

# Resource Placement in Distributed e-Commerce Servers

Gustavo M. C. Gama Wagner Meira Jr. Márcio L. B. Carvalho  
Dorgival O. Guedes Virgílio A. F. Almeida

e-SPEED Laboratory - Computer Science Department  
Universidade Federal de Minas Gerais  
Antônio Carlos Av., 6627, Belo Horizonte, Brazil

**Abstract** – e-Commerce services have become a promising and profitable application of the Internet. In order to keep them growing, solutions must be found to deal with unreliable connections and high latencies, among other problems. The best solutions to such problems tend to depend on the distribution of the service over the network, placing servers in multiple locations, closer to customers. If placement of servers is effective it tends to reduce delays and traffic-related costs. In this paper we discuss the distribution of e-commerce services by introducing a traffic-aware cost model and evaluating it using an actual log from an e-tailer. The results show that the model yields good placement solutions, which perform better than simpler ad-hoc solutions.

## I INTRODUCTION

e-Commerce services have become a promising and profitable application of the Internet. Nevertheless, the use of the Internet as the communication environment in this case is not without its problems. For an e-commerce site to be successful it has to reduce the negative effects of unreliable connections and high network latencies, among other problems. Such problems may compromise customer satisfaction and therefore the success of the virtual business.

Techniques based on improving the quality of a centralized server do not provide a reliable solution, since the difficulties faced are inherent to the network infrastructure itself, not just the server. A better solution tends to be the distribution of the service over the network, placing servers in multiple locations closer to the final users. The existence of multiple servers tends to increase availability, and assuming the placement is well planned, it tends to reduce delays. Proxy cache servers [8] are an example of a successful service distribution strategy: they replicate static content such as HTML pages and images, and are responsible for significant traffic reduction. Content distribution networks [3] have deserved special attention lately, since they provide a more reliable and controlled solution than proxy caches, but at a cost for the information providers. On the other hand, there are several applications that generate dynamic and non-cacheable responses (at least when using traditional cache technolo-

gies).

For a distributed e-commerce solution to be successful, there are several issues that must be addressed, such as resource placement and discovery, and load balancing. Resource placement policies define the number and location of servers and products. A larger number of servers allows a higher degree of parallelism, but also leads to higher complexity and context maintenance costs. Regarding product allocation in servers, it would be simpler to divide them uniformly, but a strategy that takes regional demands into account should be more cost-effective. Resource discovery is defined by strategies that determine how clients decide which server to contact. These strategies may be based on network topology or geographical affinity, and may be implemented through dynamic DNS replies, for example. Load balancing is a common issue in Web-based applications which are characterized by dynamic behavior.

In the context of e-commerce servers, resource placement may be seen as a data distribution problem. Several issues concerning data distribution have been extensively studied under different contexts. Database management systems (DBMS), in particular, employ a wide range of algorithms, techniques, and strategies that have been analyzed in the past [1]. However, many of the strategies proposed cannot be applied at all or must be adapted to be used in the Web. In this new environment, there have been some recent work that presents promising results. In [5], the authors discuss the existing algorithms and propose a new method for resource discovery in distributed applications. In [11], a locality-aware distribution scheme for clusters is implemented, achieving high cache hit ratios and good load balancing.

In this work we focus on the resource placement issue by presenting and evaluating a traffic-aware model. Our approach is to design a model that will use real data gathered from user behavior in an e-commerce application to decide where, in the Internet, servers should be placed and how products should be distributed among them in order to reduce overall network traffic.

When considering the impact of service requests on a large internetwork as a whole, traffic reductions can be directly associated with lower latencies and costs, and there-

fore higher customer satisfaction.

Our proposal is to study server and resource placement in terms of network clouds, just considering in which cloud a server would be installed. Those clouds may be Autonomous Systems (AS) in the Internet, for example. This approach tries to account for the fact that in most cases one has little control over the internal structure, routing paths and organization of a network to choose exactly where to place a server. It also has the effect of simplifying our models to a reasonable level, since we do not have to consider the fine grained structure of the Internet, but just its organization in terms of larger units.

If we consider costs, a server/product placement strategy that reduces overall traffic is also a good idea. Internet billing is currently strongly related to the capacity of the links the organization uses to connect to its neighbors, and this capacity is dictated by the amount of traffic that must travel between them. In this sense, each time a given data packet crosses a boundary between different network clouds (belonging to different organizations) it contributes to the link utilization. In summary, reducing the link utilization will also decrease connectivity costs.

Our approach to model traffic may be classified as a path-based model, where traffic is modeled by the source-sink pairs in contrast with a traditional approach to study the traffic distribution by modeling the data that goes through each individual link. The link-based approach leads to a much more complex and hard to solve model.

In the sections that follow we discuss e-commerce services and propose a model to evaluate their distribution (Sections II and III). In Section IV we illustrate the application of the proposed model by distributing services of an actual e-tailer server and compare it to two simpler strategies. The last section presents the conclusions and discusses some future work.

## II DISTRIBUTED E-COMMERCE SERVICES

e-Commerce services are usually provided using standard mechanisms of the WWW, that is, clients request services through the HTTP protocol and wait for responses from the server. In terms of server implementation, however, e-Commerce servers differ from traditional WWW servers by demanding three additional functionalities: transactional support, state maintenance, and persistent and reliable storage. Transactional support assures that the services are performed correctly, also keeping the data valid and consistent, for example, avoiding race conditions; state maintenance allows the server to store volatile data about the interaction of the user with the site, while reliable storage (usually provided by DBMS systems that run on the server machines) maintain long term information.

We distinguish, without loss of generality, three types of data that may be accessed while servicing e-commerce

requests: (1) product data<sup>1</sup>, which includes inventory and other dynamic information, (2) customer data, which includes his or her profile, and (3) session state, which records customer interactions and partial transactions..

Notice that some services may affect instances of all types of data. For instance, whenever an e-tailer customer add a product to his or her basket, the added product inventory is decremented, the customer profile is updated, and the state information (in this case the basket) is also augmented. In current centralized server architectures the accesses and modifications are performed locally by the same DBMS system and usually serially. On the other hand, maintaining all these data coherent in a distributed implementation is not a simple task, although it may be performed by distributed database systems (usually at a high cost, however).

We may potentially distribute all types of data, but usually it is not an efficient strategy. Consider a scenario where there is a set of servers geographically distributed that provide services to customers on behalf of an e-tailer. Further, assume that there is an algorithm that assigns a customer to the best server for him or her, in terms of network latency and bandwidth (i.e. each customer will be assigned to a single server). State is usually associated with a customer, so state data may also be stored in the same server. On the other hand, products are naturally “shared” among customers (i.e., concurrently accessed) that may be far apart and do not behave identically. In this context, it makes sense to distribute and keep coherent just product data.

Therefore, based on the customer behavior, we may reduce the general problem of distributing all data handled by an e-commerce server by considering just product distribution.

## III RESOURCE PLACEMENT STRATEGY

In this section we discuss our strategy for distributing e-commerce services. Our goal, as mentioned, is to reduce the overall traffic associated with requests to the set of distributed servers.

We consider two types of costs while distributing services: traffic and processing costs. Thus, in order to install a server in a given network cloud, we have to assess not only the installation and transaction processing costs for a server in that cloud, but also how much this server would reduce the overall traffic. Global traffic costs are a function of the number of requests and distance (in terms of links traversed) between clients and a server. We assume that a client is aware of the location of the products, requesting them to the nearest server that will be able to satisfy the request; requests within the cloud where the

---

<sup>1</sup>we use the term product also to designate the services that are commercialized by an e-commerce server.

server is located are not charged. The processing costs are divided into a fixed cost for installing the server in a cloud and a per-transaction cost that is a function of the inventory of products managed by the server.

Given  $N$  network clouds and  $P$  products, customers located at a cloud  $i$  are responsible for a demand of  $d_{ik}$  items of product  $k$ , where  $1 \leq i \leq N$  and  $1 \leq k \leq P$ . Our problem is to determine the number of products of the type  $k$ ,  $p_{ik}$  that are managed by a server located in cloud  $i$  in order to minimize the total cost. We use  $r_{ijk}$  to denote the number of requests for product  $k$  originated at cloud  $i$  that are serviced by a server located at cloud  $j$ . So,  $\forall d_{ik}, d_{ik} = \sum_{j=1}^N r_{ijk}$  for each  $k$  (the demand at cloud  $i$  for product  $k$  is the sum of the requests originated at this cloud and responded by any server) and  $\forall p_{jk}, p_{jk} = \sum_{i=1}^N r_{ijk}$  for each  $k$  (the total amount of a product  $k$  stored at cloud  $j$  is the sum of all requests answered by this cloud).

The overall distribution cost becomes  $C = \sum_{k=1}^P \sum_{j=1}^N \sum_{i=1}^N (c_{ij} * r_{ijk} + \alpha_j * r_{ijk}) + \beta_j * s_j$ , where  $1 \leq i, j \leq N$  and  $1 \leq k \leq P$ ,  $c_{ij}$  is the transmission cost of a product request from cloud  $i$  to cloud  $j$ ,  $\alpha_j$  is the cost of processing one single transaction by a server in cloud  $j$  (all products have the same processing cost),  $\beta_j$  is the cost of installing a server in cloud  $j$ , and  $s_j$  is a binary variable that indicates whether there is server installed in cloud  $j$  ( $s_j = 1$  iff  $p_j > 0$ ).

The model naturally handles the fact that not all clouds should have a server, since  $r_{ijk}$  may be null indicating that there are no requests for  $k$  from cloud  $i$  to cloud  $j$ . Further, as we discuss in Section IV, it allows us to verify the trade-offs between transaction and traffic-related costs. Although we do not prove it here due to space limitations, there is strong evidence [7] that this problem is NP-complete.

We should note that this model determines the best distribution for a given stream of requests. As we know, requests may vary over time and therefore the solution obtained by this model may result in non-optimal assignments. In this case, we employ a load balancing algorithm that uses the proposed model as a balancing function, as it will be discussed in Section IV.

#### IV CASE STUDY: E-TAILER

In this section we discuss the service distribution of an actual e-tailer. We start by characterizing its workload, and then we evaluate the effectiveness of our model to solve the resource placement issue in e-Commerce service distribution.

##### A. Workload Characterization

In this section, we present the workload characterization data for an e-tailer. We focus on the *allocate* and *pur-*

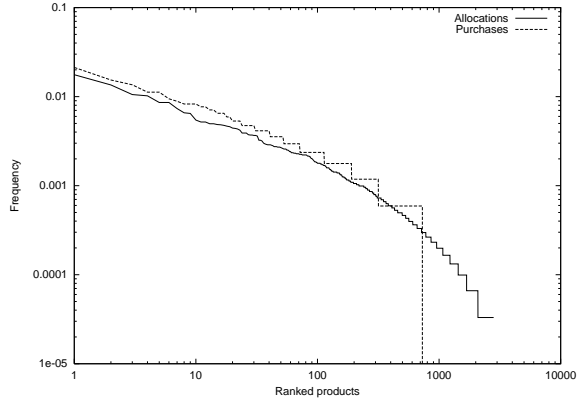


Figure 1: Frequency distribution of allocations and purchases ranked by the respective function

*chase* functions, which require transactional support [6] and will benefit from our distribution model. A more complete characterization of e-business workloads may be found in [9, 2].

The starting point of our characterization is an access log of a virtual bookstore. The log was collected on August of 1999 and covers a 30-day period, subdivided in two 15 days traces. The first 15 days – which we will refer as “log trace #1” from now on – were used to apply the model described in section III and evaluate the efficiency of our solution in terms of the server placement issue. The last 15 days – “log trace #2” – were used to validate and evaluate the resource distribution efficacy of our model. Log trace #1 was composed by 1,056,040 requests, of which 30,275 (2.86% of the requests) were allocation requests and 747 were purchase requests (users buy in 0.7% of their visits), resulting in an allocate-to-purchase ratio of 2.46%<sup>2</sup>. Log trace #2 was composed by 1,408,706 requests, being 35,238 (2.50% of the requests) allocation requests and 861 purchase requests (allocate-to-purchase ratio of 2.44%).

We start our characterization by verifying the popularity of products in terms of allocations and purchases. In Figure 1 we can see the frequency distribution of allocations and purchases of products ranked according to the frequency of each operation, that is, the leftmost points are for the products most allocated and most purchased, which are not necessarily the same product. We can observe that Zipf Law applies quite strongly to both curves. Zipf’s law[12] was originally applied to the relationship between a word’s popularity in terms of rank and its frequency of use. It states that if one ranks the popularity of words used in a given text (denoted by  $\rho$ ) by their frequency of use (denoted by  $P$ ) then  $P \sim 1/\rho$ . The high variance in terms of product allocation impacts the distribution strategy by creating very popular products, which

<sup>2</sup>The small number of purchase-related functions may be explained by a significant number of pricebots that request information to the site.

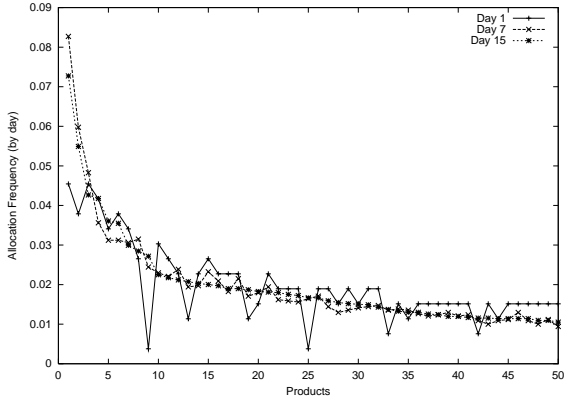


Figure 2: Temporal variation of the allocation frequency per product

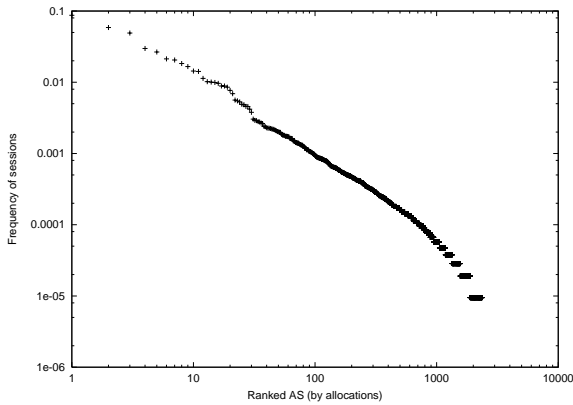


Figure 3: Frequency distribution of the number sessions originated by each AS

are requested by all customers, demanding an efficient distribution strategy.

Another interesting question is how the popularity of products for each function varies across time. In the graph of Figure 2, we plot the allocation frequency of products in three time scales: 1 days, 7 days, and 15 days. We choose the 15-day time scale to rank the products of the other two time scales. By observing the variance of the allocation frequencies in the 1-day and 7-day time scale, we can see that there is a clear demand for a load balancing strategy that adapts to the workload variations observed. The model proposed in this paper may be used as a load balancing function, as we show next. Another interesting observation is regarding how the product popularity varies over time. As discussed in [4], there is a high variance in terms of product demand under varying time-scales, which may reach a factor of 2 in a 15-day period. This variance poses a clear demand for a load balancing strategy, which may employ the model proposed in this paper.

We then characterize the number of operations performed by network clouds. In our case and in the re-

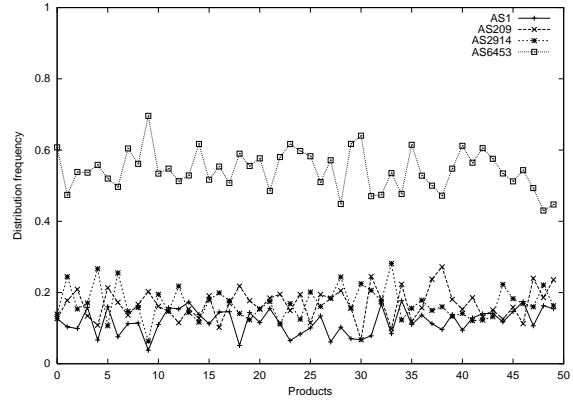


Figure 4: Distribution frequency for the 50 most popular products in log trace #1

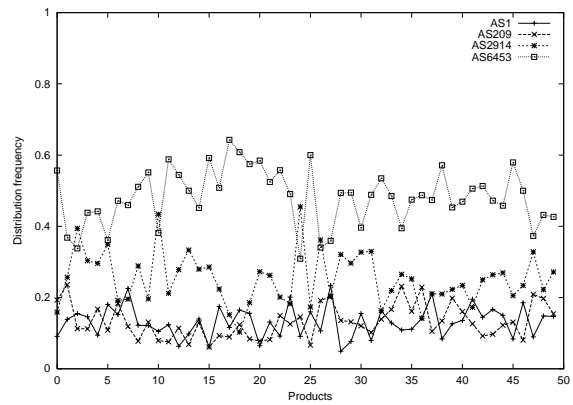


Figure 5: Distribution frequency for the 50 most popular products in log trace #2

sults presented in the next section, each cloud is an Autonomous System (AS). An AS is a geo-referenced entity that is the basis of Internet routing mechanisms. An AS usually comprises several classes of IP addresses, simplifying their administration and routing. We associated the IP requests to their ASs using the data provided by the *Router Arbiter Project*[10], which stores the routing tables of all ASs and also allows, through a WHOIS interface, IP addresses to be associated to their respective AS. In Figure 3, we can observe the frequency distribution of the allocations as a function of their originating AS. The distribution again seems to follow a power-law, indicating a very high variance among the number of requests originated by each AS.

Last, we characterize the variance of product allocations in terms of resource distribution. Using the distribution output from the model applied to the first log trace, we plotted the distribution frequency variance for the 50 most popular products considering trace #1 (Figure 4) and trace #2 (Figure 5). Our first observation is regarding the lack of a large variance in product preference across network clouds, especially for log trace #1. This is a strong

Time scale	Total requests	Allocation misses	Redirections			
			Round-Robin	Uniform/Product	Weighted/Server	Our Model
1 day	7788	937 (12.0%)	6062 (77.8%)	3076 (39.5%)	2577 (33.1%)	2586 (33.2%)
3 days	8056	1198 (14.9%)	6275 (78.0%)	2782 (34.5%)	2183 (27.1%)	2177 (27.0%)
7 days	6180	669 (10.8%)	4782 (77.4%)	1496 (24.2%)	851 (13.7%)	936 (15.1%)
15 days	7480	631 ( 8.4%)	5894 (78.8%)	2681 (35.8%)	2057 (27.5%)	2110 (28.2%)

Table 1: Model evaluation results

indication that product distribution can sometimes be approximated by the total distribution of allocations. However, the second log trace shows that this is not a constant and product preference might vary widely across mirrors, ranging from 46% to 7% in AS2914, for instance.

In summary, the behavior of customers is very dynamic and hard to predict both in terms of product popularity and access pattern across time. Our model, as shown in the next section, handles this dynamics by allowing flexible product distributions, and also is a powerful tool for evaluating load balancing strategies and their parameters.

### B. Applying the Proposed Model

We now discuss how our model may be applied to a distributed e-tailer and how to evaluate it based on the data presented in the previous section. We start by solving the model for log trace #1, demonstrating its applicability. We then use the resource placement parameters for evaluating the model accuracy on another time period, namely trace #2.

As mentioned previously, we use the AS structure as network clouds in our model. We estimate the local costs ( $\alpha$  and  $\beta$ ) of each cloud as a linear function of the connectivity of the respective AS. Our cost differentiation is based on the premise that a more connected cloud would charge more to host a distributed server. We then considered two scenarios to evaluate the model’s distribution solution: first, the centralized strategy, where there’s just one server (the e-tailer server itself) responsible for all requests; second, the distributed solution of our model. Further, we varied  $\alpha$  and  $\beta$  in order to evaluate the trade-offs between costs associated with traffic and local storage/transaction processing.

A complete description and analysis of these trade-offs is presented in [4]. In summary, solving the model showed to be extremely expensive, computationally speaking, but the execution of relaxed versions of the model<sup>3</sup> provided very promising results, reducing the overall traffic from 11 to 48% (in the 5-server model) or from 15 to 52% (in the 10-server model), as we varied  $\alpha$  and  $\beta$ . These results served as proof of concept that the model can be very effective for sake of resource placement in distributed servers.

Furthermore, we also evaluated the product distribution

<sup>3</sup>We reduce the set of products distributed to the 50 most popular and the number of candidate clouds for becoming a server to up to 10.

by using log trace #2, described in Section A.. The trace was used to simulate a workload applied to a set of distributed servers employing the proposed resource placement strategy. We measure the goodness of the strategy through the cumulative number of remote allocations performed by all servers, that is, the number of requests received but not satisfied locally by a server, requiring a redirection to another server that has the desired item, which not only increases the user-perceived latency, but also the transaction cost. An even worse problem occurs when the product is not available in any of the distributed servers, meaning that the e-tailer may loose the client. For sake of evaluation, the lower the number of remote allocations, the closer the proposed strategy was to the actual workload. In order to validate our strategy, we compare it to three intuitive distribution strategies: round-robin, uniform by product, and weighted by total requests. Round-robin is characterized by the inventory of each product being entirely assigned to just one server, in a round-robin fashion, as defined by the popularity ranking of the products. In the uniform-by-product strategy, the inventory of each product is divided by the number of servers and the same fraction of the inventory is assigned to all servers, that is, every server is responsible for an equal share of the product inventory. Finally, in the weighted-by-total-requests strategy, the fraction of a product inventory assigned to a server is constant for all products and calculated as the ratio between the total number of requests to the server and the total number of requests in the workload.

We evaluated each strategy considering different time scales of trace log #2. We employed 4 time scales: (1) the first day; (2) the first 3 days; (3) the first week; (4) the entire log (15 days). Table 1 shows the evaluation results associated with each strategy. For each time scale, we multiplied the number of requests by a factor (14, 5, 2, and 1, respectively) so that the absolute demands are comparable, which can be seen in the second column. The third column (Allocation misses) presents the number of requests that would not be satisfied by the distributed servers for each time scale. The remaining columns show the number of redirections and the relative importance of these redirections in the overall number of allocations issued to the distributed servers.

Table 1 shows that the number of allocation misses varies significantly across time-scales, as a consequence of the temporal variation of the demands, as discussed

in Section A. We also observe that strategies *round-robin* and *uniform by product* perform poorly in all time scales. These results are somehow expected, considering the per-server demand variance, as shown in figure 4. It is also remarkable the similarity between the results of the weighted by server strategy and our model, not being possible to determine which of them is better. We explain this similarity by verifying the variance of the product demand among servers, which is significantly smaller in log trace #1 (Figure 4) when compared to log trace #2 (Figure 5). More specifically, we can see that AS6453 is responsible for at least 43% of the demand of the 50 most popular products in log trace #1, while in log trace #2 the demands vary from 35% to 63%. The small variance in log trace #1 resulted in a model solution very close to the weighted-by-server approach, but we believe that this is not always a valid assumption. In summary, our model covers naturally the solution provided by load-based strategies such as weighted-by-server and also addresses the scenarios where the product popularity distribution presents a high variance among servers.

## V CONCLUSIONS

In this paper we discuss the distribution of e-commerce services among different servers by solving a traffic-aware cost model. The solution is evaluated using an actual log from an e-tailer. The use of the solution given by the proposed model reduced the traffic-related costs up to 52%, depending on the ratio between traffic costs, and server installation and operation costs. The model allows system designers to investigate cost compromises and the impact of the application workload on the effectiveness of the distribution in terms of reducing traffic, and thus operational costs and user-perceived latency. We also evaluated the application of the model using another log from the same e-tailer and concluded that our strategy may be used as a resource placement function, since it accounts for the demand variance among servers in a per-product basis.

We believe that this model may be improved for its applicability in several directions. We intend to consider also coherence protocol costs and forwarding costs, since we assume that clients are aware of the product location. We will also investigate load balancing strategies between the distributed servers and the cost of such protocols. We will propose a more realistic cost model, in which the asymmetry between the incoming and outgoing traffics and their ratio to the transaction processing costs are taken into account. We also want to account for traffic generated by non-transactional requests such as *browse* and *search* requests, extending the work presented in [6]. Further, as mentioned in Section A., a small fraction of all product allocations becomes actual purchases, making parameters such as allocation-to-purchase fundamental for a more accurate model. Another direction is to develop heuristics

for approximate solutions to the model, allowing the distribution of larger number of products among more candidates. Finally, we plan to evaluate larger logs and apply the model to other dynamic e-commerce applications.

## REFERENCES

- [1] D. Agrawal, A. El, and A. Steinke. Epidemic algorithms in replicated databases, 1997.
- [2] V. Almeida, D. Menascé, R. Riedi, R. Fonseca, W. Meira Jr., and F. Ribeiro. Characterizing and modeling robot workload on e-business sites. to appear in ACM Sigmetrics, Boston, MA, June 2001.
- [3] M. Aron, D. Sanders, P. Druschel, and W. Zwaenepoel. Scalable content-aware request distribution in cluster-based network servers. In *In Annual Usenix Tech. Conf., 2000. San Diego, CA., 2000.*
- [4] Gustavo M. C. Gama, Wagner Meira Jr., Márcio L. B. Carvalho, and Dorgival O. Guedes. Traffic-aware distribution of e-commerce services. In *Proc. of the 17th ITC*, Sep 2001.
- [5] Mor Harchol-Balter, Tom Leighton, and Daniel Lewin. Resource discovery in distributed networks. In *Proc. of ACM PODC*, pages 229–237, 1999.
- [6] W. Meira Jr., D. Menascé, V. Almeida, and R. Fonseca. E-representative: a scalability scheme for e-commerce. In *Proc. of the 2nd WECWIS*, June 2000.
- [7] Jussi Kangasharju, James Roberts, and Keith W. Ross. Object replication strategies in content distribution networks. In *Proceedings of the 6th Web-Cache Workshop*, June 2001.
- [8] A. Luotonen. *Web Proxy Servers*. Prentice Hall, Upper Saddle River, New Jersey, 1998.
- [9] D. Menascé, V. Almeida, R. Riedi, F. Peligrinelli, R. Fonseca, and W. Meira Jr. In search of invariants for e-business workloads. In *In Proceedings of the 2nd ACM e-Commerce Conference*, pages 56–65, Minneapolis, MN, October 2000.
- [10] Inc. Merit Networks. Merit radb services. <http://www.radb.net>.
- [11] V. S. Pai, M. Aron, G. Banga, M. Svendsen, P. Druschel, W. Zwaenepoel, and E. Nahum. Locality-aware request distribution in cluster-based network servers. In *Proc. of the ACM ASPLOS-VIII*, pages 205–216, Oct 1998.
- [12] G. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA, 1949.