USING ONTOLOGIES TO IMPROVE PERFORMANCE IN A WEB SYSTEM

A Web Caching System Case of Study

Carlos Guerrero, Carlos Juiz and Ramon Puigjaner
Departament de Matematiques i Informatica, Universitat de les Illes Balears
Crta. Valldemossa km 7.5, Palma de Mallorca, Spain

Keywords: Web performance, ontologies, semantic web, web cache.

Abstract: This paper gives the details of a web system architecture which uses ontologies to improve the behavior of the system from a performance point of view. The architecture presented is implemented in a cache level. As web system performance depends on states and parameters fixed in runtime period, the configuration of our system will be changed during that period. We need to monitor the system and store those gathered information. A knowledge base is used to store those information and measures. We propose the use of reasoners that use that gather information in the K.B. to change the configuration and setup of the system during runtime period. That configuration also is modelled by a knowledge base which use ontological languages.

1 INTRODUCTION

In Web pages on-the-fly building process a higher number of activities or process take part than in the static web sites. Therefore, the dynamic workload and the computational cost are higher than the workload generated by a static web server. This is the reason why research topics to improve the performance of web systems are getting more importance (Menasce and Almeida, 2001; Menasce and Virgilio, 2000; Killelea, 2002).

In web dynamic environments the process to deliver contents to users incurs: 1) user requests are dispatched to appropriate software modules that service these requests, thereby producing network overhead; 2) these software modules determine which data to fetch and present, thereby producing process overhead; 3) the disk I/O management querying the backend database produces data overhead, and finally; 4) the assembled data needs to be formatted and delivered to the browser, thereby producing cache overhead.

The aim of the research work presented is to create a tool that changes the web system setup and configuration using the user and server performance and behavior information. Recent research works present studies about the way ambient environment respond dynamically using the information about the state of the system or environment (Lera et al., 2006; Lera et al., 2007).

In those research works, ontological languages are used. Data about the state of the environment and the performance of the system is stored by instances of an ontology. The same idea is applied in the research work presented in this paper. The presented framework is a system where ontologies are used to managed the information about the state of the system (Web System Elements Knowledge Base) and the behavior of the user and the servers tiers (Behavior Knowledge Base). A reasoner analyzes the information provided by the Behavior K.B. and applies operations to the instances of the different Web System Elements K.B.

At the beginning of this paper we present the proposal of the system architecture. An example of the use of the Behavior and Performance K.B. applied to web caching tier is explained. Last section is a summary of the model domain of the ontology used at the Behavior and Performance Knownledge Base is presented.

2 GLOBAL ARCHITECTURE

In this paper is presented the partial work of a global system. The proposed global system is designed to store and analyze the information and data about the behavior and performance of the different elements in

Guerrero C., Juiz C. and Puigjaner R. (2008).
USING ONTOLOGIES TO IMPROVE PERFORMANCE IN A WEB SYSTEM - A Web Caching System Case of Study.
In *Proceedings of the Fourth International Conference on Web Information Systems and Technologies*, pages 117-122
DOI: 10.5220/0001518801170122

the web systems using ontologies and semantic web.

In Figures 1 and 2 we present a first approximation to the proposed architecture:

- *Behavior Knowledge Base:* This is the centralized module that uses an ontology to store information about all the elements in the system: users, requests, proxies, caches nodes, gateways, web servers, database systems, etc. In (Guerrero et al., 2008) is explained the design of the ontology to model performance and behaviour in a web system. In Section 4 a summary is also presented.
- *Web system elements Knowledge Base*: Chances over the configuration of the system are needed. Therefore the way the system works and its setup has to be modelled. To model that situation, each part or element of the system will have associated a knowledge base with that information. We need a specific ontological language for each part of the system because each part has its own characteristic and specific model domain. Once the different elements in the system are modelled with the correspondent ontology, those elements will use those models to change the way they work.
- *Reasoner*: It uses the information about the behavior of the clients, server tiers and server resources to determine the best tuning of the web system. It uses heuristics applying rules and operations to try changes in the setup of the different elements in the system. The operations the reasoners apply are chosen analyzing the parameters in the Behavior K.B. The operations are different for each element of the web system. For example in (Guerrero et al., 2007) are defined the operations for a cache system where the documents are fragmented to improve the performance. If we actuate over others elements of the web system, those operations are not valid and a new set of operations will be defined.

In the first approximation presented in this paper only monitoring functions are implemented. The main objective to reach as future work is to have a complete monitoring and actuating architecture. We can add reasoners to the architecture that use the information in the Behavior K.B. to change the organization or the way the system works.

We propose that the behavior of each module or element in the system will be modelled by an ontology in a knowledge base. Each of those ontologies has the information how the system works, for example: load balancing information, caches policies, task priorities, etc. Therefore the reasoners use the information in the Behavior K.B. and changes the different elements K.B. applying rules and operations (Fig.2).

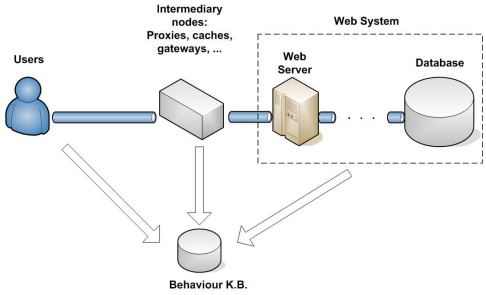

Figure 1: Behavior Knowledge Base Architecture.

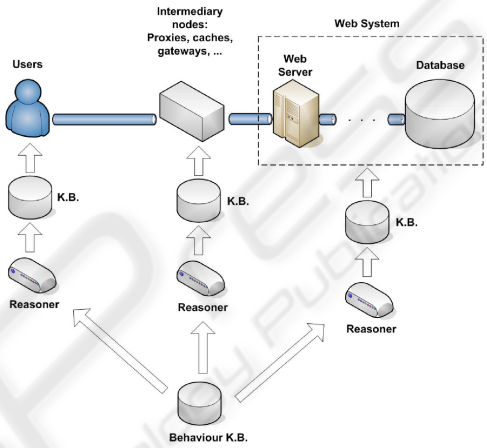

Figure 2: Reasoners and Actuators Architecture.

3 WEB CACHING SYSTEM

The first step to achieve the complete system presented is to create a concept probe tool. As we have experience in the past working with cache systems (Guerrero et al., 2007), this is the tier chosen to create the tool. So we propose an scenario to improve the web caching techniques. That scenario need information about the behavior and the performance of the different elements in the system. With that information the system would cache the web pages in different ways, trying to get a better performance in the whole system. The ontology presented in the Section 4 has been used to model the behavior and the performance.

Typically web-based applications are based in tiered architecture. Usually in these applications the user interface, functional process logic (”business rules”), data storage and data access are developed and maintained as independent as is possible, most often on separate platforms.

Dynamic web sites commonly are developed using different tiers to reduce the cost of maintenance and the idealistic layered application is three-tier websites:

- A front end Web server serving static content
- A middle dynamic content processing and generation level Application server.
- A back end Database, comprising both data sets and the Database management system or DBMS software that manages and provides access to the data.

As the number of tiers and interfaces between tiers increase, the generation process is also bigger and computationally more expensive. To minimized the workload generated by each user request response, a cache tier could be placed between clients and server layers. Web caching also contributes to reduce bandwidth usage, server load, and user perceived lag. The caching tier stores generated content to avoid the load over the server tiers at next request of the same page.

Traditional Web Cache Tier performs the tasks of listen users requests, analyze if those requests are store locally in the cache, in which case, the local copy is returned or, if the copy is not present, send the request to the server and updated the local version with the server response (Wills and Mikhailov, 1999).

There are a lot of techniques to improve the web caching efficiency, but the Web Cache Tier needs to gather information about the behavior of the web system and the users. It's important to know the user profiles: pages most requested, navigation paths, thinking times, etc. From the server side is important to know the response time and the size of each request, the changing ratio of the web pages contents and, finally, the sharing ratio between web pages contents. Those parameters are the most interesting to achieve the fragment design with the best performance.

In a first step we have implement a Web Cache Tier which gather information about the system and store it in an Knowledge Base which use the ontology defined at Section 4. As a future work we will develop the reasoner and actuator that use that gathered information to change the way fragments are cached.

3.1 System Description

The architecture of the system is an usual tiered web system architecture: (a) server side; (b) cache tier; (c) client side. The server side its a three layer application (Figure 3). We decided install a generic web system. We installed WordPress (Automatic, 2007), thus web logs are wide extended over Internet, and it is once of the type of system with a higher update ratios.

WordPress is structured in three layers: (a) MySQL as the database system; (b) PHP is the scripting language; (c) use of templates for the presentation and design of the HTML document.

Our main objective is to study the cache system. We need to introduce some monitors to gather information in that layer of the system. In one hand we need an open source tool. In the other hand the most extended ontological tools are programmed with Java. Therefore we need a open source Java application, to avoid the implementation of the whole tier. We have chosen Smart Cache (Kolar, 2007). Smart Cache is HTTP 1.1 and HTTPS proxy cache written in pure Java 1.1.

The cache tier (Smart Cache tool) has been modified to gather information about the requests that serves to the clients, requests to the origin servers, measures of the performance, etc. All that gathered information is use to create the model in the Ontological Language presented at Section 4. We use Jastor (Ben Szekely, 2007) to integrate the Ontological Language and Java. Jastor is an automatic generator of Java interfaces, implementations, factories and listeners based on the properties and classes hierarchies in the Ontological Language.

Finally, we use Apache JMeter to emulate the users (Apache, 2007). JMeter is a Java desktop application designed to load test functional behavior and measure performance. It may be used to test performance both on static and dynamic resources. It can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types. It can be used to make a graphical analysis of performance or to test your server/script/object behavior under heavy concurrent load.

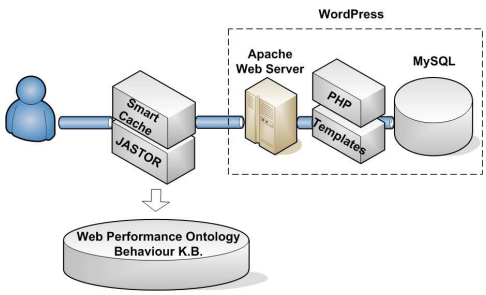

Figure 3: Concept Prove System Description.

3.2 Validation

The concept tool presented is created only to prove the availability of the use of ontological languages to gather and store information about the behaviour of a web system. So the probes to validate this fact will be consist on execute a synthetic load on the web system and compare the logs of the different web elements monitored with the information stored in the

Behaviour Knowledge Base.

We run the system with a synthetic user load to evaluate all the system. The synthetic load correspond to one hundred concurrently users requesting random pages to the server. The system has one thousand pages generated randomly. After runtime period of one million requests, we need to validate the behavior and performance information of the different elements in the system with the values observed and gathered by the cache system and the behavior knowledge base.

In that first experiment we only gather information about the individual user requests: number or request, response times and response sizes. Both in the user side and in the origin server side we have logs about the activity in the system. Apache provides modules to create web logs in a standardized text file format. Common Logfile Format (CLF) is the most usual format. CLF, for each request received by the server, add a new line in the text file with the next format:

```
host ident authuser date request status bytes
10.0.0.1 - mathew [12/Jul/2004:14:50:13 -0700]
   "GET /index.php HTTP/1.0" 200 3465
```

We can extract the number of requests and the size of that requests. In the other side, with JMeter we can extract a huge number of measures and statistics, in particular, response sizes, response times and requests number.

Once we have analyzed the Apache log and the JMeter log we can compare those measures with the measures gathered by the cache level in the knownledge base. We observed that the mean value of the number of requests over each web page is one thousand. This values are the same in the cache log and in the Behaviour K.B.

For the response time analysis we could not use the Apache log because all the pages stored in the cache tier are not requested to the server, so these request are not registered in the Apache log. As the cache tier log is not implemented to measure the response time of the served requests only JMeter could be used to gather that information. Once we have compared the resquest time registered by JMeter and stored in the Behaviour K.B. we observed that the means time are practically the same, with a small overhead over the JMeter measures. The reason of that difference is the communication time between the cache tier and the JMeter.

After the analysis of the measures we can conclude the Ontology defined in Section 4 can be used to model the behavior and performance of web systems. Our next future work is to implement some reasoning rules to analysis that data and take decisions over the configuration of the system.

4 WEB PERFORMANCE AND BEHAVIOR ONTOLOGY

In this section is presented a summary of the Web Performance and Behavior Ontology described in (Guerrero et al., 2008). Ontologies development is needed of the use of some methodology. In our case, we use the ontology building life-cycle explained in (Davies et al., 2002; Uschold, 1995) and used in other research works as (Lera et al., 2006; Lera et al., 2007).

The elements which take importance at our scenario are: user sessions, user requests, HTTP requests, HTTP responses and performance metrics. Web applications and systems are built over HTTP protocol (Internet based application protocol). Therefore the definition of our Ontology, which is used to represent the performance information, has to be determined by the definition of HTTP (Fielding et al., 1999).

For users, architecture of the server tiers are completely clear and they make requests to an URI (Berners-Lee et al., 2005). Users do not worry about if that URI corresponse to an isolated server, a proxy server, a cache server, load balancing server. When the HTTP request arrive to the server identified in the URI, the server processes that request. This process could be divided in two different types of process: local tasks and remote tasks.

The local tasks, that a server makes to give response to user requests, generate a local workload on that tier of the web-system. We could identified different kind of local workload in the different elements or components of the server: disks, processors, DB systems, memory, scripting interpreters modules, web server modules. Some of that elements or components could need tasks associated to other tiers in the web-system. In those cases, new HTTP requests are generated between the different system tiers. These new requests generate network workload and local workload in the target layer (remote tasks). Transmission times, latency and node process are the elements corresponding to the network workload (Baldi et al., 2003). In Figure 4 we present the model domain of concepts corresponding to workload.

When local task in a tier is done, a HTTP response is generated. If that HTTP response arrives to another tier, once all the responses will arrive and all the local tasks will be done, another HTTP response will be generated. That path is repeated right to the tier that received the user request. The HTTP response generated by that last tier goes directly to the user. HTTP responses generate network workload, in the same way HTTP requests do. Figure 5 shows a simple model domain which corresponses to the HTTP

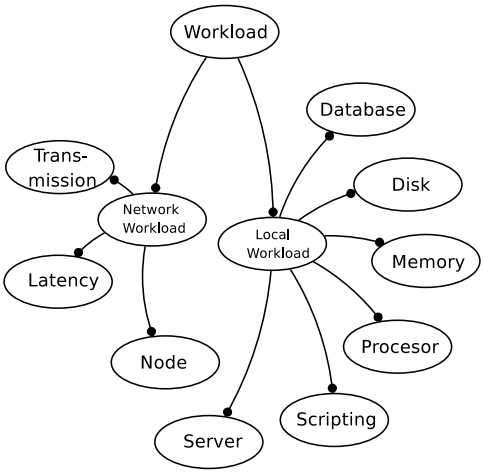

Figure 4: The abbreviate Workload model domain.

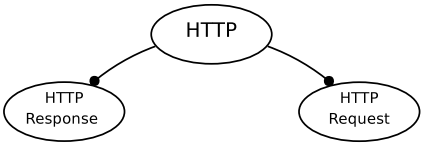

Figure 5: The abbreviate HTTP model domain.

protocol concepts.

The main measures to represent performance evaluation in a web system are not the same for the different elements in the system. In the model domain, each of those measures corresponse to a subclass of the class representing the performance evaluation. In Figure 6 we see those subclasses. For each of those classes a different measure system is used. In that point, we took advantage of one of the most useful features of OWL, knowledge sharing, and we have imported other important ontologies and their well-defined semantic knowledge to our model domain.

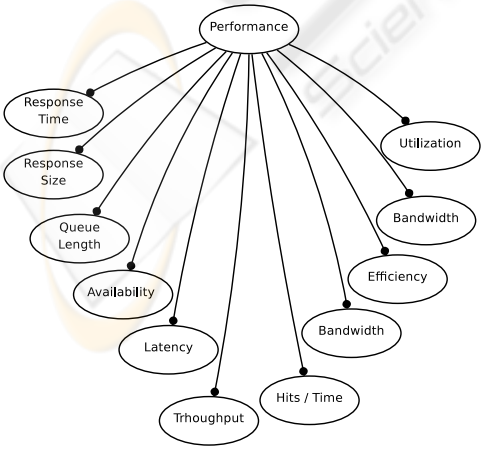

Figure 6: The abbreviate Performance Evaluation model domain.

The relationships between the classes of the model are represented in Figure 7. Ontological models represent relationships as object properties.

Users usually interact with the system (User session), generating a continuous flow of requests as result of the response information from the system (Figure 8). Each time a user generate a request a HTTP request is associated to demand the server a service or data. The server answer with a HTTP response. Each HTTP message (response or request) generates workload over the network and over the different tiers in the web system.

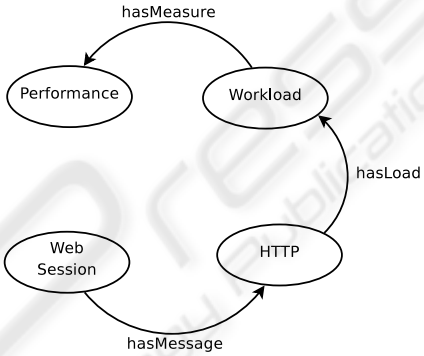

Figure 7: Object properties between classes.

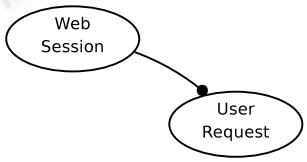

Figure 8: The abbreviate Web Session model domain.

5 CONCLUSIONS

An architecture to improve web performance has been proposed. That architecture is based on the idea of improve web performance changing the configuration and setup of the system during runtime period. Those chances are made at runtime period because the parameters that condition the performance and behavior of the system are not known in the development phase and could change along time.

An ontological system is proposed to analysis the behavior of the system and make changes to the fragments trying to improve performance. The model domain of an ontology to reach that aim has been presented. A knowledge base to store information about the behavior and performance of the system has been create using that ontology.

A case of study based on testing and monitoring the information in the web cache tier has been presented. Smartcache has been used as cache system, and has been modified to gather performance information. Jastor has been used to integrate the ontological language in the cache system. Therefore, the cache tier has been available to gather information from the system and store those information in the K.B.

In a validation phase the information stored in the K.B. and the information store in the traditional logs of the different tools of the web system have been compared. That comparison results in a positive way and validates the process of gathering information and the use of the ontological language. The next step is to add reasoning process to the system to take decision about chances in the setup of the web cache system.

Future work is open to two different branches. One branch opens research work to create the specific ontological language to model the different elements in the web system and the integration of that ontologies in the different elements that would use the store information in the correspondent K.B.

A second branch concerns to the definition of the operations and rules that changes the models of each element in the web system using the information in the performance and behavior knowledge base.

ACKNOWLEDGEMENTS

This work is partially financed by the Spanish Ministry of Education and Science through TIN2007-29683-E project.

REFERENCES

Apache (2007). Apache jmeter. http://jakarta.apache.org/jmeter/.

Automatic (2007). Wordpress 2.0. http://wordpress.com/.

Baldi, P., Frasconi, P., and Smyth, P. (2003). *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*. Wiley.

Ben Szekely, Rob Gonzalez, J. B. (2007). Jastor. http://jastor.sourceforge.net/.

Berners-Lee, T., Fielding, R., and Masinter, L. (2005). Uniform resource identifier (uri): Generic syntax. RFC 3986, Internet Engineering Task Force.

Davies, J., van Harmelen, F., and Fensel, D., editors (2002). *Towards the Semantic Web: Ontology-driven Knowledge Management*. John Wiley & Sons, Inc., New York, NY, USA.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Berners-Lee, T. (1999). Hypertext transfer protocol – http/1.1. RFC 2616, Internet Engineering Task Force.

Guerrero, C., Juiz, C., and Puigjaner, R. (2007). The applicability of balanced esi for web caching. In *Proceedings of the 3rd International Conference on Web Information Systems and Technologies*.

Guerrero, C., Juiz, C., and Puigjaner, R. (2008). Web performance and behavior ontology. In *Proceedings of the Second International Conference on Complex, Intelligent and Software Intensive Systems*.

Killelea, P. (2002). *Web Performance Tuning*. O'Reilly & Associates, Inc., Sebastopol, CA, USA.

Kolar, R. (2007). Smart cache 0.93. http://scache.sourceforge.net/.

Lera, I., Juiz, C., and Puigjaner, R. (2006). Performance-related ontologies and semantic web applications for on-line performance assessment intelligent systems. *Sci. Comput. Program.*, 61(1):27–37.

Lera, I., Sancho, P. P., Juiz, C., Puigjaner, R., Zottl, J., and Haring, G. (2007). Performance assessment of intelligent distributed systems through software performance ontology engineering (spoe). *Software Quality Control*, 15(1):53–67.

Menasce, D. A. and Almeida, V. (2001). *Capacity Planning for Web Services: metrics, models, and methods*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Menasce, D. A. and Virgilio, A. F. A. (2000). *Scaling for E Business: Technologies, Models, Performance, and Capacity Planning*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Uschold, M. (1995). Towards a methodology for building ontologies.

Wills, C. E. and Mikhailov, M. (1999). Examining the cacheability of user-requested Web resources. In *Proceedings of the 4th International Web Caching Workshop*.