge, we gave Squid no preferential treatment. In fact, our bake-off commitments absorbed virtually all time allocated to prepare Squid for the event.

I.5 Participant Comments

Squid was run with almost no modifications of default options. We had no time to experiment with various optimizations and decided to test an out-of-the-box performance. Cache administrators may be able to achieve even better results after performance tuning.

Squid team is currently working on several major performance improvements, including the long-awaited SquidFS. We decline the opportunity to speculate about the future performance, but are certainly looking forward for the next benchmarking event.

Appendix J

Verifying Benchmarking Environment

This Chapter describes the experiments that we run prior to the core bake-off tests. Recall that the purpose of these experiments is to verify that each benchmarking harness is configured correctly and is able to support required request rates.

J.1 Running *netperf*

To measure raw network bandwidth, we use a well-known network benchmarking tool called *netperf*. For the bake-off, we used the default netperf configuration and ran the bandwidth measuring test for at least 5 minutes. The test was run for every client-server pair. All pairs within a harness were tested simultaneously, placing maximum stress on the network. The results for each participant are reported in the corresponding Appendix.

The netperf server was run on port 5000. The netperf client was generating TCP traffic for at least 5 minutes. By default, the netperf client generates a single steady TCP stream.

```
% netserver -p 5000 # start netperf server on Poly-server machine, port 5000
% netperf -p 5000 -H $Server -l 300 # run netperf client on Poly-client
```

A typical netperf output for one client-server pair looks like:

```
TCP STREAM TEST to 10.11.11.11 : histogram
Recv  Send  Send
Socket Socket Message Elapsed
Size  Size  Size  Time  Throughput
bytes bytes bytes secs.  10^6bits/sec

16384 16384 16384 300.01 93.43
```

Increasing duration of an experiment from 5 minutes to 1 hour always gave us the same results.

J.2 No-proxy Tests

No-proxy tests are essential to guarantee that Polygraph machines and connecting network are not a bottleneck. We configure Poly-clients to talk directly to Poly-servers. Other Polygraph parameters

are the same that we use for the core experiment set. For the no-proxy experiment, we used 300 req/sec constant request rate unless a participant instructed us to lower the rate due to network bandwidth limitations. All core experiments used request rates lower than the no-proxy rate.

The no-proxy request rate for each participant is reported in the corresponding Appendix.

Appendix K

FreeBSD Tweaks

Running Polygraph at high request rates requires a number of operating system tweaks in order to perform really well. This list itemizes the changes that we made to FreeBSD-2.2.8 on the Polygraph clients and servers.

- Increased the per-process memory limits in the kernel configuration file:

```
options      "MAXDSIZ=(512*1024*1024)"
options      "DFLDSIZ=(128*1024*1024)"
```

- Increased the limit on mbuf clusters in the kernel configuration file:

```
options      "NMBCLUSTERS=40960"
```

- Disabled TCP delayed acknowledgments in the kernel configuration file:

```
options      TCP_ACK_HACK
```

- Increased the system-wide and per-process filedescriptor limits in `/sys/conf/param.c`:

```
define MAXFILES (16384)
int      maxfilesperproc = 12288;
```

- Decreased TCP's Maximum Segment Lifetime (MSL) value to three seconds in `/usr/src/sys/netinet/tcp_timer.h`:

```
#define TCPTV_MSL      ( 3*PR_SLOWHZ)
```

- Applied the following patch to `/usr/src/sys/kern/uipc_socket.c` to fix TCP bugs with small packets:

```
RCS file: /home/ncvs/src/sys/kern/uipc_socket.c,v
retrieving revision 1.40
retrieving revision 1.41
diff -p -u -r1.40 -r1.41
--- src/sys/kern/uipc_socket.c 1998/05/15 20:11:30      1.40
```

```
+++ /home/ncvs/src/sys/kern/uipc_socket.c      1998/07/06 19:27:14      1.41
@@ -491,6 +491,7 @@ restart:
```

```
                mlen = MCLBYTES;
                len = min(min(mlen, resid), space);
            } else {
+                atomic = 1;
nopages:
                len = min(min(mlen, resid), space);
                /*
```

- Increased the ephemeral port range to 1024-30,000 by adding this command to `/etc/rc.local`:

```
/usr/sbin/sysctl -w net.inet.ip.portrange.last=30000
```

- Increased the socket listen queue size by adding this command to `/etc/rc.local`:

```
/usr/sbin/sysctl -w kern.somaxconn=1024
```

- Set resource limits in `/etc/login.conf`:

```
default:\
    :cputime=infinity:\
    :datasize-cur=infinity:\
    :stacksize-cur=infinity:\
    :memorylocked-cur=infinity:\
    :memoryuse-cur=infinity:\
    :filesize=infinity:\
    :coredumpsize=infinity:\
    :maxproc-cur=infinity:\
    :openfiles-cur=infinity:\
    :openfiles-max=12288:\
    :priority=0:\
    :requirehome@:\
    :umask=022:\
    :tc=auth-defaults:
```

- Disabled unnecessary processes and services in `/etc/rc.conf`:

```
inetd_enable="NO"
portmap_enable="NO"
sendmail_enable="NO"
cron_enable="NO"
```