The Inktomi Climate Lab: An Integrated Environment for Analyzing and Simulating Customer Network Traffic

Stephane Gigandet, Ashok Sudarsanam, and Anshu Aggarwal
{stephane, ashok, anshu}@inktomi.com
Inktomi Corporation
4100 East 3rd Avenue, Foster City, CA 94404
tel 650 653 6980
fax 650 653 1848

Abstract-- **This paper describes the Inktomi Climate Lab, an in-house environment that analyzes and simulates the HTTP traffic at our customer deployments. This lab is used to verify the stability and measure the performance of our web proxy cache, Inktomi Traffic Server[1], which has been deployed at many customer sites. The ability to support all HTTP headers, methods, and status codes, as well as features such as pipelining and chunked-encoding, enables our Climate Lab to accurately simulate any deployment.**

Having an in-house Climate Lab that is able to reproduce the network traffic at any customer site affords several benefits. For instance, it gives us the ability to thoroughly stress-test the software before deployment, thus ensuring its high quality and robustness in the field. Also, it gives us the ability to quickly reproduce and resolve software issues that do occur in the field.

A. INTRODUCTION

We describe the **Inktomi Climate Lab**, an in-house environment that analyzes and models the HTTP traffic at our customer deployments, thus enabling us to accurately simulate the web traffic load, or *climate*, on our proxy caches at these deployments. Our lab takes its name from the agricultural laboratories whose greenhouses can be configured to simulate any kind of world climate in order to test new types of seeds[2]. Whereas these greenhouses can accurately reproduce the climate in different parts of California, the Sahara Desert, etc., our Climate Lab aims to accurately reproduce the network traffic load on the proxy caches at a dial-up ISP, a broadband ISP, a CDN, etc.

The **Inktomi Traffic Server**[1] web proxy cache has been deployed at many customer sites throughout the world. Having an in-house Climate Lab that is able to accurately simulate any customer site climate affords several benefits:

- It gives us the ability to thoroughly stress-test the software before deployment, thus ensuring its high quality and robustness in the field.
- It gives us the ability to quickly reproduce and resolve software issues that occur in the field.
- It allows us to accurately estimate the performance of our software in the field.
- It allows us to obtain a finer understanding of the various issues involved in web caching.

Our climate lab is composed of three components: a *log analysis* tool that analyzes logs from customer proxy cache deployments, and outputs various network traffic distributions; a *load generation* tool that generates network traffic load based on these distributions; finally, a set of dedicated high-performance machines that run Traffic Servers, synthetic clients, and synthetic servers.

This paper is organized as follows: in Section B, we compare the Inktomi Climate Lab to the *Web Polygraph* benchmarking tool[3]; in Section C, we describe the designs of the log analysis, load generation, and hardware components of the Climate Lab; in Section D, we present several case studies that detail the application of our Climate Lab to various customer proxy deployments; finally, we present our conclusions in Section E.

B. COMPARISON WITH PREVIOUS WORK

In this section, we compare our Climate Lab with **Web Polygraph**[3], which is the most widely used benchmarking tool for proxy caches and is freely available as open-source software. Specifically, we explain why we chose not to use Web Polygraph for load generation purposes.

Although Web Polygraph and the Inktomi Climate Lab both simulate web clients and servers, they emphasize different goals: the former emphasizes *benchmarking*, while the latter emphasizes *testing*. Specifically, Web Polygraph is used to compare proxy caches by measuring an indication of their performance, while the Inktomi Climate Lab is used to uncover coding flaws in Inktomi Traffic Server. The following differences in design clearly highlight the different end-goals of these two environments:

- Since the Climate Lab aims to exercise every control path of Traffic Server code, it provides *complete* support for all HTTP protocols[4,5,6]. In particular, the Climate Lab is able to generate web traffic that includes all HTTP methods, request and response headers, and response status codes. In contrast, the web traffic that Web Polygraph is capable of generating includes support for the HTTP GET method only, and just a few headers and status codes.
- Both Web Polygraph and the Inktomi Climate Lab simulate network conditions such as initial delay, burstiness, bandwidth limitations, etc. Web Polygraph relies exclusively on FreeBSD's **Dummynet** tool[7] to simulate these network conditions at the *kernel* level. In order to achieve greater flexibility, our Climate Lab simulates these network conditions at the *application* level, thus enabling us to simulate differing and changing network conditions across, and even within, client sessions. We have observed, however, that operating system buffering does interfere with our application-level implementation. Consequently, we are in the process of evaluating new solutions by coupling our implementation with Dummynet, as well as hardware WAN simulators such as Shunra Storm[8].
- The most important design difference is that the Inktomi Climate Lab features a *log analysis* component while Web Polygraph does not. For benchmarking purposes, Web Polygraph provides several versions of two well-defined workloads, **PolyMix** and **WebAxe**, that are used to evaluate the performance of every caching product. However, in order to fully test a web caching product, an environment must be able to generate many different workloads. Ideally, each customer deployment should have a corresponding workload. Furthermore, given the highly dynamic nature of the Internet, each customer deployment should have multiple corresponding workloads. The log analysis component of the Inktomi Climate Lab aims to generate representative workloads in a practical manner. As will be described in the following section, this component analyzes customer cache access logs in order to create representative workload plans.

Web Polygraph does implement some interesting features that are currently not implemented in the Climate Lab, such as the ability to vary workload over time and DNS simulation. However, the primary reason why the Climate Lab does not interface directly to Web Polygraph and automatically generate Polygraph configuration files is because of our need for complete HTTP support during load generation.

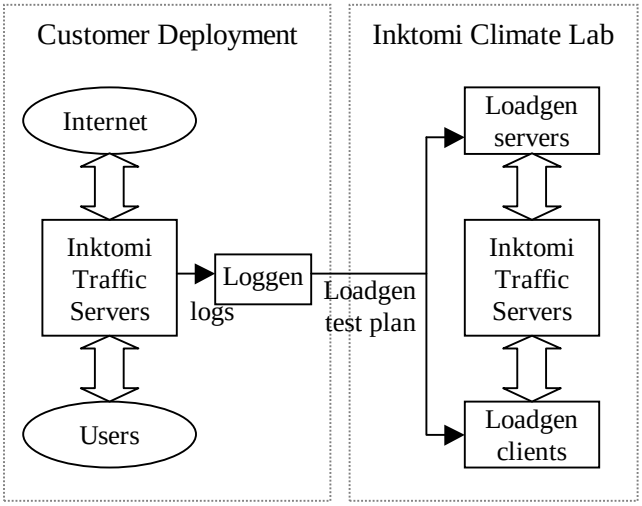

Fig. 1. Interaction between the components of the Inktomi Climate Lab.

C. CLIMATE LAB DESIGN

In order to fully test our web proxy cache, it is necessary to have a tool that simulates both web clients and servers. We have developed a load generation tool that can simulate thousands of clients and servers on a single machine. This tool is configured through *test plans*, which specify the distributions of various web traffic parameters. These test plans are generated by performing log analysis on the logs from customer proxy cache deployments. The overall design of our Climate Lab is illustrated in Figure 1. We now describe in detail the log analysis, load generation, and hardware components of our Climate Lab.

*1st. Log Analysis Tool: **Loggen***

Our log analysis tool, *Loggen*, generates test plans that shape the web traffic that is generated by our load generation tool, *Loadgen*. We have used Traffic Server's built-in custom logging facility to enable our customers' caches to log a variety of useful information. This information, which is unavailable in standard *Squid* log files[9], includes the set of all HTTP headers in each client request and server response, the size of each request and response body, and transfer times. Loggen also supports standard Squid log files at the expense of using default values for most header distributions.

Loggen then parses the custom logs and computes various statistics, such as the probability of a specific HTTP header being in a client request or server response. More complex properties, such as the frequency of a document, can be defined using *Zipf-like* distributions[10]. However, fitting log data to theoretical distributions is a very complex task (see the discussion on the *SURGE* Scalable URL Request Generator[11]). Thus, to model some properties such as the document size, we are currently using distributions of discrete intervals for which parameters and pseudo-random numbers are easier to compute and that can model non-expected distributions from a particular customer.

Loggen also keeps track of individual client sessions to compute distributions for the number of clicks per session and the number of requests per clicks and to estimate network conditions for the client and server.

It is important to note that we are primarily interested in the network load as seen by the proxy. For instance, other proxies or dynamic allocation of IP addresses might result in two client sessions by two different individuals being identified as one, but it is consistent with what the proxy actually sees.

The log analysis results are then used to generate an XML-based test plan that Loadgen uses to generate web traffic. A small extract of a sample test plan is presented and described in Figure 2.

It should be noted that for user confidentiality purposes, customer logs are *not* provided to us. Instead, log analysis is performed at the customer site itself, and only the resulting test plans that contain distributions but no actual access data are provided to us.

2nd. Load Genation Tool: ***Loadgen***

Our load generation tool, *Loadgen*, is able to simulate both clients and servers. For performance reasons, it is implemented as a single-threaded process that constantly polls client and server connections within a tight loop. There are three reasons why we chose to simulate web servers within Loadgen, as opposed to replaying the logged URLs and allowing Inktomi Traffic Server to contact the origin servers directly:

- The amount of bandwidth needed to fully stress-test a proxy cache on real URLs is essentially unaffordable.
- It is considered very bad etiquette to send a large amount of traffic to real web sites for testing purposes.
- Stress-testing becomes non-deterministic due to origin server down-times and content changes, etc.

```
<header>
 <name>Cache-Control</name>      <p>0.05</p>
 <multiple_value>
   <name>public</name>           <p>0.10</p>
   <name>private</name>          <p>0.10</p>
   <name>no-cache</name>         <p>0.10</p>
   <name>no-store</name>         <p>0.10</p>
   <name>no-transform</name>     <p>0.10</p>
   <name>must-revalidate</name>  <p>0.10</p>
   <name>proxy-revalidate</name> <p>0.10</p>
   <name>max-age=60</name>       <p>0.10</p>
   <name>s-maxage=45</name>      <p>0.10</p>
   <name>community="INKT"</name> <p>0.10</p>
 </multiple_value>
</header>
```

Fig. 2. Specification of the Cache-Control request header in a sample XML test plan: the Cache-Control header is present in 5% of the requests and its arguments are comma-separated values, which have a 10% probability. If none are selected, then one is chosen at random.

Each simulated client attempts to reproduce the behavior of a user interacting with a browser. Each client session consists of one or more *clicks* on a particular web-site -- the individual clicks are separated by a *user-think* time. To model temporal locality, the web-site is chosen according to a site popularity distribution. To model spatial locality, all the documents requested during the session are considered to belong to the site and are chosen according to a page popularity distribution. The requested url is a combination of the site and the page identifiers.

Each click begins by requesting a document from the chosen web-site. As soon as the client begins to receive the response body, it may generate additional requests for more documents from this site. These requests correspond to images and files that are embedded within the initially-requested document. The client may send these requests to the server using a combination of the following three methods: establishing a new connection to the server for each request, reusing the existing connection to send a new request after the response has been received, and pipelining all requests over a single connection.

In order to be able to generate realistic web traffic, Loadgen provides support for every feature that is defined in the HTTP specifications[4,5,6], such as persistent connections, request pipelining, chunked-encoding, and range requests. Additionally, client requests and server responses are capable of employing every request method, header, and response status code defined in the HTTP specifications. A very important feature that has been implemented in Loadgen is the ability to maintain *state* information, which includes:

- Managing cookie jars for each client session so that clients can send back cookies that have been sent by a particular web-site.
- Managing *Last-Modified* times and *ETags* so that servers can respond realistically to *If-Modified-Since* and *If-None-Match* conditional requests.

Loadgen also has the ability to simulate network conditions such as initial delay, burstiness, and bandwidth limitations. We chose to model network conditions at the application level rather than using hardware WAN simulators[8] or kernel-level simulators such as Dummynet[7]. This choice of implementation enables Loadgen to simulate completely different network conditions for each client session, and to also dynamically change them within a single session.

Finally, web proxy caches must be robust enough to handle all abnormal conditions, such as broken HTTP implementations in clients and servers and unexpected aborts. Consequently, Loadgen provides support for generating HTTP headers and bodies that have been corrupted in a number of ways (e.g. errors in time-based header values[12]), and the ability of clients and servers to abort transactions at any time.

3rd. Hardware and Network Configuration

We have dedicated a total of 16 high-performance machines to the operation of the Inktomi Climate Lab: 8 *Solaris* PCs, 6 *SUN E220*s, and 2 *Linux* PCs. All machines belong to the same sub-net and are configured to use virtual IP addresses. A single machine can simulate thousands of clients and servers which use up to 256 different IP addresses. Furthermore, depending on the desired traffic, one machine can generate up to 1,000 HTTP transactions per second.

D. CASE STUDIES

We have generated and analyzed custom web traffic logs at two major ISP deployments. We have also analyzed the logs of our own internal proxy.

Customer A logged 3 million operations during a 2 hour period, Customer B logged 2 million operations during a 24 hour period and our proxy (referred to as Customer C) logged 330,000 operations during a 24 hour period. Note that the legend in Fig.3 also applies to all other figures. In the frequency density graphs, the area below the graph is proportional to the number of samples whose values are in a given range of the X axis.

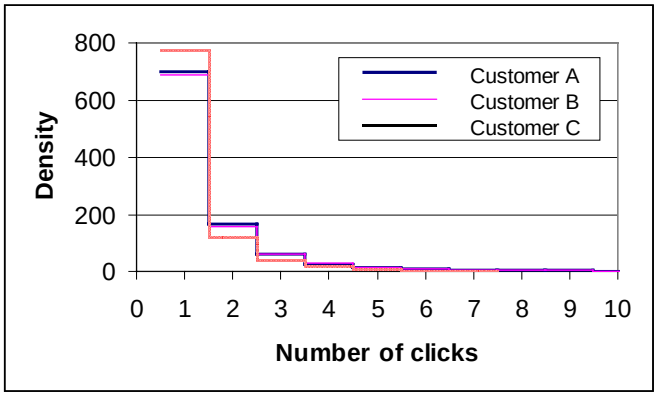

Fig. 3. Frequency density of the number of clicks distribution.

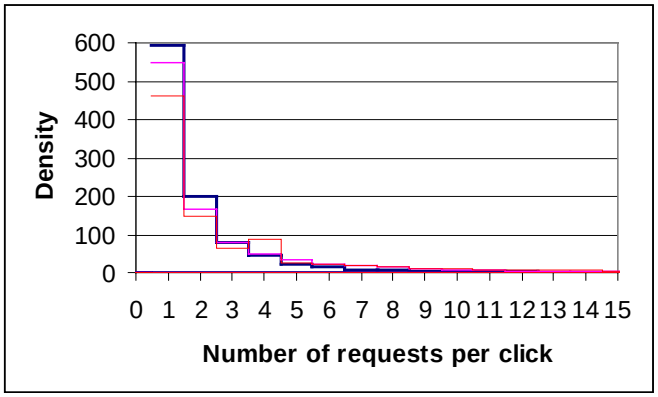

Fig. 4. Frequency density of the number of requests per click distribution.

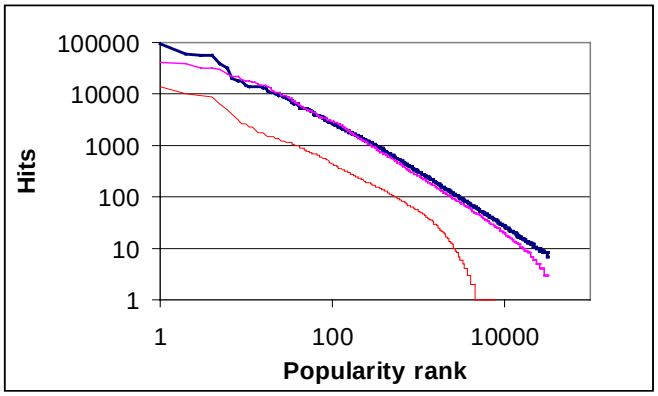

Fig. 5. Site popularity distribution. (double log scale)

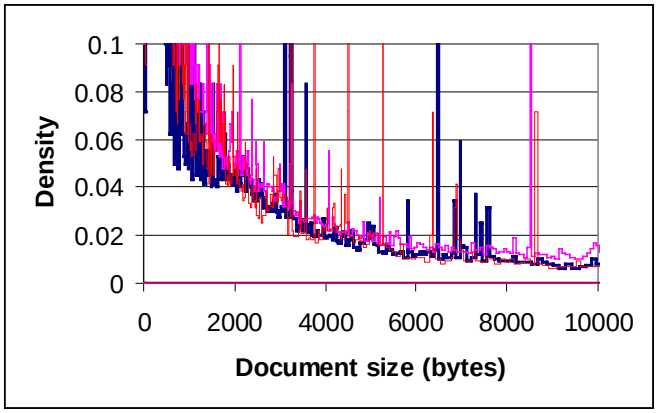

Fig. 6. Frequency density of the document size distribution.

In these figures, we present the number of clicks, number of requests per click, site popularity and document size empirical distributions because they can be rendered visually and they illustrate our click model. The following observations can be made:

- The distributions of the three deployments are similar.
- The site popularity distribution plotted in a double algorithmic scale forms a line. This verifies earlier observations[10] that site popularity follows *Zipf's Law*. The slope is similar for the three deployments.
- The document size distributions show numerous spikes. One could reason that those spikes correspond to the most popular documents. However, the spike locations are different for the three deployments. A more in-depth analysis of Customer C's logs has shown that they result from very few individuals who access pages that auto-update very frequently such as stock quotes or more customer specific pages such as our own real time proxy statistics.

We have used test plans generated by Loggen and sample test plans to configure Loadgen, and it has uncovered several software flaws that other internal load generation tools did not previously uncover. This is primarily due to the fact that unlike other tools, Loadgen provides complete support for all HTTP methods, headers and status codes. The flaws included small memory leaks that occurred in rare occasions and bad handling of unexpected headers and status codes.

We are currently in the process of validating the Climate Lab model. Although quantifying the degree of accuracy of the reproduction of customer deployments is a very complex issue, we have identified several possible approaches:

- Compare the caching proxy statistics generated during the log collection and during a Climate Lab simulation.
- Compare the test plan resulting from the analysis of customer logs with the test plan resulting from the analysis of logs collected during a simulation using the first test plan. Using this approach, we confirmed that due to insufficient timing data, the network condition distributions currently computed by Loggen were inaccurate.
- Attempt to reproduce bugs reported by customers on previous versions of Traffic Server.

E. CONCLUSIONS

We have presented the Inktomi Climate Lab, a powerful in-house environment that analyzes and simulates real-world HTTP traffic. The Inktomi Climate Lab represents our efforts at applying Internet traffic measurement to solving real-world problems. In particular, the ability to accurately reproduce in-house the HTTP traffic at any customer deployment allows us to stress-test our Inktomi Traffic Server proxy cache with a very high degree of confidence that no software issues will arise in the field (in fact, our Climate Lab may be used to stress-test *any* web proxy caching product). In the event that software issues do arise in the field, our Climate Lab allows us to quickly reproduce and resolve these issues.

We have generated custom logs at several major ISP deployments as well as at our own internal deployment. The results of performing log analysis on these logs have enabled us to simulate several real-world climates. We are currently working on evaluating the accuracy of these simulations, but even in its present state, the Climate Lab has already been deployed in-house, and is being used to uncover bugs and test the stability of Traffic Server.

References

[1] *Inktomi Traffic Server*. http://www.inktomi.com.

[2] *Monsanto.com*. http://www.monsanto.com.

[3] *Web Polygraph*. http://www.web-polygraph.org.

[4] W3C. *Hypertext Transfer Protocol – HTTP/0.9*. 1991.

[5] T. Berners-Lee, R. Fielding, and H. Frystyk. *Hypertext Transfer Protocol – HTTP/1.0*. RFC 1945, May 1996.

[6] R. Fielding, J. Gettys, J.C. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616, June 1999.

[7] L. Rizzo. *Dummynet: a simple approach to the evaluation of network protocols*. ACM Computer Communication Review 27, 1. January 1997.

[8] *Shunra Storm WAN Simulator*. http://www.shunra.com.

[9] *Squid Web Proxy Cache*. http://www.squid-cache.org.

[10] L. Breslau, P. Cao, L. Fan, G. Phillips and S. Shenker. *Web Caching and Zipf-like distributions: Evidence and Implications*. IEEE INFOCOM, March 1999.

[11] P. Barford and M. Crovella. *Generating Representative Web Workloads for Network and Server Performance Evaluation*. ACM SIGMETRICS, June 1998.

[12] J.C. Mogul. *Errors in timestamp-based HTTP header values*. WRL Research Report, December 1999.