Continuous Measurement of Web Proxy Cache Efficiency

Simon Patarin
INRIA — REGAL group
Domaine de Voluceau, B.P. 105
Rocquencourt, France
simon.patarin@inria.fr

Mesaac Makpangou
INRIA — REGAL group
Domaine de Voluceau, B.P. 105
Rocquencourt, France
mesaac.makpangou@inria.fr

Categories and Subject Descriptors

H.3.5 [**Online Information Services**]: Web-based services; G.3 [**Probability and Statistics**]: statistical computing

General Terms

Algorithms, Design, Experimentation, Measurement, Performance

Keywords

Network monitoring, World Wide Web, Proxy caches, Evaluation

1. INTRODUCTION

Web proxy caches are deployed almost everywhere, at organisation boundaries as well as at ISPs. Proxy caches improve latency when retrieving Web documents while reducing the overall network traffic on the Internet. Proxy cache administrators spend some effort configuring the caching software and determining their suitable placements in order to obtain the best achievable quality of service, with respect to user demands. However, despite this effort, the efficiency of caches may decline over time. Indeed, the efficiency of a given caching infrastructure depends heavily on the actual use of the caches: number of users, type of requested documents, load of machines running caches. Since these characteristics are subject to change, it is unlikely that any particular configuration and deployment will remain effective during the whole cache life cycle. Therefore, to maximise the benefits offered by Web proxy caches, administrators should reconfigure the existing infrastructure as the traffic characteristics evolve; this implies frequent *measurements* of cache *efficiency*.

This raises two distinct issues. First, how often should one perform measurements? With respect to the frequency of evaluations, it is important to be able to react to sudden modifications (flash crowds, network or servers failures), as well as slower ones with strong trends (growing user community, increasing use of multimedia documents). The appropriate frequency depends mainly on the impact of the cache's efficiency on the actual quality of service. Hence the best choice of frequency is particular to each system. Each administrator should be given a way to adapt the measurement frequency of his caches, taking into account the objectives in terms of overall service improvement and the effect of the degradation of performance. Second, how to cope with the inherent subjectivity of the notion of efficiency? The suitable efficiency metric depends on the goals defined by the cache administrators; the same level of performance may lead to different experiences by different people in different environments. It must be customisable. Examples of basic metrics that can be used include the consistency of the returned documents, bandwidth savings and latency savings.

Regardless of which metric is considered, the measurement of cache efficiency requires three distinct steps. First, one needs traces recording the behaviour of the system of caches. Two different methods can be used to obtain these traces. The obvious one is to use log files that all caching software generates. However, this technique is overly simple because those logs often lack information and, even worse, sometimes contain inaccurate information [1]. To circumvent these drawbacks, one needs to instrument the software; however, this is not always possible: source code may not be available, or it may require a large amount of work. Another approach to build such traces is to use passive traffic monitoring. Several tools performing HTTP traffic extraction from raw network packets have been proposed in the past. These include BLT [2] and Windmill [3], which can collect information pertinent to the measurement of all necessary metrics. However, none of them consider the biases introduced by caches. Hence, the traffic characteristics they would capture in presence of Web proxy caches are corrupted, leading to inaccurate measures of cache efficiency. The second stage consists of analysing the traces to extract suitable basic metrics. This is commonly achieved by cache log analyser software such as Calamaris or Squeezer. They produce statistical summaries of the given log file in plain text or HTML format. Other packages like WebLog provide software components (Python classes for WebLog) to manipulate logs and perform their own analysis. The final step combines the basic metrics previously extracted to produce the actual efficiency measurement. Trace-driven simulation tools [4, 5, 6] are widely used. However, these tools are usually dedicated to the evaluation of a limited number of metrics. This makes it difficult to extend them in order to take into account metrics that were not thought of beforehand. Furthermore, it is difficult to perform such simulations in real-time, which does not fit our design goals.

What stems from this quick review of existing tools related to cache evaluation is that there does not exist a single tool that integrates all of these three stages. Consequently, it is often necessary to have glue software inserted between two consecutive steps. This also requires each piece of software to be executed one after the other, complicating their use on a continuous (and automated) basis. Finally, the diversity of tools used makes it difficult to have them deployed easily on a wide range of platforms and environments.

2. WEB PROXY CACHE EFFICIENCY

In a previous paper [7], we described the design and the implementation of Pandora, our flexible monitoring platform. Our focus here is to present how this can be used to build a well-integrated

Copyright is held by the author/owner(s).
WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM xxx.

tool for continuous efficiency measurements of proxy caches. Pandora allows the collection of unbiased metrics for an arbitrary complex system of caches. Pandora uses passive traffic capture and online HTTP trace extraction to monitor the behaviour of the caching system. To correct the modifications introduced by proxies on the traffic, it performs active probes. Then, this trace is analysed on the fly to compute the various base metrics it uses to evaluate the efficiency of the system. Those metrics include document consistency and cooperation efficiency which are of particular interest because they are not easily captured by legacy analysis software.

As soon as the HTTP traffic trace is collected, it is used to perform the measurement of the efficiency of the system of proxy caches. To this end we define a set of metrics that maps their input (cache or ICP transactions) into unitless decimal numbers. Negative values indicate harmful configurations (i.e., those that actually *degrade* efficiency), positive values denote improved quality of service and 0 corresponds to a situation where caches were not present (which should be the expected minimum of the function in real conditions).

Round-trip time is the time needed for a request to be completed, from the *first* byte of the request until the *last* byte of the response. To compute the metric, we consider the difference between the round-trip time for the *server* transaction and the *client* transaction. In the case of a miss, we expect this to be lightly negative, whereas it should be largely positive in case of hit. Then we compute the ratio of this time with the round-trip time of the *server* transaction. Hence the value computed is equal to the percentage of improvement (or degradation) compared to the situation without the proxy.

Latency is the time spent between the *last* byte of the request and the *first* byte of the response. This corresponds to the time a user will have to wait before seeing anything in her Web browser (assuming that the browser starts displaying data as soon as it receives it). It captures both the network latency and the latency of both servers: the Web server and the proxy cache. This metric is computed like the previous one by considering the percentage of improvement compared to the no-proxy case.

Bandwidth savings are simply the number of bytes transferred for each transaction, including the headers of the request and the response. The same percentage is computed as for the above metrics, by making the ratio between the number of bytes saved (or wasted) by the proxy and the total number of bytes for the transaction without the proxy. One must note that proxy caches usually add a few headers to the requests they emit. Also, they may transform, in some circumstances, plain GET requests into "if-modified-since" ones.

Document consistency aims at capturing the freshness of the documents returned by the cache. Indeed when a cache returns a document it holds, there is no warranty that it has not been updated on the original server. Usually Web servers include in their response a `Last-Modified` header giving the most recent modification time of the document. When determining if a document is stale, three timestamps are to be considered: t_p, the last modified timestamp of the document returned by the proxy, t_o the last modified timestamp of the original document and t_r the timestamp at which the request was made. If $t_p = t_o$, the document is consistent. If $t_p < t_o < t_r$ the document returned by the cache is stale. If $t_p < t_r < t_o$ we cannot conclude anything. The last case may happen because we cannot ask the server precisely at the time the client request was done: there is always a small delay (necessary to decide whether the lack of transaction between the proxy and the server is really a hit). By reducing the delay, we reduce the probability of this case, but we cannot eliminate it completely. Concerning this metric, we compute the ratio of stale documents with the total number of hits.

Cooperation efficiency: when proxies are cooperating, they necessarily exchange control information in order to know each other's cache content. To capture the efficiency of this cooperation, we compute the ratio of the number of document bytes fetched between peers with the number of control bytes.

3. CONCLUSION

Compared to existing software, Pandora proposes a well integrated support for the collection, analysis and evaluation stages of efficiency measurement. Moreover, it is easy to take into consideration new metrics for this evaluation thanks to the flexibility of Pandora. It is usually the matter of a few lines of code to add a new component that will extract the information and format it for Pandora. Moreover, the fact that Pandora operates online allows one to evaluate running systems with no perceptible perturbations for their users.

Among the metrics we have studied, document consistency and cooperation efficiency gives new insight into the understanding of the efficiency of a complete proxy cache system. Combined with other "standard" metrics (round-trip time and bandwidth improvement) it allows to adapt the configuration of the system with greater precision.

4. REFERENCES

[1] Brian D. Davison, "Web traffic logs: An imperfect resource for evaluation," in *Proceedings of th INET'99 Conference*, June 1999, `http://www.isoc.org/inet99/proceedings/4n/4n_1.htm`.

[2] Anja Feldmann, "BLT: Bi-Layer Tracing of HTTP and TCP/IP," in 9^{th} *International World Wide Web Conference*, Amsterdam, The Netherlands, May 2000, `http://www.www9.org/w9cdrom/367/367.html`.

[3] G. Robert Malan and Farnam Jahanian, "An extensible probe architecture for network protocol performance measurement," in *Proceedings of ACM SIGCOMM '98*, Vancouver, British Columbia, September 1998, `http://www.eecs.umich.edu/~rmalan/publications/mjSigcomm98.ps.gz`.

[4] Junho Shim, Peter Scheuermann, and Radek Vingralek, "Proxy cache design: Algorithms, implementation and performance," *IEEE Transactions on Knowledge and Data Engineering*, 1999, `http://www.ece.nwu.edu/~shimjh/publication/tkde98.ps`.

[5] Syam Gadde, Jeff Chase, and Michael Rabinovich, "A taste of crispy Squid," in *Proceedings of the Workshop on Internet Server Performance (WISP'98)*, June 1998, `http://www.cs.duke.edu/ari/cisi/crisp/crisp-wisp.ps.gz`.

[6] Roland P. Wooster and Marc Abrams, "Proxy caching that estimate page load delays," in *Proceedings of the 6rd International WWW Conference*, Apr. 1997, `http://www.scope.gmd.de/info/www6/technical/paper250/paper250.html`.

[7] Simon Patarin and Mesaac Makpangou, "Pandora : A flexible network monitoring platform," in *Proceedings of the USENIX 2000 Annual Technical Conference*, San Diego, June 2000, `http://www-sor.inria.fr/publi/PFNMP_usenix2000.html`.