Policy-based autonomous bidding for overload management in eCommerce websites

TONI MORENO[1,2], NICOLAS POGGI[3], JOSEP LLUIS BERRAL[3], RICARD GAVALDÀ[4], JORDI TORRES [2,3]

[1] Department of Management, U. Politècnica de Catalunya, Av. Diagonal, 647 08028 Barcelona, Spain. e-mail: toni.moreno@upc.edu

[2] Barcelona Supercomputing Center, Barcelona, Spain

[3] Computer Architecture Department, U. Politècnica de Catalunya, Barcelona, Spain

[4] Department of Software, U. Politècnica de Catalunya, Barcelona, Spain

Abstract: In eCommerce applications with heterogeneous traffic, which typically run on execution environments where resources are shared with other applications, being able to dynamically adapt to the application workload in real time is crucial for proper self-management and business efficiency. The AUGURES platform has recently introduced a novel approach to deal with server overload situations, based on the prioritization of sessions according to the expected revenue that the session is likely to generate. However, by denying access to exceeding low priority users, the website may be loosing potential customers, while there might be other resources available in the market of the execution environment, be it a server, a server farm, or a grid. This paper presents an extension of the AUGURES architecture with a simple policy-based autonomous bidding agent that generates automated bids for the extra resources needed to execute the transactions that have been refused by the server due to overload.

Keywords: autonomous bidding, utility computing, overload management, admission control, autonomic computing

1 Introduction

During the recent years there have been important changes in the way eCommerce websites work. First, there has been a shift from originally serving mainly static content to fully dynamic websites. Second, the typical execution environments are evolving from a scenario where each company used to maintain their own servers –or at least they had dedicated servers hosted by an external service provider- to a shared hosting scenario. In shared hosting, applications even from different companies share resources which are allocated dynamically. On the other hand, while the traditional bottleneck of the web has traditionally been network bandwidth, dynamic applications place a huge demand on CPU power. Not only web programming languages have become more complex over the years, but also user requirements. Websites now rely on technologies such as AJAX to make pages more

interactive, XML-based web services to exchange B2B information among companies, and SSL for security. While these technologies improve the user experience and privacy, they also increase the demand for CPU power. Under these resource-intensive conditions an efficient management of CPU time is crucial, since it can dramatically reduce the cost of ownership. An attractive solution is to make the system components able to manage themselves (Cheliotis and Kenyon, 2003). This can be solved using proposals from the Autonomic Computing research area (Kephart and Chess, 2003; Kephart 2005) that draws in an enormous diversity of fields within and beyond the boundaries of traditional research in computer science. For example, autonomic computing is one of the latest application areas for machine learning (Rish et al, 2006).

The AUGURES project presents an autonomic computing approach, based in the self-management of limited resources, seeking the maximization of revenues obtained from sales in an eCommerce application. Poggi et al (2007a) consider the case where the eCommerce application has a fixed and limited amount of CPU resources available. They apply machine learning techniques to analyse the probability of purchase of each of the sessions and use these probabilities to prioritize the sessions with higher purchase probability in case of overload. The basics of the AUGURES system and the most relevant results obtained so far are summarized in Section 2. Section 3 proposes an extension of the functionalities of the AUGURES system by allowing to dynamically acquire extra resources if the currently available static resources are fully committed; e.g. in case of system overload caused by increased number of users. The bids to increase resource capacity with extra resources would be autonomously performed based on policies according to expected benefit of the transactions. Finally, section 4 presents some conclusions and future work.

2 The AUGURES System

2.1 The architecture

The AUGURES architecture has an offline component which takes a historical user access log file from the eCommerce application and produces a model (the predictor), that is used in real time to estimate the probability of purchase for each user session. The real-time component gets the incoming requests, runs them through the predictor, and outputs the priority along with other static information- for the session, which is then used by the firewall's session manager to enforce the required actions according the company's rules and current load on the system. These two subsystems are presented graphically in Figure 1.

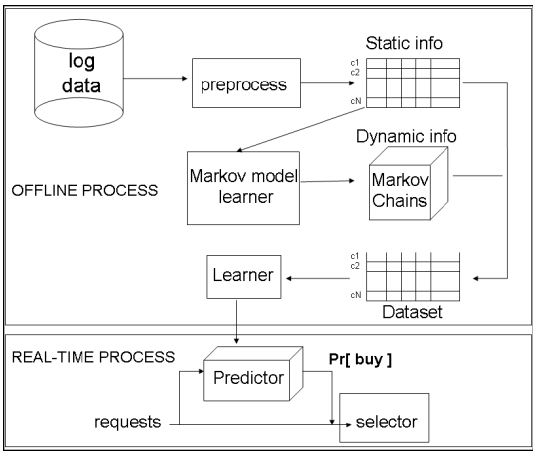

Figure 1. The AUGURES System

The log file used by the offline module is produced by the dynamic application of the company site. It contains non ambiguous session and page actions (tags) as historical data, and is first cleaned and reorganized by a preprocessor. The preprocessor produces an intermediate dataset with one line for each transaction. These lines are largely computed independently from each other, so they do not contain information about the user navigation pattern; that is why we call the information in this dataset static. Next, the dataset is enriched with dynamic information reflecting the user navigation sequence, relating the different transactions of the same section. This is done by computing Markov models of both buying and non-buying sessions; the prediction of these models for each individual request is added as extra information to each line of the preprocessor. Finally, this enriched dataset is passed to a learning module that produces a predictor, some mathematical function that, given a request, assigns it a buying probability. See Poggi et al. (2007b) for more details.

2.2 Some results

The architecture described above proves useful when the system is working in overload conditions. The percentage of buying transactions in the original dataset is 6.6 %. If the AUGURES system is used to prioritize the transactions and only the top priority transactions are kept, the percentage of buying transactions in the filtered dataset significantly increases. Figure 2 shows the precision (understood as the percentage of purchasing requests among the admitted ones) vs the percentage of admitted transactions. For example, we see that if we can only serve the 10% of the incoming requests, selecting them at random would lead to a 6.6% of purchasing requests, while if we use AUGURES to prioritise the percentage of purchasing requests would increase to almost 40%.

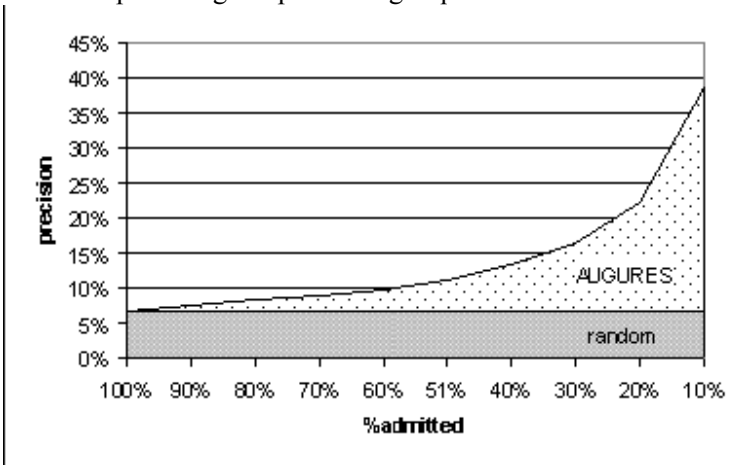

Figure 2. Precision vs %admitted

3 Policy based autonomous bidding in AUGURES

The AUGURES architecture does its best to maximize the throughput in economic terms given the amount of resources that are permanently allocated to the application (that is, with a static allocation), but does not do anything to leverage the existing extra resources in the market (dynamic allocation). Since many applications share resources in a server farm with several other applications, and those resources can be used almost instantaneously (Guitart et al, 2005), it seems reasonable to think about deviating to the extra dynamic resources some of those requests that have not been served by the static server, because AUGURES had given them a low priority. AUGURES ranks the sessions

according to their purchase probability and if the static server is able to process K sessions simultaneously, it keeps the top K sessions. This does not mean that the dismissed sessions are not interesting. If the system had enough resources to serve them, they would be served and some of them would end up making purchases and providing some revenue. It can be the case that it is worth paying for extra dynamic resources in order not to loose this revenue.

Markets seem to be useful to solve resource allocation problems (J. Shneidman et al, 2005). We propose to use automated bidding for resource trading (Neumann, 2006) in this scenario. A policy-based autonomous bidding system should decide the bids for the extra resources required to serve all or some of the requests that have not been served by the static server. The bid should take into account the expected profit that the request is likely to give. The current AUGURES prototype does not estimate the expected profit of a request, but only provides an estimation of the purchase probability. As an approximation, the average profit per purchase can be used, although in future versions of the prototype a more precise estimation based in the probability distribution will be given.

A bid consists of the request of a number of CPUs during a certain time in exchange of a certain amount of money. Both the number of CPUs and the time needed are values that we assume that can be calculated, and we concentrate on the price. The price assigned to a bid to execute a request using extra resources will be determined by the expected value of the requests and the policy defined by the user. We propose to use a very simple policy to define maximum bid prices taking into account the risk aversion of the decision maker:

Bid price $BP = \alpha \cdot EP$

where EP the expected profit of the transaction and a α value close to 1, lower if the decision maker is risk-averse and higher if he is risk-prone.

If the market is offering the necessary resources to execute a request at a price that is lower or equal to BP, then the request not served by the static server is executed using extra resources.

Overload management in eCommerce websites is an interesting area of application for policy-based autonomous bidding techniques, since decisions are taken in timescales too short for humans to personally take the decisions, and the definition of a policy that is autonomously implemented seems to be a natural choice. However, the concepts in this field are sometimes difficult to understand, since they imply some kind of knowledge about technical details of web servers and web applications. But policies for resource overload are not restricted for web traffic management. There are a number of more general situations where the techniques studied in this work could be applied. An example of another application field would be overbooking management. Most air carriers do overbook their existing capacity, i.e. they sell more tickets for a flight than the number of passengers they can actually carry, trying to increase their benefits accounting for the no-shows. They must often face the situation of having more passengers showing for a flight than the number of available seats. In this case, the airline typically offers compensation to the passengers that are not accepted to the flight, who in some cases volunteer to postpone their trip in exchange of the compensation. If the number of passengers that volunteer to get the compensation is not enough to free all the necessary seats to accommodate all the passengers, the airline can choose whom to prioritize and whom to deny access according to some rules incorporating the existing knowledge about passengers and their expected value, analogously to the presented approach for the prioritization of online requests suggested by the AUGURES approach. Similarly, they might want to buy the necessary additional capacity out in the external market, when doing so is less expensive (considering both economic and image costs) than paying the compensations and postponing the overbooked passengers' trips.

4 Conclusions and future work

The policy-based autonomous bidding seems to be an interesting strategy to leverage the external resources in a CPU market to deal with overloads. Several aspects have to be improved in order to be able to define a policy that can be used in a real case. The quantification of the expected revenue of a transaction is crucial to define the bids, and it should be estimated in the future versions of the AUGURES system; currently only the probability of purchase is estimated, not the magnitude of the transaction. The AUGURES project is still ongoing, and it is expected that the ideas suggested in this paper are incorporated to a newer prototype in order to test their applicability in real cases.

Acknowledgements

This work is supported by the Ministry of Science and Technology of Spain and the European Union under contract TIN2004-07739-C02-01. R. Gavaldà is partially supported by the 6th Framework Program of EU through the integrated project DELIS (#001907), by the EU PASCAL Network of Excellence, IST-2002-506778, and by the DGICYT MOISES-BAR project, TIN2005-08832-C03-03. For additional information about the authors, visit the Barcelona eDragon Research Group web site at http://research.ac.upc.edu/eDragon

References

G. Cheliotis, C. Kenyon (2003). *Autonomic economics: a blueprint for selfmanaged systems.* IJCAI workshop on AI and autonomic computing: developing a research agenda for self-managing computer systems.

J. Guitart, D. Carrera, V. Beltran, J. Torres, E. Ayguadé (2005). *Session-Based Adaptive Overload Control for Secure Dynamic Web Applications.* Proc. 34th International Conference on Parallel Processing (ICPP'05), 341–349.

J. Kephart (2005). *Research Challenges of Autonomic Computing.*Proc. 27th International Conference on Software Engineering, 15–22.

J. Kephart, D. Chess (2003). *The Vision of Autonomic Computing.* IEEE Computer, january 2003..

D. Neumann, S. Lamparter, B. Schnizler (2006).*Automated Bidding for Trading Grid Services.* European Conference on Information Systems (ECIS).

N.Poggi, T. Moreno, J. Berral, R. Gavalda, J. Torres (2007a). *Web customer modeling for automated session prioritization on high traffic sites.* 11th International Conference on User Modeling, 2007 (accepted)

N.Poggi, T. Moreno, J. Berral, R. Gavalda, J. Torres (2007b*).AUGURES: Self-Adaptive Web Customer Prioritization for Admission Control.* Technical Report, UPC, 2007.

I. Rish, G. Tesauro, R. Das, Jeff Kephart (2006). *ECML/PKDD-2006 workshop on Autonomic Computing: A New Challenge for Machine Learning*.

J. Shneidman et al (2005). *Why Markets Could (But Don't Currently) Solve Resource Allocation Problems in Systems*. 10th Workshop on Hot Topics in Operating Systems, Santa Fe.