Bayesian Methods to Estimate Future Load in Web Farms

José M. Peña[1], Víctor Robles[1], Óscar Marbán[2], and María S. Pérez[1]

[1] DATSI, Universidad Politécnica de Madrid, Madrid, Spain
{jmpena, vrobles, mperez}@fi.upm.es
[2] DLSIS, Universidad Politécnica de Madrid, Madrid, Spain
omarban@fi.upm.es

Abstract. Web Farms are clustered systems designed to provide high availability and high performance web services. A web farm is a group of replicated HTTP servers that reply web requests forwarded by a single point of access to the service. To deal with this task the point of access executes a load balancing algorithm to distribute web request among the group of servers. The present algorithms provides a short-term dynamic configuration for this operation, but some corrective actions (granting different session priorities or distributed WAN forwarding) cannot be achieved without a long-term estimation of the future web load. On this paper we propose a method to forecast web service work load. Our approach also includes an innovative segmentation method for the web pages using EDAs (estimation of distribution algorithms) and the application of semi-naïve Bayes classifiers to predict future web load several minutes before. All our analysis has been performed using real data from a world-wide academic portal.

Keywords. Web farms, web load estimation, naïve Bayes, EDAs

1 Introduction

Today's commerce is, in many cases, fought on the web arena. Form marketing to online sales, many tasks are supported by a web architecture. Even several internal procedures of a company are achieved using a web-based application. On this scenario, the quality of the services, or at least the way the user observes how these services are been performed, depends on some characteristics of the web server itself. Subjective values, like page design, are tackled by specific methodologies [16]. But, there are other factors that are also very important in terms of client satisfaction when he/she navigates the site. One of these factors is *web service response time*. There are different strategies to speed-up this response time. Proxy servers and web caches are intermediate storages of the most frequent pages. Another innovative alternative is what has been called **web farms**.

1.1 Web Farms

For high performance services, when it grows beyond the capabilities of a single machine, groups of machines are often employed to provide the service. In the case of web

J. Favela et al. (Eds.): AWIC 2004, LNAI 3034, pp. 217–226, 2004.

servers, this is often referred to a Web Farm. Web farms typically employ both high availability and scalability technologies in order to provide a highly available service spread across multiple machines, with a single point of contact for clients.

The architecture of these systems is divided into three different layers: (i) a first layer, the front-end of the web service, is a single point of access, (ii) a second layer of distributed web servers, which perform the actual HTTP service and (iii) a third layer of a shared data storage, like a back-end database or distributed/replicated file system.

The service provided by a web far is performed as follows. When a request is received by the point of access, this one redirects the request towards the most suitable web server on the second layer. Usually, the response sent by the web server is addressed directly to the client (not using the point of access as a relay).

There are different types of web farms depending on the topology of the architecture itself:

- **Cluster web farms**: All the web servers are inside of the same local area network. In many cases these machines serving on the second layer are connected using a virtual private network (VPN). On this architecture, the point of access selects the most appropriate server depending on short-term load-balancing algorithm. [2]
- **Globally distributed web farms**: In this cases, web servers are connected at different points on a wide area network (like the Internet). On these geographically distributed web farms the point of access decides the server taking into account other aspects like closest web server or interconnection speed. [9]
- **Hybrid approaches**: Sometimes both architectures coexist, a first step of redirection forwards the request to the closest cluster web farm. A second stage is performed inside of the cluster to select the server with less work. A very good example of this approach is Google[3].

1.2 Web Load Balancing

In order to achieve the most important task of this process, *request forwarding*, there have been done many efforts to perform this operation in the most flexible and efficient way. NCSA [19] provides a scalable web server using server-side Round-Robin DNS [6]. LVS (Linux Virtual Server) [3] provides many different methods to redirect request inside of a cluster web farm, like NAT (Network Address Translation), IP Tunneling and Direct Routing. There are also application level solutions, like Reverse-proxy [25] or SWEB [2]. Other techniques available is Load Sharing with NAT (LSNAT) [27].

Although, these are different techniques to forward client requests (at different levels of the IP protocol stack), another open issues is to get the best selection for each of these incoming requests. Cluster web farms and many DNS-based approaches select the next web server are Round-Robin scheduling with additional features like, weighted least-connection correction [30] (also Round-robin DNS). Other approaches select the most responsive web server for a *ping* package. Other approaches consider QoS (quality of service) algorithms to estimate load distribution in replicated web servers [9]. In [14] a client-based approach is presented.

[3] http://www.google.com

There are two possible load balancing strategies, depending on how far the status of the system could be forecasted.

- Immediate load balancing: Load balancing decisions take into account present or very near past situations to perform corrective and balancing actions. On this schema cluster web farms forwards messages towards the least overloaded server.
- Long-term load balancing: Although the past strategy is quite efficient and provides dynamic adaptability, there are important limitations due to: (1) a single session from the same user should be forwarded to the same web farm server, since important information held by the server, like session cookies, are stored locally, and (2) high level forwarding technique (like IP Tunneling) performs better if the tunnel that supports the redirection is used several times, or other techniques like DNS-Round Robin once the first message from a user is addressed to one server the next ones will follow the same path.
  Since under these circunstances the load balancing algorithms take some actions with a stable application frame of several minutes, the most efficient way to perform these tasks is to estimate the future load of the system, with a time window frame of at least 10-15 minutes.

2 Future Load Estimation Issues

Dealing with future estimation of a web service load has some specific aspect to be taken into account.

First of all is to know what is exactly expected from the estimation. In order to support load balancing algorithms the most important aspect to be known is the future number of requests or the amount of information to be sent. Being able to predict how many bytes should be served in advance provides the possibility to forward clients to other mirror site or discard some requests, in order to provide an acceptable response time for most of the user.

The amount of information requested is a continuous value, but for the objective we are tackling is not necessary to deal with the attribute in the continuous form. A load balancing algorithm uses discrete data input, for most of the algorithms they work with intervals of low, medium or high load. On the other hand, a significant number of classification methods are based on discrete or nominal classes. There are several methods to build these discretization intervals, but, in order to be appropriate with the usage the values are hardware or architecture restrictions from the current system, like network band-width or memory capacity.

Other important issue is to select the appropriate input attributes. The main source of information we have is the server web log. This log records all the connections requested to the web server. For each record we have the requested page, from which page it was accessed, and when it was requested. Very simple methods use the time of the day to predict web load, while others also consider previous load (amount of bytes transferred) to estimate future requests. These approaches (as we will see in section 5.3 are limited and unaccurated). It is very important to have some background knowledge from the application field of these techniques. On our approach an academic portal[4]

[4] http://www.universia.es

was used. This portal stores web pages from courses, universities, grants, an so on. The information on this service is addressed both to students, teachers, and people working on the industry. Pages represent news and important notes updated daily. A detailed description of these datasets in commented in section 5.1. These characteristics imply two important hints:

- The patterns from different users or the accesses to different pages on the web site is more intense depending on the contents of the pages. Page subject has an important influence on when this pages is accessed and the range of users interested on it.
- The structure of the web site changes daily. Any page-based analysis using data form several days before will not consider new pages (which are also the most popular visit in the web site).

One important aspect of this method is the segmentation of the pages hosted by the web site, among the different techniques (based on the contents of the page) we have proceeded with this segmentation considering the relationships between the pages (how many times one pages is addressed from other) as discriminant criteria to group similar pages.

According to these assertions we have considered the following method to proceed, (1) the attribute to classify is a discretized value defined over the amount of requested information, (2) accesses to the groups of pages (in a previous time frames) are input data, and (3) only information for few days before the prediction is used in order to support dynamic changes on the contents of the web site.

This method will be compared with simple approaches that only takes into account the time an previous web load.

3 General Process Overview

To be able to estimate the future load in a Web Farm we need to do the following steps (see figure 1):

1. Starting with the log files from *www.universia.es* of four different days (see section 5.1) we create a *Site Navigation Graph*. In this graph, vertices represent Web pages, edges represent links and edges weight represent the number of times the corresponding link was clicked by the users.
2. Partition the *Site Navigation Graph*. We have decided to split the graph in ten different partitions. In this way we will be able to collect users click information for each partition, given us more useful information. This process segmentates web pages based on the relationships among them. Pages visited together will be clustered in the same group.
3. Once data has been partitioned, time windows of 15 minutes have been defined. For each of these windows a new instance is created with the following information:
   - Current time.
   - Clicks and bytes transferred for each of the groups during the las two time frames (15 and 30 minutes before).
   - The amount of information requested to the web site during this period. This is the actual value to be predicted.

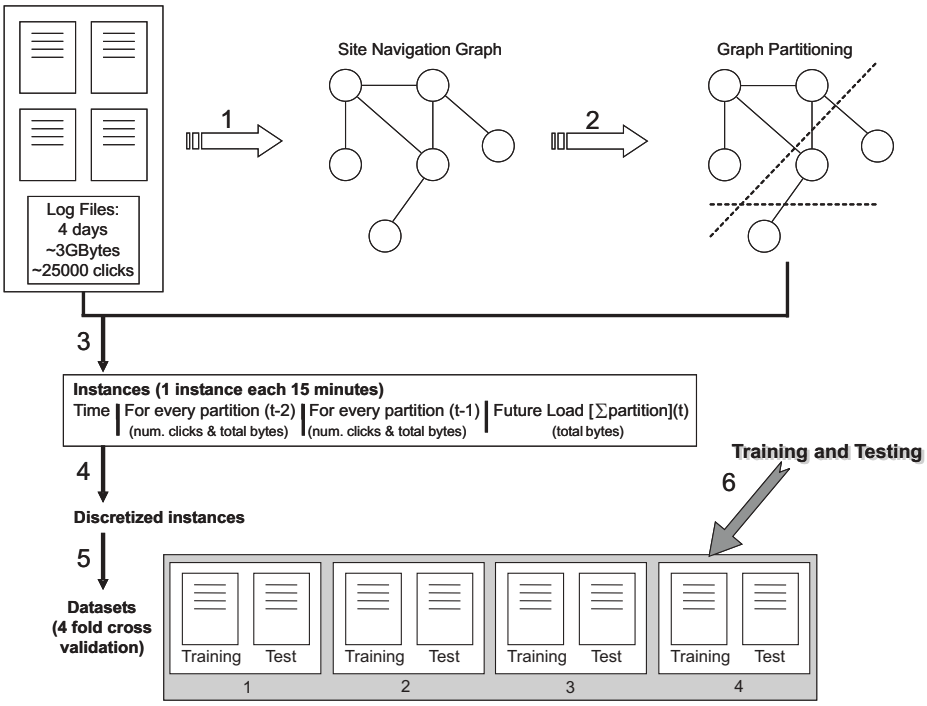

Fig. 1. Overall description of the process

4. Discretization:
   - Class discretization: Rather than the number of bytes, we are interested on the intervals, like idle state, low load, medium load, high load, very high load or similar ranks. As we have said before, these intervals depend on hardware constraints. A discretization method based on probability density with optimal classes definition has been performed, using XLSTAT.
   - Attribute discretization. In order to use the classification methods it is necessary to discretice the instances of the datasets. Thus, a discretization step with the method suggested by [11] has been performed using MLC++ tools. This supervised discretization method is described by Ting in [28] that is a global variant of the method of Fayyad and Irani [12].
5. Once we have created and discretized the instances, they must be splitted in four different datasets, each one including a training set and a testing set. In this way, we are going to make the training with the instances of 3 days, keeping the instances of the last day for validation purposes. This means that we are doing a *4-Fold cross validation*.
6. The last step consist on apply all the classification methods (see section 4.2) to the datasets. Experiment results are in section 5.3.

4 Detailed Experimentation

4.1 Graph Partitioning

The partitioning problem on an undirected graph $G = (V, E)$, V being the set of vertices and E the set of edges arises in many different areas such as VLSI design, test pattern generation, data-mining or efficient storage of data bases. In general, the graph partitioning problem consists of partitioning the vertices into k disjoint subsets of about the same cardinality, such that the *cut size*, that is, the sum of edges whose endpoints are in different subsets, is minimized.

The graph partitioning problem is NP-hard, and it remains NP-hard even when k is equal to 2 or when some unbalancing is allowed [7]. For large graphs (with more than 100 vertices), heuristics algorithms which find suboptimal solutions are the only viable option. Proposed strategies can be classified in combinatorial approaches [13], based on geometric representations [26], multilevel schemes [29], evolutionary optimization [5] and genetic algorithms [8]. We can find also hybrid schemes [4] that combines different approaches.

To solve the graph partitioning problem in this paper, we have used a novel approach based on a heuristic optimization technique named EDA. EDAs [20] are non-deterministic, stochastic heuristic search strategies that form part of the evolutionary computation approaches, where number of solutions or individuals are created every generation, evolving once and again until a satisfactory solution is achieved. In brief, the characteristic that differentiates most EDAs from other evolutionary search strategies such as GAs is that the evolution from a generation to the next one is done by estimating the probability distribution of the fittest individuals, and afterwards by sampling the induced model. This avoids the use of crossing or mutation operators, and the number of parameters that EDAs require is considerably reduced.

To find the best partition we are going to use the next fitness function [5] to evaluate each candidate s,

$$f(s) = \alpha \cdot n_{cuts}(s) + \beta \cdot \sum_{k=1}^{K} 2^{deviation(k)} \tag{1}$$

where, $n_{cuts}(s)$ is the sum of the edges whose endpoints are in different subsets, and $deviation(k)$ is the amount by which the number of nodes in the partition G_k varies from the average number expected.

However, with the objective of partitioning the *Site Navigation Graph* we are going to consider the number of cuts as an objective, while the deviation degree is going to be a restriction. Thus, the fitness function will take the parameters $\alpha = 1$ and $\beta = 0$; with $deviation(k) < 20$ for $k \in [1, K]$. This means that we are allowing a deviation degree of 20% in the partitions.

4.2 Learning Methods

We are using three different types of classification methods.

- **Rule induction methods**.
  - *1R Algorithm*. The 1R procedure [17] for machine learning is a very simple one that proves surprisingly effective. The aim is to infer a rule that predicts the class given the value of the attributes. The 1R algorithm chooses the most informative single attribute and bases the rule on this attribute alone.
  - *NNGE*. NNGE (Non-Nested Generalized Examplers) [21] is a promising machine learning approach that combines nearest neighbor with rule generation.
- **Tree induction methods**.
  - *J48*. J48 is an improved version of C4.5 [23] that is implemented in the Weka toolset [1], an open source machine learning software in Java.
  - *C5.0 with boosting*. C5.0 is quite similar to C4.5 [23] but incorporates several new facilities such as variable misclassification costs, new data types including dates and facilities for defining new attributes as functions of other attributes.
- **Bayesian methods**.
  - *Naïve Bayes*. The naïve Bayes classifier [15] is a probabilistic method for classification. It can be used to determine the probability that an example belongs to a class given the values of the predictor variables. The naïve Bayes classifier guarantees optimal induction given a set of explicit assumptions [10].
  - *Pazzani-EDA*. Pazzani [22] tries to improve the naïve Bayes classifier by searching for dependencies among attributes. He proposes two algorithms for detecting dependencies among attributes: *Forward Sequential Selection and Joining* (FSSJ) and *Backward Sequential Elimination and Joining*. However, Pazzani-EDA [24] makes a heuristic search of the Pazzani structure with the target of maximize the percentage of successful predictions.

The rule induction methods and the tree induction method J48 were run with Weka, an open source machine learning software in Java. The method C5.0 with boosting were run with the See5.0/C5.0 program. For the Bayesian methods we have used our own developed programs.

5 Results

5.1 Data Set

For validating our approach we have used log files from *www.universia.es*. Universia is a community portal about Spanish-speaking higher education that involves 379 universities world-wide. It contains information about education, organized in different groups depending on the user profile.

We have used the log files from four different days. The total size of the logs files is 3 GBytes and they contain a total of 25000 user clicks in approx 5000 different pages.

5.2 Validation Method

As commented before we have used a 4 fold cross validation method. Thus, the learning process is made with 3 days, validating with the remaining day.

Table 1. Experiment results with the simple dataset

Training Days	*1, 2, 3*	*1, 2, 4*	*1, 3, 4*	*2, 3, 4*	
Tested Day	*4*	*3*	*2*	*1*	*Average*
Naive Bayes	70.21	75.00	79.17	79.16	**75.88**
Pazzani-EDA	84.78	89.36	93.61	87.23	88.75
NNGE	64.89	63.54	71.88	71.87	68.05
1R	71.28	72.92	70.83	76.04	72.77
J48	64.89	72.92	73.96	76.04	71.95
C5.0	74.50	72.90	83.30	80.20	**77.72**

Table 2. Experiment results with the enriched dataset

Training Days	*1, 2, 3*	*1, 2, 4*	*1, 3, 4*	*2, 3, 4*	
Tested Day	*4*	*3*	*2*	*1*	*Average*
Naive Bayes	71.28	70.83	76.04	70.83	72.25
Pazzani-EDA	97.82	100.00	97.87	91.49	**96.80**
NNGE	74.47	67.71	78.13	72.91	**73.31**
1R	67.02	73.96	75.00	76.04	**73.01**
J48	72.34	69.79	72.91	76.04	**72.77**
C5.0	70.20	70.83	73.96	75.00	72.50

5.3 Experimental Results

One of the issues tackled by our approach is to evaluate any possible improvement obtained when data are enriched with the segmentation the web pages. On table 1, results based on a simple data input are presented. These simple data are just the number of clicks an bytes requested during the last two time windows (15 and 30 minutes before).

Table 2 shows the results from enriched data including number of clicks and bytes transferred for each of groups of pages clustered by the partitioned graph. On this table we also consider the previous two time windows as in the case before.

We have experimented with simple 1R algorithm with the assumption that just one attribute (either time or the previous requested clicks or bytes) is able to predict web service load.

Induction tree algorithms, like C5.0, performs poorly with the extended input data while complex bayesian classifiers, like Pazzani-EDA, get a significant advantage when more and enriched information is provided. On this case Pazzani-EDA clearly outperforms any other of the classifiers evaluated.

Another interesting result is that 1R selects time as discriminant attribute only in two out of the four experiments. With this we consider that time is not always the best approach to estimate web server load.

6 Conclusion and Further Work

In this paper two innovative techniques have been introduced.

First, a new graph partitioning method has been perform using estimation of distribution algorithms. On this field new open issues arises that could be interesting to explore in a further works.

Second, enriched information has been proved very useful to obtain significant improvements to estimate web service load. The segmentation of the pages using the relationships among them, represented by the number of navigation clicks from one page to the other (site navigation graph) is a very simple way to cluster similar pages without the need to deal with semantic information about the contents of each page. Other similar approaches use information about the subjects or topics related to one page make them difficult to be performed on complex an not well-organized web sites.

Although time is not a discriminant attribute on this kind of analysis, it is very significant. Thus, semi naïve Bayes algorithms like NB-Tree [18] could achieve interesting results as this classifiers is an hybrid method between decision trees and Bayesian classification methods. This approach uses the more discriminant attributes to build a decision tree with leaves that are Bayesian classifiers for solving the more complex relationships.

7 Acknowledgement

We would like to thanks *www.universia.es* for the possibility of working with their log files.

References

[1] Weka 3: Data mining with open source machine learning software in java. http://www.cs.waikato.ac.nz/ml/weka/, 2003.

[2] Daniel Andresen, Tao Yang, and Oscar H. Ibarra. Towards a scalable distributed WWW server on workstation clusters. In *Proc. of 10th IEEE Intl. Symp. Of Parallel Processing (IPPS'96)*, pages 850–856, 1996.

[3] Wensong Zhang andShiyao Jin and Quanyuan Wu. Creating Linux virtual servers. In *LinuxExpo 1999 Conference*, 1999.

[4] R. Baños, C. Gil, J. Ortega, and F.G. Montoya. Multilevel heuristic algorithm for graph partitioning. In *Proceedings of the 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization. LNCS 2611*, pages 143–153, 2003.

[5] R. Baños, C. Gil, J. Ortega, and F.G. Montoya. Partición de grafos mediante optimización evolutiva paralela. In *Proceedings de las XIV Jornadas de Paralelismo*, pages 245–250, 2003.

[6] T. Brisco. RFC 1794: DNS support for load balancing, April 1995. Status: INFORMATIONAL.

[7] T.N. Bui and C. Jones. Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters*, 42:153–159, 1992.

[8] T.N. Bui and B. Moon. Genetic algorithms and graph partitioning. *IEEE Transactions on Computers*, 45(7):841–855, 1996.

[9] Marco Conti, Enrico Gregori, and Fabio Panzieri. Load distribution among replicated Web servers: A QoS-based approach. In *Proceedings of the Workshop on Internet Server Performance (WISP99)*, 1999.

[10] P. Domingos and M. Pazzani. Beyond independence: conditions for the optimality of the simple Bayesian classifier. In *Proceedings of the 13th International Conference on Machine Learning*, pages 105–112, 1996.
[11] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the 12th International Conference on Machine Learning*, pages 194–202, 1995.
[12] U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Conference on Artificial Intelligence*, pages 1022–1027, 1993.
[13] C. Fiduccia and R. Mattheyses. A linear time heuristic for improving network partitions. In *Proceedings of the 19th IEEE Design Automation Conference*, pages 175–181, 1982.
[14] Vittorio Ghini, Fabio Panzieri, and Marco Roccetti. Client-centered load distribution: A mechanism for constructing responsive web services. In *HICSS*, 2001.
[15] D.J. Hand and K. Yu. Idiot's Bayes - not so stupid after all? *International Statistical Review*, 69(3):385–398, 2001.
[16] Esther Hochsztain, Socorro Millán, and Ernestina Menasalvas. A granular approach for analyzing the degree of affability of a web site. *Lecture Notes in Computer Science*, 2475:479–486, 2002.
[17] R.C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11:63–90, 1993.
[18] R. Kohavi. Scaling up the accuracy of naive-Bayes classifiers: a decision-tree hybrid. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining*, pages 202–207, 1996.
[19] Thomas T. Kwan, Robert E. McGrath, and Daniel A. Reed. NCSA's World Wide Web server: Design and performance. *IEEE Computer*, pages 68–74, November 1995.
[20] P. Larrañaga and J.A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publisher, 2002.
[21] B. Martin. Instance-based learning: Nearest neigbour with generalisation. working paper series 95/18 computer science. Technical report, Hamilton, University of Waikato.
[22] M. Pazzani. Constructive induction of Cartesian product attributes. *Information, Statistics and Induction in Science*, pages 66–77, 1996.
[23] R. Quinlan. *C4.5 Programs for Machine Learning*. Morgan Kauffman, 1993.
[24] V. Robles, P. Larrañaga, J.M. Peña, E. Menasalvas, M.S. Pérez, and V. Herves. Learning semi naïve Bayes structures by estimation of distribution algorithms. In *Lecture Notes in Computer Science (LNAI)*, volume 2902, pages 244–258, 2003.
[25] Ralf S.Engelschall. Load balancing your web site: Practical approaches for distributing HTTP traffic. *Web Techniques Magazine*, 3(5), 1998.
[26] H.D. Simon and S. Teng. How good is recursive bisection? *SIAM Journal of Scientific Computing*, 18(5):1436–1445, 1997.
[27] P. Srisuresh and D. Gan. RFC 2391: Load sharing using IP network address translation (LSNAT), August 1998. Status: INFORMATIONAL.
[28] K.M. Ting. Discretization of continuous-valued attributes and instance-based learning. Technical Report 491, University of Sydney, 1994.
[29] C. Walshaw and M. Cross. Mesh partitioning: a multilevel balancing and refinement algorithm. *SIAM Journal of Science Computation*, 22(1):63–80, 2000.
[30] Wensong Zhang. Linux virtual server for scalable network services. In *Ottawa Linux Symposium*, 2000.