

Traffic-aware Distribution of e-Commerce Services

Gustavo Gama Wagner Meira Jr. Márcio Bunte Carvalho

Dorgival Guedes

e-SPEED – Department of Computer Science

Universidade Federal de Minas Gerais

Belo Horizonte – MG – Brazil

`{gmcgama,meira,mlbc,dorgival}@dcc.ufmg.br`

Abstract

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 others. 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 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 through an actual log from an e-tailer. The results show that the model allow 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.

1 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 others. 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 [9] 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 (using traditional cache technologies) responses. For those applications, there are already some proposals that are discussed in the next section. Peer-to-peer (P2P) service architectures such as Napster [7] are the latest proposal towards distributing services across the Internet, in this case employing the clients themselves as servers.

For a distributed e-commerce solution to be successful, we identify four related problems that must be properly addressed: server placement, product placement, resource discovery, and server load balance. They can be detailed as follows:

Server placement: how many servers should be used, and where should they be installed?

More servers lead to a higher degree of parallelism for accesses, but also to higher complexity as they need to exchange information.

Product placement: how many units of each product should be stored in each server?

An uniform distribution would be simpler, but most likely a distribution considering demands in the server neighborhood should work better.

Resource discovery: how should clients decide which of the servers to contact? Should they decide based on information on the resources available in each server, or simply use a simple assignment function based on network topology or geographical affinity?

Load balancing: if a server gets too many requests, should it forward some of them to other

server? And how to proceed if a server gets a request for an item it does not have, should it request it from another server or redirect the request?

In this context, resource discovery is usually handled by dynamic DNS replies which can be based on different strategies, but mostly on topological affinity in order to reduce delays, routing each request to the nearer server. Load balancing is a complex issue in itself, usually addressed by researchers in the area of distributed database systems. In this work we will focus on the two first problems, server and product placement.

Our main 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 should products be distributed among them in order to reduce overall network traffic.

When considering the impact of requests for a service on a large internetwork as a whole, traffic reductions can be directly associated with lower latencies, lower costs, and therefore higher customer satisfaction.

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 by each individual link. The link based approach leads to a much more complex and hard to solve model.

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. Any increase in link capacity to handle traffic will be eventually transferred to the source of their traffic, whether it is charged directly on each packet or on the cost of the connecting links.

Having that in mind, our proposal is to study server and product placement in terms of network clouds, just considering in which cloud a server would be installed. That 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. That has also 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, which may be the Autonomous Systems in it, for example.

This paper is divided into six sections. The next section briefly discusses some related work. In the sections that follow we discuss e-commerce services and propose a model to evaluate their distribution (Sections 3 and 4). In Section 5 we illustrate the application of the proposed model by distributing services of an actual e-tailer server. The last section present the conclusions and list some future work directions.

2 Related Work

In this section we discuss three topics that are related to e-commerce service distribution: (1) cache strategies to dynamic content; (2) distributed data placement, recovery and consistency; and (3) Internet topology modeling.

There has been developed few mechanisms to support dynamic content caching. In [4], the authors propose to store dynamic objects as Java *applets* in proxy cache servers, and executing them upon a user request to the cache server. Although this solution is generally applicable, cache servers may get overloaded by the number of Java Virtual Machines that run on it. Another approach, based on the replication of lists of items is proposed in [11] and [8]. In the former, the lists are used to build dynamic responses for a search engine, while in the later, the lists are used along with HTML templates to answer non-critical users' requests such as browse and search. None of these strategies, though, provide means to deal with transient data, such as product availability in a retail store or price in an auction merchant.

Several issues concerning data distribution have been extensively studied under different contexts. Database management systems, in particular, employ a wide range of algorithms, techniques, and strategies that were analyzed and applied in the last few years [1]. However, many of the strategies proposed cannot be applied or must be adapted when they are used in the WWW environment. Under this new environment, there are some recent work that presented some promising results. In [6], the authors describe the existing algorithms and propose a new method for resource discovery in distributed applications. In [15], a locality-aware distribution scheme for server clusters is implemented, achieving high cache hit ratios and good load balancing.

Modeling the Internet topology, which was an adjacent work in this paper, is one of the main goals of the RADB project [13]. The CAIDA (Cooperative Association for Internet

Data Analysis) [5], also provides several tools to help modeling and mining of Internet related information. In [14], the authors define three simple power-laws that characterize inter-domain topologies, through observation and analysis of snapshots of routing table logs. Furthermore, the causes behind those same power-laws are studied in [10], associating each of them to a set of factors (namely, preferential connectivity, incremental growth, distribution of nodes in space, and locality of edge connections). Additionally, a comparative analysis of network topology generators is presented, showing the relevance of each of the aforementioned factors to the resemblance and accuracy of synthetical network topologies. We are not aware of any studies that focus on modeling the Internet for sake of analyzing traffic-related issues.

3 e-Commerce Services

In this section, we discuss the e-commerce services and the nature of the data usually handled by a server while performing these 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, 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. In general, most e-commerce servers support the traditional four ACID¹ properties. Transactional support is important, for instance, to avoid race conditions while two simultaneous requests are answered. State maintenance allows the server to store temporarily data about the interaction of the user with the site, such as pages visited, products selected, and bids submitted. Reliable storage is usually provided by DBMS systems that run on the server machines. In some cases, the e-commerce servers exploit not only the robustness and reliability of DBMS systems, but also their transactional support.

We distinguish, without loss of generality, three types of data that may be accessed, either for reading or writing, while servicing e-commerce requests: (1) product², (2) customer, and (3) state. Product data comprise not only descriptive information about the products being

¹Atomicity, Consistency, Isolation, and Durability

²we use the term product to also designate the services that are commercialized by an e-commerce server.

commercialized but also inventory and other dynamic information, that is, information that changes as a consequence of the customer activity. Customer data comprise not only name and address but also his or her profile and history, which may be used for sake of personalization. State data, as mentioned, record the customer interactions and partial transactions, such as a shopping basket or the sequence of bids while auctioning for a product.

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 a hard task. The challenge while distributing services is that the locality of reference to the various instances of data is not simple, since a customer may choose two products or a given product may be chosen by more than one customer. We may potentially distribute both types of data, but usually it is not always an efficient strategy, as discussed next.

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 to him or her, in terms of network latency and bandwidth, that is, each customer will be assigned to a single server. Since the state is usually associated with a customer, the state data will also be stored in the same server where the respective customer data is stored. On the other hand, as we discuss in the workload characterization, products are naturally shared among customers that are far apart and do not have a identical profile. In this context, just product data is effectively distributed and must be kept coherent. In summary, based on the customer behavior, we may reduce the general problem of distributing all three types of data handled by an e-commerce server by considering just product distribution.

4 Traffic-aware service distribution

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 for satisfying requests. Traffic costs are function of the number of requests and distance (in terms of links traversed) between clients and a server. For sake of modeling the distribution of services, we assume that a client is aware of the location of the products, thus requesting it to a server that will be able to satisfy the request, and requests from the same cloud where the 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.

We first present the distribution model for a single product. Given N network clouds, the customers accessing from a cloud i is responsible for a demand of d_i items, where $1 \leq i \leq N$. In this case, our problem is to determine the number of products p_i that are managed by a server located in cloud i in order to minimize the total cost. We use r_{ij} to denote the number of requests originated at cloud i that is serviced by a server located at cloud j . So, $\forall d_i, d_i = \sum_{j=1}^N r_{ij}$ and $\forall p_j, p_j = \sum_{i=1}^N r_{ij}$. The global cost of a single-product service distribution is $C = \sum_{j=1}^N \sum_{i=1}^N (c_{ij} * r_{ij} + \alpha_j * r_{ij}) + \beta_j * s_j$ where $1 \leq i, j \leq N$, c_{ij} is the cost of transmitting a product request from cloud i to cloud j , α_j is the transaction cost of a request in a server in cloud j , β_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 cloud should have a server, since r_{ij} may be null indicating that there are no requests from cloud i to cloud j . Further, as we discuss in Section 5, it allows us to verify the trade-offs between transaction and traffic-related costs.

Extending the cost equation to multiple products is trivial, since we need just to add a new dimension k associated with each of the P products of the inventory. The overall cost for multiple products become $C = \sum_{k=1}^P \sum_{j=1}^N \sum_{i=1}^N (c_{ijk} * r_{ijk} + \alpha_j * r_{ijk}) + \beta_j * s_j$, where r_{ijk} is the demand from clients in cloud i to items of product k managed by a server located in cloud j , and the remaining parameters are the same of the single-product model.

We should note that this model determines the best distribution for a stream of requests, but the characteristics of this stream may change across time or even within a single time period, which may result in non-optimal assignments. In this case, we may employ a load

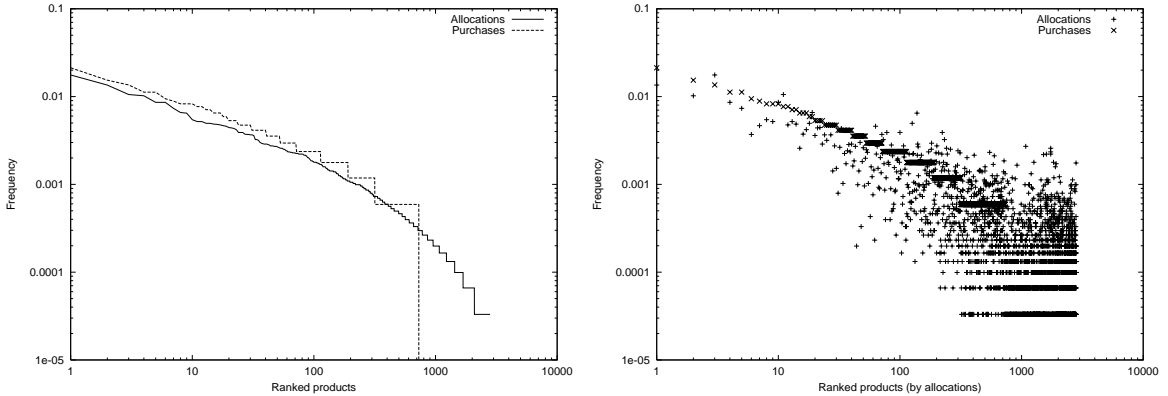


Figure 1: Frequency Distribution of Allocations and Purchases: (a) ranked by the respective function (b) ranked by purchases

balancing algorithm that uses the proposed model as a balancing function, as it will be discussed in Section 5.

5 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 role of the various costs on the distributions and how the variations in terms of workload affect the model utilization.

5.1 Workload Characterization

In this section, we present the workload characterization data for an e-tailer. We focus on the functions allocate and purchase, which require transactional support [8] and will benefit from our distribution model. A more complete characterization of e-business workloads may be found in [12, 2].

The starting point of our characterization is an access log of a virtual bookstore. The log was collected on August of 1999 and contains the 1,056,040 requests divided into 110,138 sessions. There are 30,275 (2.86% of the requests) allocation requests and 747 purchase requests (users buy in 0.7% of their visits), resulting in an allocate-to-purchase ratio of 2.46%³.

³The small number of purchase-related functions may be explained by a significant number of pricebots that request information to the site.

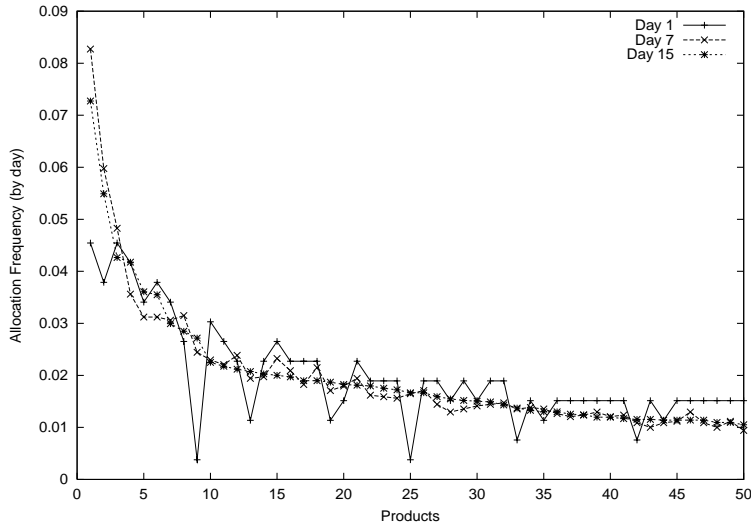


Figure 2: Temporal Variation of the Allocation Frequency per Product

We start our characterization by verifying the popularity of products in terms of allocations and purchases. In Figure 1 (a) 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[16] 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 ρ) 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 are requested by all customers, demanding an efficient distribution strategy.

Further, we can also observe that there is low correlation between the allocation and purchase popularity, that is, the fact that a product is frequently allocated does not mean that it is purchased at the same frequency. In Figure 1 (b), we can observe again the frequency distribution of both allocations and purchases now ranked just by the number of purchases (i.e., the leftmost points of both curves are associated with the most purchased product). It's remarkable how the allocation popularity varies for the same purchase popularity. For sake of distributing services, the high variance in terms of allocation-to-purchase ratio demand flexibility from the strategy in the sense that it has to eventually deallocate products

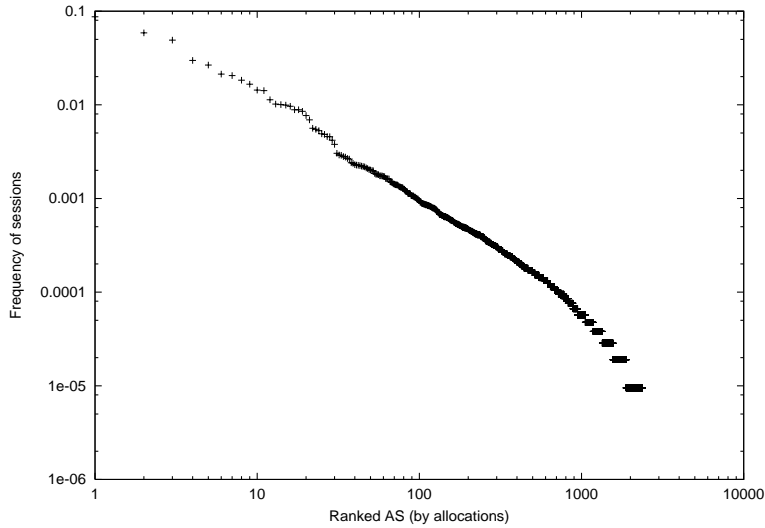


Figure 3: Frequency distribution of the number sessions originated by each AS

distributedly.

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.

We then characterize the number of operations performed by network clouds. In our case and in the results 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*[13] that 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.

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,

β/α	0	0.2	0.4	0.6	0.8	1
0	11386 (100.0%)	12880 (88.4%)	14316 (83.4%)	15507 (77.0%)	16262 (85.3%)	16770 (87.0%)
0.4	12277 (92.7%)	13771 (82.7%)	15210 (78.5%)	16269 (85.2%)	16869 (82.2%)	17386 (83.9%)
0.8	13168 (86.5%)	14663 (77.7%)	16104 (74.1%)	16876 (82.2%)	17476 (79.3%)	17933 (85.1%)
1.2	14060 (81.0%)	15554 (73.2%)	16863 (77.7%)	17484 (79.3%)	18083 (76.7%)	18470 (82.6%)
1.6	14951 (76.2%)	16445 (69.2%)	17491 (79.3%)	18091 (76.6%)	18687 (81.7%)	19007 (80.3%)
2	15798 (75.2%)	17259 (76.0%)	18098 (76.6%)	18698 (74.1%)	19224 (79.4%)	19543 (78.1%)

Table 1: Results using 5 candidates

handles this dynamicity by allowing flexible product distributions, and also is a powerful tool for evaluating load balancing strategies and their parameters.

5.2 Evaluation of a distributed e-tailer

In this section we present the evaluation of the application of the model proposed in Section 4 to the e-tailer data presented in the previous section. Our basic metric is the overall cost provided by the model. We first verify the impact of variations on the installation and per-transaction costs and then investigate how the temporal variation on the operation frequency affects the model-generated distributions, assessing the potential of dynamic load balancing strategies.

As mentioned in the last section, we use the AS as network cloud in our model. We estimated the local costs (α and β) of each cloud as a linear function of the connectivity of the respective AS, that is, the higher is the connectivity, the greater are the values of α and β . Our cost differentiation is based on the premise that a more connected cloud would charge more to host a distributed server. For sake of evaluating the trade-offs between traffic and local costs we also applied a varying factor to the α and β parameters, while the traffic is kept constant. In the experiments discussed below, we varied α 's factor from 0 to 1 and β 's factor from 0 to 2.

Our starting point is the non-distributed configuration, where all requests are serviced by the e-tailer server itself. In this case, the traffic-related cost is 21911 and the increase of both α and β to the maximum factor values considered resulted in a cost of just 21928, which is explained by the low values of α and β associated with the AS where the e-tailer server is located.

β/α	0	0.2	0.4	0.6	0.8	1
0	10613 (100.0%)	12492 (86.3%)	14311 (75.0%)	15592 (76.9%)	16535 (81.3%)	17219 (84.9%)
0.4	11320 (93.8%)	13253 (80.6%)	15084 (71.2%)	16190 (80.3%)	16944 (81.3%)	17545 (83.3%)
0.8	12028 (88.2%)	13979 (75.9%)	15756 (78.1%)	16555 (78.7%)	17268 (80.1%)	18180 (76.8%)
1.2	12736 (83.3%)	14687 (72.3%)	16083 (81.0%)	17202 (86.7%)	17639 (78.1%)	18085 (82.5%)
1.6	13429 (81.7%)	15374 (79.3%)	16716 (73.5%)	17496 (73.8%)	17965 (83.0%)	18407 (81.0%)
2	14044 (78.1%)	15830 (77.0%)	17189 (70.9%)	17779 (79.3%)	18287 (81.5%)	18729 (79.6%)

Table 2: Results using 10 candidates

We then evaluate the gains from distributing the service of the e-tailer. According to the model proposed in Section 4, any cloud may host a server, but this premise is not realistic, and also make the model computationally intractable, since we have thousands of products to be distributed among thousands of clouds. In order to solve the model, we employed a relaxed version of the model, where the clouds that are candidates to host a server are limited to a pre-defined set. More specifically, we chose the most connected clouds as candidates to host a server. Further, we also limited our distribution to the 50 most popular products, again for keeping the model computationally feasible.

Our first set of experiments solved the model for distributing 50 products among 5 clouds. The results are presented in Table 1 where each entry in the table show the overall cost and the contribution of the traffic to the overall cost (in the parenthesis) for each pair of factors applied to α and β . We can see that our distribution strategy reduce the costs significantly, from 11 to 48%, compared to the non-distributed case. As expected, traffic accounts for most of the cost in all cases, ranging from 69 to 100% of the cost, when the factors applied to α and β are 0.

In order to evaluate the impact of a larger number of candidates and also of our relaxation strategy, we also solved the same model, for varying α and β factors, considering 10 clouds as candidates. For the 10-candidate results presented in Table 2, we had to limit the number of iterations performed by the linear solver, because of processing time restrictions. The results show, however, that the solutions found are almost always better than the 5-candidate solutions. More specifically, the gains ranged from 15 to 52%, when compared to the non-distributed case.

We then analyze the impact of the temporal variation of the product demands on the

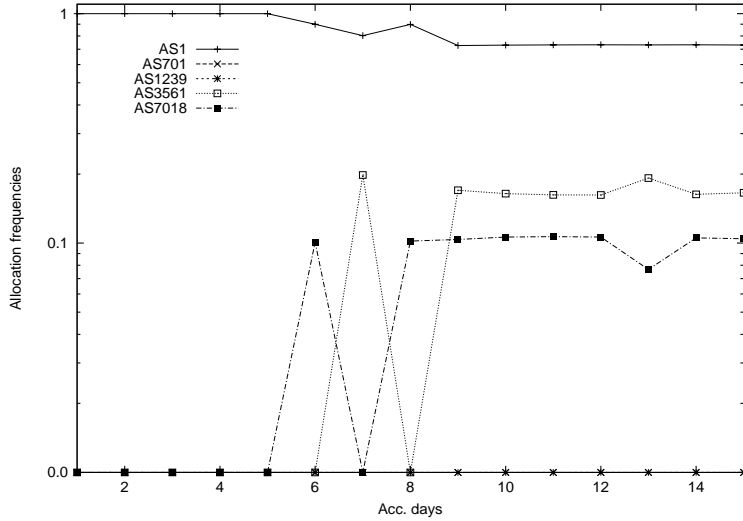


Figure 4: Product distribution among servers across time

service distribution, by analyzing the number of items assigned to each candidate across time. This analysis is based on solving the model for the cumulative workload for each of the 15 days of the log, that is, day 1 comprises the requests from the first day, day 2 comprises the requests from the first 2 days and so on. The results are presented in Figure 4, where we can see that AS1 is the only cloud to which products would be assigned till the sixth day, when items start being assigned to AS3561 and AS718. After the ninth day, the assignments become quite stable, providing a threshold of the minimum amount of information needed for sake of a load balancing strategy.

6 Conclusions

In this paper we discuss the distribution of e-commerce services between different servers by introducing a traffic-aware cost model and evaluating it through an actual log from an e-tailer. The use of the proposed model reduced the traffic-related costs up to 52%, depending on the ratio between traffic costs, and server installation and operation costs. We also evaluated the impact of the temporal workload variation, concluding that a time-period of 9 days would be enough for defining a distribution strategy. We should emphasize that these numerical results depend on the workload characteristics and should not be taken as absolute numbers, but as a proof of concept of the proposed model. In summary, 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 believe that this work may continue in several directions, which will improve the model and its applicability. We want to consider 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 generate a more detailed cost model, in the sense that the asymmetry between the incoming and outgoing traffics and their ratio to the transaction processing costs are more realistic. We also want to account for traffic generated by non-transactional requests such as browse and search requests, extending the work presented in [8]. Further, as mentioned in Section 5.1, a small fraction of the product allocations becomes an actual purchase, making parameters such as allocation-to-purchase fundamental for a more accurate model. Another direction is to develop heuristics to generate approximate solutions to the model, allowing the distribution of larger number of products among more candidates. Finally, we will evaluate larger logs and apply the presented 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. Menasc, 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 Technical Conference, 2000. San Diego, CA., 2000*.
- [4] P. Cao, J. Zhang, and Kevin Beach. Active cache: Caching dynamic contents on the web. In *Proceedings of IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 373–388, 1998.

- [5] Cooperative Association for Internet Data Analysis. The caida web site. <http://www.caida.org>.
- [6] Mor Harchol-Balter, Tom Leighton, and Daniel Lewin. Resource discovery in distributed networks. In *Proceedings of ACM Principles of Distributed Computing*, 1999.
- [7] Napster Inc. Napster home page. <http://www.napster.com>.
- [8] W. Meira Jr., D. Menascé, V. Almeida, and R. Fonseca. E-representative: a scalability scheme for e-commerce. In *Proceedings of the Second International Workshop on Advanced Issues of E-Commerce and Web-based Information Systems (WECWIS'00)*, June 2000.
- [9] A. Luotonen. *Web Proxy Servers*. Prentice Hall, Upper Saddle River, New Jersey, 1998.
- [10] A. Medina, I. Matta, and J. Byers. the origin of power laws in internet topologies, 2000.
- [11] W. Meira Jr., M. Cesário, R. Fonseca, and N. Ziviani. Integrating www caches and search engines. In *Proceedings of the IEEE 1999 Global Telecommunications Internet*, Rio de Janeiro, RJ, December 1999.
- [12] 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.
- [13] Inc. Merit Networks. Merit radb services. <http://www.radb.net>.
- [14] C. Michalis and F. Petros. On power-law relationships of the internet topology, 1999.
- [15] Vivek S. Pai, Mohit Aron, Guarav Banga, Michael Svendsen, Peter Druschel, Willy Zwaenepoel, and Erich Nahum. Locality-aware request distribution in cluster-based network servers. In *Proceedings of the ACM Eighth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*, Oct 1998.
- [16] G. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, Cambridge, MA, 1949.