P-Jigsaw: Extending Jigsaw with Rules Assisted Cache Management

Bin Lan[+]
Microsoft Corporation
One Microsoft Way
Redmond, WA98052-6399 USA
davelb@microsoft.com

Stéphane Bressan
Department of Computer Science
National University of Singapore
3 Science Drive2, Singapore117543
steph@comp.nus.edu.sg

ABSTRACT
P-Jigsaw is an extension of W3C's Jigsaw Web-server implementing a cache management strategy for replacement and pre-fetching based on association rules mining from the access-log.

Keywords
Web server, caching, pre-fetching, association rules, Jigsaw.

1. INTRODUCTION
We present our approach to caching documents at the Web-server's side [8]. We propose a solution offering original pre-fetching and replacement policies leveraging knowledge mined from the access-logs in the form of association rules**.**

1.1 Web Caching and Pre-fetching
Traditional file system caches do not perform well when serving Web requests [1]. Nevertheless existing Web-servers rely on the file system's cache or implement similar generic strategies such as Least Recently Used (LRU) for their own cache. This observation compels the integration into Web servers of dedicated caching and, possibly, pre-fetching mechanisms. Several authors have proposed to use access-log information to learn replacement and pre-fetching strategies. In [2], Arlitt showed, for instance that a frequency based strategy outperforms LRU for caching. Similar results for pre-fetching were presented in [7] by Tatarinov et al.

1.2 Web Mining
The application of data mining techniques to the Web, referred to as Web-mining [4], has recently received an increased attention. Of particular interest for us, is a series of approaches attempting to exploit the specificity of access patterns to hyper-documents. For instance, in [3], Bestavros estimates the probability of documents to be requested next from the information recorded in the access-log. The author uses this probability to push documents to the client side. However, so far and to our knowledge, there has been no other proposal but ours to apply Web mining to caching and pre-fetching at the server's side.

2. Rule-Assisted Cache Management, RAC
The technique we propose, called Rule-Assisted Cache management, RAC, is based on the mining of association rules from the access-log [5].

[+]This research was done while the author was a PhD student at the National University of Singapore

2.1 Mining the Log
Caching and pre-fetching strategies leverage either knowledge or hypothesis about access patterns. In most Web servers a comprehensive history of accesses is recorded in the access-log. We call a transaction the chronological list of entries for a given user-agent over a given period of time. A transaction is a projection of a portion of the access log. Looking at all transactions, we extract rules of the form $D_i \rightarrow D_j$ where D_i and D_j are document references (urls). The intuitive interpretation of such rules is that, from past experience, document D_j is likely to be requested by a user sometimes after he or she requests D_i. These rules are association rules. The quality of a rule can be measured by its support and confidence. The two numbers characterize the amount of information supporting the rule, and the amount of positive evidence gathered, respectively.

2.2 Rule Assisted Pre-fetching, RAP
If a user-agent requests a document D_I then the association rules of the form $D_i \rightarrow D_j$ which we have mined from the access-log, suggests that this user-agent is likely to eventually request document D_j. In anticipation of this event, the Rule Assisted Pre-fetching strategy pre-fetches one of the D_j documents from the disk into the cache. RAP could use the rules $D_i \rightarrow D_j$ only. However, considering D_i alone ignores other user-agent requests for other documents. Therefore RAP maintains a set of active documents, i.e. the documents last requested by user-agents within a given time window. The pre-fetched document D_j is then determined by the rule $D_k \rightarrow D_j$ with the highest confidence, for which D_k is an active document, such that D_j is not already in the cache, and whose support is above a given threshold.

2.3 Rule Assisted Replacement, RAR
Eventually the cache is full. When a request for a document not in the cache is received or a decision of pre-fetching a document is made, the documents least likely to be requested next should be removed from the cache to free the necessary space. RAR decides the document to be sacrificed. Similarly to RAP, it uses the rules $D_k \rightarrow D_j$ where D_k is an active document and D_j is in the cache. However, it sacrifices the document(s) D_j for which there is no rule or such that the rule has the lowest confidence.

2.4 Simulation Results
Using three independent workloads (access-logs): NASA, ClarkNet [2] and SF100 [5] we compare our strategy with the most competitive replacement and pre-fetching strategies available. Our Rule-Assisted Replacement strategy, RAR, was compared, among others, to LRU, LRU-MIN [9], LFU-MIN [5], and OPT (theoretical optimal strategy) In the simulation, our strategy dominates all practical strategies and is near-

optimal in terms of hit-rate (Figure 1). RAP was compared to STATIC [7], and other strategies leveraging the hyper-link structure of the hypertext (called Hyper-Link-Random, HLR, and Hype-Link-All, HLA) (Figure 2). Again our strategy performs best.

3. P-Jigsaw

Jigsaw is W3C's Web server platform. The Jigsaw Web server [6] implements full HTTP/1.1. Jigsaw is open-source and its architecture is modular. It is implemented in Java. Its design is object-oriented making the system easily extensible. P-Jigsaw implements RAC as an extension of Jigsaw.

3.1 Cache Management in Jigsaw and P-Jigsaw

The cache replacement policy of the Jigsaw Web server is Least Recently Used. The cache management is also controlled by three parameters: the maximum authorized size for files to be stored in the cache, the maximum number of files kept in the cache, the maximum retention time for a given file. The administrator sets these parameters using Jigsaw's administration interfaces. In P-Jigsaw both the replacement policy and the pre-fetching policy are based on the rule-assisted framework we are proposing. In P-Jigsaw the cache management is therefore controlled not only by the three parameters available to the Jigsaw administrator but also by several other parameters defining the rule mining, pre-fetching, and replacement strategies. For example, there are parameters to control the maximum number of documents that can be pre-fetched or the support threshold of the rules. The administrator can set these parameters using P-Jigsaw's administration interfaces. For instance, Figure 3 is a screen shot of the interface for the stetting of the rule mining parameters: minimum support, minimum confidence, etc.

3.2 P-Jigsaw Implementation

P-Jigsaw is an extension of Jigsaw 2.0. It is relatively easy for programmers to add new features to the Jigsaw server. The management of resources, such as files, directories, request, or the cache can be adapted by writing the corresponding resource object class. In the Jigsaw terminology, resources have frames and filters implementing their interaction with other resources. We have modified the cache filters to implement the pre-fetching and replacement policies. The rule- mining module is an additional resource. The administrator interface was also extended to allow the setting of the mining, pre-fetching and caching parameters.

3.3 Performance Evaluation

For the evaluation of the performance of the P-Jigsaw implementation, we use the SF100 workload for which we have both the access-log and the hypertext documents. We compare RAC as implemented in P-Jigsaw with Jigsaw both without a cache and with its replacement policy LRU. The results on Figure 4 show that RAC yields a 30% improvement of the average response time of the Web-server compared to LRU, consistently with the simulation.

4. Conclusion

The P-Jigsaw prototype can be downloaded from http://www.comp.nus.edu.sg/~icom/P-JIGSAW.

5. REFERENCES

[1] C. Aggarwal, J. L. Wolf and P. Yu. Caching on the World Wide Web. *TKDE*,11(1):94-107,99

[2] M. Arlitt. A Performance Study of Internet Web Servers. *Master's Thesis, Uni. of Saskatchewan, Canada*, 96

[3] A. Bestavros. Using speculation to server load and service time on the WWW. In *CIKM*, 95

[4] R. Cooley, B. Mobasher and J. Srivastava. Web mining: Information and pattern discovery on the World Wide Web. In *Proc.of the 9th IEEE Conf. On Tools with AI*, 97.

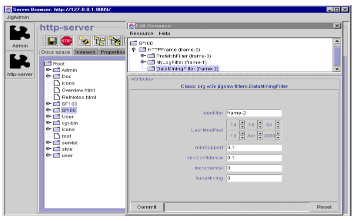
[5] B. Lan, S. Bressan, B. C. Ooi and K. Tan . Rule-assisted prefetching in Web-server caching. In *CIKM*, 00

[6] Jigsaw-W3C's Web server. http://www.w3.org/Jigsaw/.

[7] I. Tatarinov, A. Rousskov, V. Soloviev. Static Caching in Web servers. In *IC3N*, 97

[8] J. Wang. A survey of Web caching schemes for the Internet. *ACM SIGCOMM Computer:Communication Review*,29(5), 99

[9] S. Williams, M. Abrams, et al. Removal Policies in Network Caches for World-Wide-Web Documents. In *Proc. of ACM SIGCOMM*, 96.



Class: org.w3c.jigsaw.filters.DataMiningFilter

Identifier: frame-2
Last Modified: 14 16 54 18 Apr 2000
minSupport: 0.1
minConfidence: 0.1
incremental: 0
forceMining: 0

Figure 3.

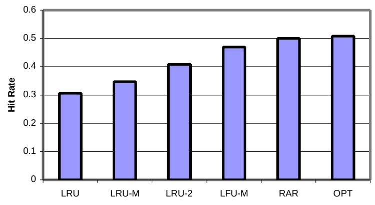


Figure 1

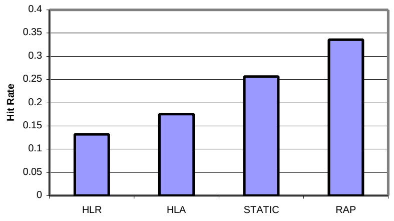


Figure 2

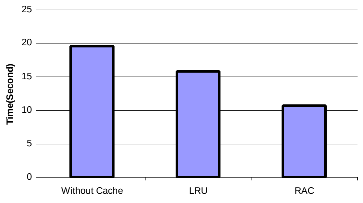


Figure 4