Association Rule Based Data Mining Agents for Personalized Web Caching

Sujaa Rani Mohan, E.K. Park, Yijie Han
University of Missouri, Kansas City
{srmhv7 | ekpark | hanyij}@umkc.edu

Abstract

Proxy web caching is commonly implemented to decrease web access latency, internet bandwidth costs and origin web server load. We propose a transparent shareable proxy caching methodology in which the proxy caches maintain a continuously optimal performance with a significant improvement in the cache hit ratio without requiring any additional overhead at the client or at the routers. This approach allows caches to be personalized based on the users’ web access patterns. Our approach employs four light weight intelligent agents in a system of proxy caches which use classification algorithms, data mining techniques and performance monitors to study the client’s web access patterns and configure caches to best serve the clients with available resources. These agents run with predictable run times with an optimal use of computer resources.

1. Introduction

A system of shareable proxy web caches and a web cache server are responsible for caching different types of web objects. The proxy caches, located at the backbone routers switch incoming requests from clients (using switch lists), either randomly or more recently based on data mining techniques such as URL (Universal Resource Locator) text mining, web content mining, etc. These techniques add additional overhead to the caching protocols, clients and/or routers.

Our approach has been developed with the following design considerations in mind:

a. Use an existing caching architecture so that ISP’s do not need to change the existing costly deployment of proxy caches which is scalable to larger networks.
b. There should be no re-configurations necessary for either the clients or the routers (transparency)
c. No constraints on physical location of proxy caches

2. Caching Architecture

A distributed caching architecture is used to implement our personalized caching approach. Figure 1 gives the system deployment for our approach. The web cache server (Master cache) does not cache anything and does not have any clients attached to it. It manages the system of proxy web caches. The web cache server however has a temporary store of pre-processed log data received from individual proxy caches, the switch list that it maintains and re-configuration information for the proxy caches. Proxy caches manned by the same Master cache are shareable and initially configured using default settings with clients being randomly assigned to a proxy cache. The log data pre-processing agent and cache performance agent are deployed at the proxy web caches. Most of the processing occurs in the system of proxy web caches and web cache server.

Cache communication is via simple IP multicasts. All caches (1…k) work by transparent inter-proxy cooperation. All client requests are routed through the proxy caches transparent to the clients. The client requests are sent to the appropriate proxies through interception at the backbone routers based on their IP addresses. This client to proxy cache switch list is updated and sent to the routers by the classifying agent.

3. Multi-Agent System

The Multi-agent system comprises of automatic agents that run continuously and semi-automatic agents which only run when another agent triggers them. An overall working of the various agents is shown in the algorithm in Figure 2 and a brief description is given below:

3.1. Automatic agents

3.1.1. Log data pre-processing agent. Every proxy cache logs client requests along with its service status (whether the requested Web object was available in the cache: Cache_HIT, Cache_MISS, Cache_REFRESH_HIT, whether the requested web object was denied though it
was present in the cache, etc.). The agent continuously filters the data for required information and rearranges data on a per client basis.

![Diagram](image)

Figure 1 Multi-agent deployment in proxy cache set up

### 3.1.2. Cache performance agent.

The performance of a cache is measured by its ability to serve a client’s request as efficiently as the origin web server. The Cache performance agent is deployed in the proxy caches where the performance is measured for every cache cycle.

For every pre-determined cache cycle, various performance parameters such as the throughput, mean response time, connection length, cache run-time hit ratio, etc. are monitored by studying the cache logs. The cache run-time hit ratio (%) measures the ratio of the amount of requests served from the cache, as a percentage of total successful requests serviced but only for the most recent (about 10,000) requests serviced (unlike the cache hit ratio). This will give a more accurate estimate since the cache configuration is changing dynamically.

A steady decline in the cache performance (random fluctuations in performance are ignored/filtered as they are usually not the cause for poor cache performance) triggers the classifying agent to modify the re-assignment of clients to caches based on updated web access patterns. However, if this still does not improve performance then the cache maintenance agent is called to re-configure the proxy caches based on new trends identified by it.

### 3.2. Semi-automatic agents

#### 3.2.1. Classifying agent.

This agent is triggered by the cache performance agent to assign clients to a proxy cache based on the results obtained from the cache performance agent. The switch list is updated at the backbone router to handle future client requests.

#### 3.2.2. Cache maintenance agent.

This agent is triggered by the cache performance agent and has access to the pre-processed proxy cache web logs. The purpose of this agent is to identify the clients’ frequent web access patterns so that the cache can be re-configured with the rules which satisfy the unique feature sets (itemsets) of the clients.

We use association rules to mine for web access patterns of users using a variation of the Apriori algorithm [1] to identify frequent sequence patterns (itemsets) from the pre-processed cache logs in linear time [2] [3].

![Algorithm](image)

Figure 2 Multi Agent system algorithm

A unique Itemset for a client can be defined as the features which define the client’s web object requests such as average size of downloaded files, number of requests made per unit time, etc. Itemsets comprise of items chosen from a set of pre-defined items (cache configuration parameters) each of which has a support value to determine whether or not to include it to the client’s itemset.

These patterns have been selected for optimal use of resources and increased cache performance. This architecture can give optimal cache performance for different client types with available resources.

### 5. References

