Proc. 2001 Computer Measurement Group Conference, Orlando, FL, Dec., 2001.

TESTING E-COMMERCE SITE SCALABILITY WITH TPC-W

Ronald C Dodge JR
United States Army
Fort Leavenworth, KS 66027
ronald.dodge@us.army.mil

Daniel A. Menascé
Dept. of Computer Science
George Mason University
Fairfax, VA 22030-444
menasce@cs.gmu.edu

Daniel Barbará
Dept. of Information and Software Engineering
George Mason University
Fairfax, VA 22030-444
dbarbara@ise.gmu.edu

Being able to assess the scalability of Web and e-commerce sites is a challenge. The Transaction Processing Council (TPC) recently released a benchmark for e-commerce sites. This benchmark, called TPC-W, measures a system under test using Web Interactions per Second (WIPS) as its primary metric. This paper describes the benchmark's main features, the authors' experience in building a TPC-W compliant e-commerce site and a workload generator for TPC-W. It also presents performance metrics and an analysis of self-similarity obtained with the TPC-W site.

1. Introduction

By any measure, the growth of e-commerce activity is remarkable, prompting the need for continuing evaluation of how to provide better services under an enduring and increasing demand on Internet resources. The scalability analysis of e-commerce sites presents many challenges as described in [11]. One of these challenges has to do with the need to characterize the workload of these sites in a representative way. Many studies have been conducted with the purpose of understanding the workload of e-commerce sites and searching for invariants that cut across more than one type of e-commerce site [12,13]. The workload imposed by robots (e.g., pricebots and crawlers) was characterized recently [3] and their impact on Web caching was analyzed [4].

When comparing two systems (software, hardware, and architecture) that deliver e-commerce services we need to use a standard benchmark that is representative of e-commerce workloads and that can scale as needed. The term scalability here is related to both an increase in the number of users and an increase in the database size (e.g., number of customer orders, number of items in the catalog). The only available benchmark for e-commerce is TPC-W, designed by the Transaction Processing Council (www.tpc.org). This paper describes our experience in building a TPC-W compliant e-commerce site, called hereafter a TPC-W site, as well as a workload generator for that site.

The rest of the paper is organized as follows. Section two talks about workload characterization for e-commerce and describes the *Customer Behavior Model Graph (CBMG)*. Section three, discusses TPC-W and presents its main elements and metrics. The next section presents the TPC-W site we developed along with the workload generator. Section five discusses some results on the use of the TPC-W workload generator. Section six presents concluding remarks.

2. Workload Characterization for E-commerce

The workload of Web sites that provide information has been extensively studied and characterized at the level of HTTP requests [1, 2, 5, 7]. In e-commerce, customers interact with the site through *sessions*, which are sequences of consecutive requests to execute e-business functions (e.g., search, browse, select, add to cart, login, register, and pay) during a single visit to the site. Therefore, the workload of e-commerce sites is better described at the level of sessions. One way to capture the navigational pattern within a session is through the Customer Behavior Model Graph (CBMG) [10, 13], which describes patterns of user behavior, i.e., how users navigate through the site, which functions they

use and how often, and the frequency of transition from one e-business function to the other. Figure 2.1 depicts an example of a CBMG showing that customers may be in several different states—Home, Browse, Search, Select, Add, and Pay—and they may transition between these states as indicated by the arcs connecting the states. The numbers along the transitions indicate the probability of making the transition. A state not explicitly represented in the figure is the Exit state. Transitions to the Exit state are indicated by arrows leaving a state and not going to any other state in the figure. For example, the probability of going to the Exit state out of the Browse state is 0.15.

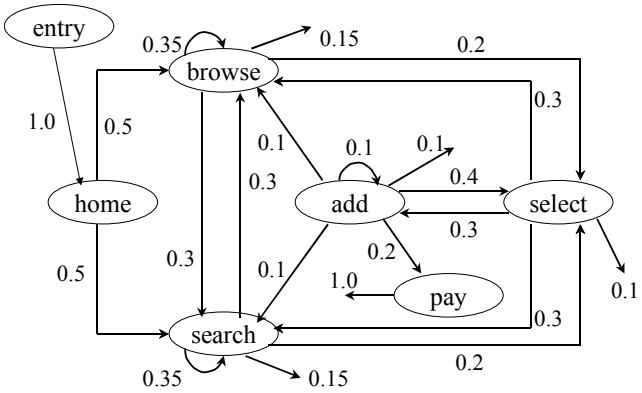

Figure 2.1 – Example of a Customer Behavior Model Graph (CBMG)

Clearly, several patterns of user behavior may be detected for the same site. For example, a fraction of the visits may come from occasional buyers who spend a lot of time "window-shopping" but very seldom buy anything. Other visits to the site may exhibit a pattern of heavy buyers, i.e., customers who know what they want and with a few clicks select one or more items, order and pay for them. It is important to realize that different customer behavior patterns generate different loads on the IT resources that support the site.

Another important result that has been reported in the recent literature [1,12] on workload characterization of Web and e-commerce sites shows a strong indication of Zipf's Law behavior. Zipf's Law [14]—a rather old result that empirically related the occurrence and popularity of words in a text—rephrased for the Web, states that if the objects requested are ranked from most popular (rank = 1) to least popular, then the number of references, *P*, to an object is inversely proportional to its rank *r*. That is, $P = k / r$ for some positive constant *k*. This result indicates that relatively few objects account for the majority of the references. The workload of high-volume online bookstore and auction sites was analyzed in [12]. The results indicated that 71% and 85.3% of all requests (from the bookstore and auction site, respectively) were for inline images. This high percentage of *cacheable* data coupled with the results of Zipf's Law strongly support the effectiveness of caching and the use of Content Delivery Networks (CDNs) in e-commerce environments.

Another result from [12] furthered the evidence presented in [2,5,7] of self-similar and bursty behavior. The resulting data showed both the bursty nature of e-commerce sessions and self-similar behavior. Self-similarity refers to similar bursty behavior across multiple time scales. The workload spikes experienced in WWW traffic are particularly critical for an e-business.

The results from [12] show that the session length, measured in number of requests to execute e-business functions, is heavy-tailed. In particular, the study showed that e-commerce sites are subject to factors that impact the tail of this distribution. Such factors include robot agents from price comparison robots and search engines that traverse the structure of an e-commerce site and tend to skew the length of sessions toward heavy-tailed distributions (e.g., Pareto). Sites that do not exhibit the agent behavior in the logs tend to exhibit lighter tails than those where the agent activity is observed.

3. TPC-W: A Benchmark for E-commerce

The Transaction Processing Performance Council (TPC) (www.tpc.org) provides industry standard methods for quantitatively measuring a system's transaction processing performance against a common standard as well as the performance of other systems. The TPC regards a transaction, as it is commonly understood in the business world: as a commercial exchange of goods, services, or money. A typical transaction, as defined by the TPC, would include the updating of a database system for such things as inventory control (goods), airline reservations (services), or banking (money). The council has developed performance benchmarks that measure transaction processing (TP) and database (DB) performance in terms of how many transactions a given system and database can perform per unit of time, e.g., transactions per second or transactions per minute.

The TPC expanded its suite of benchmarks in 2000 to include a transactional web benchmark for E-commerce systems, the TPC™ Benchmark W (TPC-W). The workload is performed in a controlled Internet commerce environment that simulates the activities of a business-oriented transactional web site. In the case of TPC-W the benchmark mimics the functionality of an online bookstore. The workload exercises a breadth of system components associated with such environments, which are characterized by:

- Multiple on-line browser sessions
- Dynamic page generation with database access and update
- Consistent web objects
- Use of Secure Socket Layers (SSL) for authentication
- Databases consisting of many tables with a wide variety of sizes, attributes, and relationships
- Database transaction integrity (ACID property, i.e., atomicity, consistency, isolation, and durability)

The primary performance metric reported by TPC-W is the number of web interactions processed per second (WIPS). TPC-W simulates three different profiles by varying the ratio of browse to buy activities: primarily shopping (WIPS), browsing (WIPSb), and web-based ordering (WIPSo). The primary metrics are the WIPS rate, the associated price per WIPS ($/WIPS), and the availability date of the priced configuration.

Some examples of actual values of the metrics posted on the TPC site as of June 14, 2001 are shown in Table 3.1 for illustration purposes. The names of the system that were used to obtain the results are omitted. The table was obtained for databases with 10,000 items and shows that the best system (system A) is able to process 5,745 Web Interactions per Second at the price of $69,00/WIPS. So, the total price (software + hardware + maintenance) of System A is $396,405, i.e., 5,745 x $69.00. System D costs almost the same, i.e., $349,675 but can only deliver 22% of the maximum throughput measured in WIPS.

Table 3.1 – Example of TPC-W for 10,000 Items in the Catalog.

Rank	System	WIPS	$/WIPS
1	A	5,745	69.00 US $
2	B	3,130	67.50 US $
3	C	3,008	81.77 US $
4	D	1,262	277.08 US $

3.1 TPC-W Emulated Browsers

The derivation of the WIPS family of metrics to describe the performance of the system under test (SUT) requires that a series of requests be presented to the system. TPC-W uses the concept of a group of Emulated Browsers (EB) to facilitate the generation of requests. An EB is defined as a process or thread that emulates a user communicating with the system under test using a browser by sending and receiving HTML content using HTTP and TCP/IP over a network connection. The number of EBs used for a given test is determined by the size and scaling factor of the SUT, as described further in section 3.4.

The software component that implements the EBs is called the Remote Browser Emulator (RBE). The RBE is responsible for starting each individual EB. The number of EBs created is subject to the scaling factor and each EB is responsible for executing a series of requests known as a user session. TPC-W defines a user session duration as the elapsed time between the first transaction executed by an EB and the current time. The session duration is controlled by a User Session Minimum Duration (USMD) time, defined as the minimum session duration for which a user session must last. The USMD is calculated as:

$$USMD = -\ln(r) \times \mu \qquad (1)$$

where *r* = *rand* (0.0, 1.0] and $\mu = 15$ minutes (the mean session duration) to mimic an exponentially distributed USMD. The USMD is truncated at 4 times the mean value (i.e., 60 minutes) in the case of extreme results. Each EB must generate a new USMD for each new session.

Each user session, is composed of multiple requests to the SUT. These requests are separated by a unique *think time* value Z defined as

$$Z = T2 - T1, \qquad (2)$$

where T1 is the time at which the last byte of the current page is received (this includes all requests for in-line images) and T2 is the time before the first byte of the next request is sent to the SUT. Think times implemented by emulated browsers are derived from an exponential distribution as

$$Z = -\ln(r) \times \mu \qquad (3)$$

where *r* = *rand* (0.0, 1.0] and $\mu = 7$ sec , the mean think time duration. The think time value is truncated at 10 times the mean value (i.e., 70 seconds) in the case of extreme results. Each EB must generate a new think time for each new request.

Since it is expected that the EB does some processing during the think time (e.g., parsing the response page, generating the next request, logging statistics, etc.), the amount of time the EB must wait before submitting the next request is actually the difference between the sampled think time and the processing time at the EB. This closely follows the concept of active and inactive think times first defined in [6].

We can now use the Response Time Law [8] to establish a relationship between the average response time *R* and the number *M* of Emulated Browsers. According to the Response Time Law,

$$R = \frac{M}{WIPS} - \overline{Z}\,, \qquad (4)$$

where *WIPS* is the throughput of the site and $\overline{Z}$ the average think time. Using $\overline{Z} = 7$ sec, the value specified by TPC-W for the average think time, and considering system A in Table 3.1, we would get an average response time of 1.7 sec for 50,000 concurrent users (1.7 = 50,000 / 5,745 – 7). System D in Table 3.1 would provide an average response time of 32.6 sec for the same number of concurrent users and the same average think time (32.6 = 50,000 / 1,262 – 7).

The session duration can be viewed graphically in Fig. 3.1.

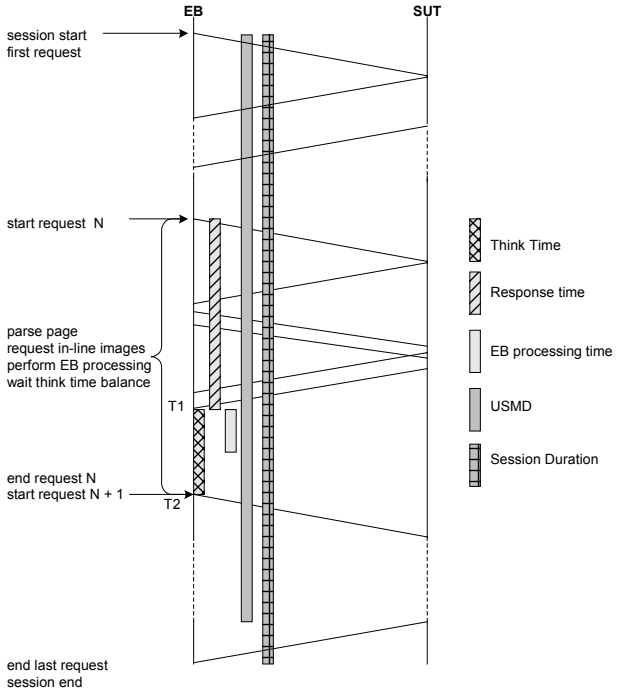

Figure 3.1 –Session Sequence

A new session can be started when the following conditions hold:

- The USMD has been met.
- The EB has just completed the think time following a request.
- The next request is determined to be the Home Page (this maintains the consistency of the CBMG.)

Each EB will continue to generate sessions until at least the minimum experiment run time has been met (30 minutes after steady state has been achieved).

A specific web interaction mix, i.e., a CBMG, guides an EB's request generation.

3.2 TPC-W CBMG

As we discussed, TPC-W is designed to mimic an online bookstore. A simplified version of the CBMG for TPC-W is shown in Fig. 3.2. A session always starts at the Home Page state.

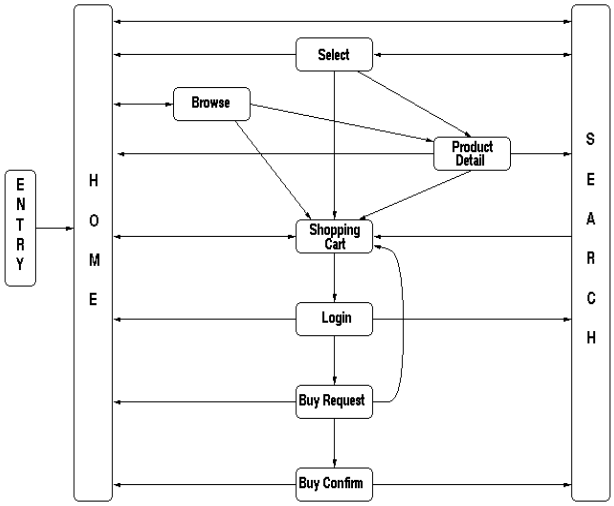

Figure 3.2 - Simplified CBMG for TPC-W

The full CBMG for the system, specified by TPC-W, consists of 14 unique pages, shown in Table 3.2. The pages are divided into two categories, Browse and Order. The Browse pages typically consist of non-secure requests with few transaction processing requirements. The Order pages are distinguished from the Browse pages by their greater processing requirements (i.e., database transaction count and/or secure processing requirement, SSL.)

Table 3.2 – TPC-W specified states

Browse	Order
Home	Shopping Cart
New Products	Customer Registration
Best Sellers	Buy Request
Product Detail	Buy Confirm
Search Request	Order Inquiry
Search Results	Order Display
	Admin Request
	Admin Confirm

TPC-W defines the set of probabilities that drive user transitions from one state to another as a Web Interaction Mix or CBMG. TPC-W defines three CBMGs: shopping, browsing, and ordering. The

shopping mix defines a browsing CBMG based on an average shopping scenario, and is the primary web interaction pattern. The browsing and ordering CBMGs present the SUT with requests that offer a significant change in the percentage of order requests being processed. Typically an order request requires the most processing time by the SUT due to increased database transactions and secure communication protocols (SSL).

The TPC-W specification outlines a set of minimum requirements for each page. These requirements include input and output requirements, minimum HTML page size, minimum number and size of inline objects, transaction processing requirements, and 90-percentile for the response time. For example, 90% of the Buy Confirm pages need to return in 5 seconds whereas 90% of Search Requests must have a response not exceeding 1 sec. The TPC provides an image generation function that produces JPEG images for use in each page.

3.3 The TPC-W Database

In addition to describing the requirements for each state of the CBMG, TPC-W also outlines the content of the e-commerce site database.

The database and the set of transaction requirements posed by each page are designed to exercise the e-commerce site in a manner representative of an Internet commerce application. The database is required to support look-up, insert, and update functionality as well as the capability to commit and rollback transactions.

The database consists of a minimum of eight tables with a defined minimum number of fields:

- Customer: Customer name and ID information,
- Address: Customer address data,
- Country: Country name and exchange rate information,
- Order: Order total and shipping information,
- Order line: Order line item data,
- Credit card: Credit card data,
- Item: Book information, and
- Author: Author data.

Additional tables required to support shopping cart transactions and state preservation are left to the individual implementation. The entity relationship (ER) diagram in Fig. 3.3 shows the implementation we used in our experiments.

TPC-W provides a function called WGEN to generate the item title (I_TITLE) and author last name (A_LNAME) fields in the database. We wrote the program required to populated all other fields and tables of the database.

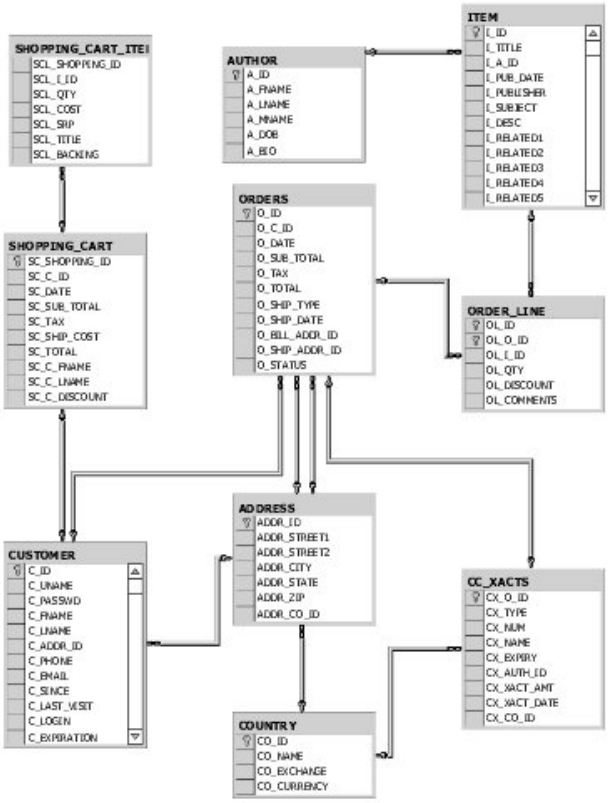

Figure 3.3 – Entity Relationship Diagram for TPC-W Implementation

3.4 TPC-W Scalability

The different components of the framework for the TPC-W e-commerce web site are tightly integrated. Scaling is fundamentally defined by the size of the concurrent customer population and the size of the store. Scaling requirements maintain the ratio between the cardinality of the database tables, the required storage space for the database, the transactional load placed on the site, and the number of EBs. Figures 3.4 and 3.5 display the relationships, specified by TPC-W, between the required data base table cardinalities and the desired concurrent customer population and store size. The cardinality of the country database table is constant at 92.

The idea behind the scaling method is that as the online store supports a higher number of concurrent users (i.e., EBs), the size of the databases that support the store operation has to scale up to accordingly. In other words, more users mean more customers in the CUSTOMER table, more orders in the ORDER table and more addresses stored in the ADDRESS table. The number of order lines and credit card transactions grows with the number of orders.

For example, according to Fig. 3.4, if 100 EBs are used, the CUSTOMER table should have 288,000 (=100 x 2,880) rows. The ORDER table should have 259,200 (= 288,000 x 0.9) rows and the ADDRESS table should have 576,000 (=288,000 x 2) rows. The ORDER_LINE table should have 777,600 (=259,200 x 3) rows and the CC_XACTS table should have 259,200 rows.

When reporting a value for the WIPS metric, the cardinality of the ITEM table has to be specified. For example, the WIPS value for system A in Table 3.1 would be specified as 5,745@10,000 .

It is important to note that since the number of EBs is constant throughout a TPC-W experiment, the number of initial EBs determines the achievable throughput (WIPS) of a given system for that run. The TPC-W specification places upper and lower bounds on the value of the WIPS metric that can be reported. Let us first look at the maximum possible theoretical value for the WIPS value. We rewrite below the Response Time Law equation given in Eq. (4).

$$R = \frac{\text{No. EBs}}{\text{WIPS}} - Z \qquad (5)$$

Using Z = 7, we can rewrite Eq. (5) as

$$\text{WIPS} = \frac{\text{No. EBs}}{R + 7} \qquad (6)$$

An upper bound for Eq. (6) is obtained when the response time is zero. So, an upper bound on the throughput is given by,

$$\text{WIPS} < \frac{\text{No. EBs}}{7}. \qquad (7)$$

On the other hand, if the response is too high, the value of WIPS will be very small, according to Eq. (6). To further constrain the system configuration, the reported WIPS must be at least 50% of its maximum theoretical value of No EBS/7. So, the value of WIPS must satisfy the following constraint:

(No. of EBs / 7) > WIPS > (No. of EBs / 14). (8)

This requirement is implemented to prevent under or over scaling the SUT for a given number of EBs.

4. TPC-W Workload Generator

The workload is generated from a multi-threaded Browser Emulator application connected to the e-commerce system by a 100Mbps Ethernet LAN. Figure 4.1 depicts the TPC-W site we built as well as the TPC-W Workload Generator.

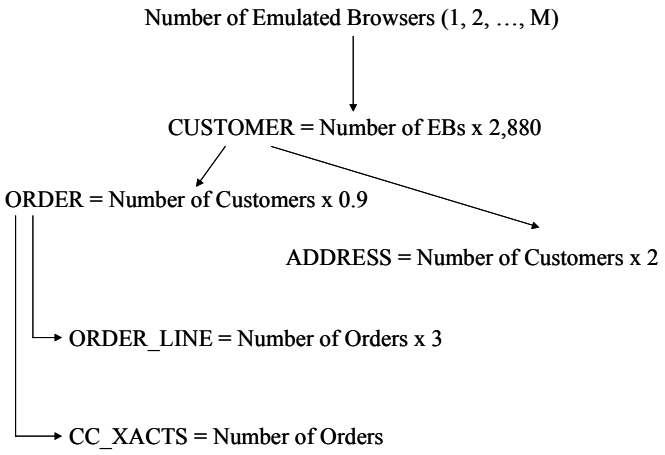

Figure 3.4 – Scalability: Concurrent Emulated Browsers

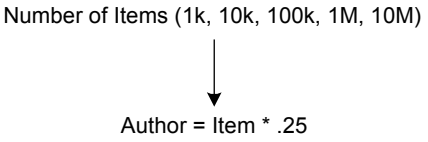

Figure 3.5 – Scalability: Store Size

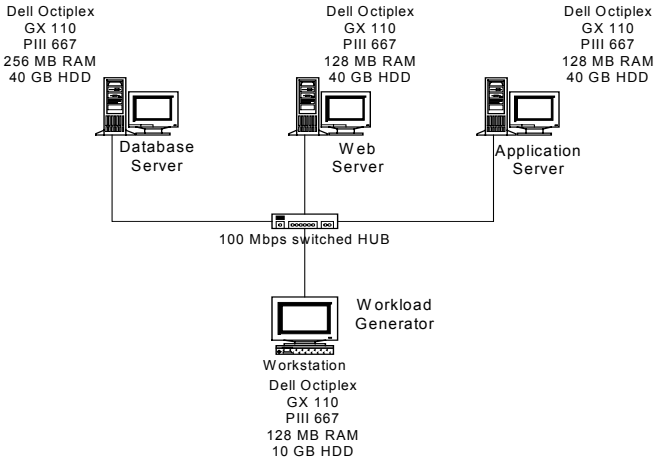

Figure 4.1 – Architecture of the TPC-W Site and TPC-W Workload Generator.

The Browser application uses multi-threading to emulate a group of client browsers and consists of a controller process and a variable number of browser threads. Each of the browser threads presents a unique workload to the e-commerce system. The navigation of a given thread through the e-commerce system follows the TPC-W navigation probabilities.

The workload generator can provide repeatable workloads to the server by having each thread seed its random number generator with the same number each time it is run. For example, if the experiment starts with ten initial simultaneous sessions, each browser thread seeds its random number generator with its generation number (e.g., 0, 1, 2… 9). When additional threads are created to increase the number of active sessions, they will continue the series and seed the random number generator accordingly. While this provides a very high degree of repeatability, the randomness with which the e-commerce system will refuse connections when it becomes heavily loaded will inject variability in the results. This however is unavoidable and would occur even if each browser used a trace for its workload generation. To statistically distinguish two workloads, each client browser can use a random seed for its random number generator.

Each browser thread emulates an HTTP/1.1 compliant browser. The browser thread first requests the HTML home page for the e-commerce site (dynamically generated). After receiving the response from the server, the browser parses the page, extracting information such as inline objects to request (images), customer ID and session ID, and any items that may be in the shopping cart.

Once the page has been parsed, the browser thread divides up the inline requests between a set of reader threads (three in our experiments) that use a pipelined/keep-alive request framework to retrieve the images. This technique, common in today's browsers, involves combining a series of requests into a single request message (pipelining) and then retrieving the response for each segment of the message without opening a new TCP connection to the server (keep-alive).

After the inline images have been received, the browser thread determines the composition of its next request. Again, this is accomplished through a routing table based on a given CBMG. Each possible request available to the browser has a set of tasks associated with it, as outlined in the TPC-W specification. For example, if the next state chosen is the shopping cart page, the browser thread determines if it has any item already in its cart, and then determines which items to increase, decrease or remove from the cart.

The controller process in the workload generator expands the TPC-W specified generator capabilities by optionally allowing for variability in the workload presented to the e-commerce system by increasing or decreasing the number of active sessions. When the controller process needs to reduce the workload, it signals the browser threads that the number of active sessions has been reduced. At the completion of each request, a browser thread will check to see if the browser population has been reduced to the new level. If not, it decrements the population count by one and sets its own session stop flag to true. The next time the browser thread completes a home page request it will exit. This is done to retain the probability distribution in the CBMG. If additional browsers are needed, the controller increments the population counter by the additional browser count and spawns the required number of browsers.

5. Experimental Results

This section describes some of the results we obtained when using the TPC-W workload generator in our TPC-W site. The first set of graphs shows average response time and the probability that requests are rejected, using exponentially distributed think times, as prescribed by TPC-W.

Figure 5.1, shows how the average response time varies as the arrival rate of requests increases. The vertical bars in the figure are the 95% confidence intervals on the measurements. Each point in the curve corresponds to an average over a very large number, in the thousands, of requests. As we can see, the response time increases and then levels off at high loads due to the fact that at that point, the probability of requests being rejected starts to increase very fast because the maximum number of requests in the system is achieved. This can be seen in Fig. 5.2.

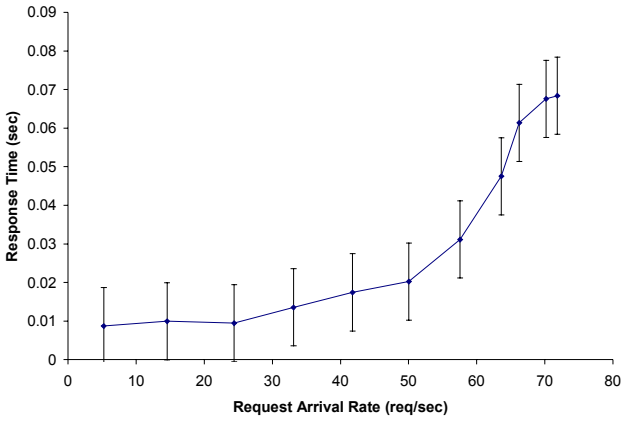

Figure 5.1 – Average Response Time vs. Request Arrival Rate.

We conducted an analysis of self-similarity on the number of requests that arrive at the TPC-W site. We used a data set from one of the experiments, which contained roughly 250,000 requests over a period of one hour and fifteen minutes and used a Pareto distribution to generate the think time values, which has been shown to more accurately represent the arrival characteristics at e-commerce sites [12].

The specification of TPC-W does not really reflect the bursty behavior of e-commerce workloads found in [MARPFM00]. It is our belief that a more realistic assessment of e-commerce sites should be carried out by slightly modifiying TPC-W to account for these more realistic distributions that exhibit heavy tails.

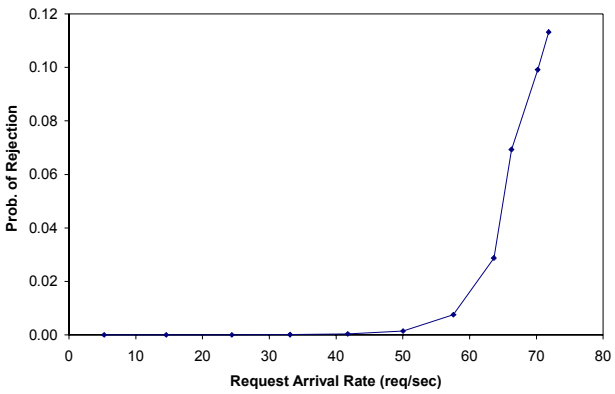

Figure 5.2 – Probability of Rejection vs. Arrival Rate

To carry out the analysis, we used a technique called the Variance Time Plot (VTP), which works as follows. We start with a time series $\{N_i, i = 1,...\}$ where N_i is the number of requests that arrived at time interval i. A time interval in our case is 2 sec. We now take the following steps:

1. For each m = 1, 2, …, build an aggregated time series by dividing the original time series $\{N_i, i = 1,...\}$ into blocks of m values and averaging these values. The resulting series is denoted $\{N_k^{(m)}, k = 1,...\}$, where k is the index of a block obtained by averaging m values in the original sequence. In other words,

$$N_k^{(m)} = \frac{1}{m} \sum_{i=m(k-1)+1}^{km} N_i$$

2. For each m = 1, 2, … compute the variance $Var[N^{(m)}]$, i.e., the variance of all values in the aggregated series for m. Suppose there are K blocks for a value of m in the series $\{N_k^{(m)}, k = 1,...\}$, then

$$Var[N^{(m)}] = \frac{1}{K-1} \sum_{k=1}^{K} (N_k^{(m)} - \overline{N}^{(m)})^2$$

where

$\overline{N}^{(m)} = \frac{1}{K} \sum_{k=1}^{K} N_k^{(m)}$, is the average of all values in the aggregated m-series.

3. Plot $\log_2 Var[N^{(m)}]$ vs. $\log_2 m$. This is the Variance Time Plot (VTP). If the original sequence $\{N_i, i = 1,...\}$ has a short-range dependence or no dependence, then for large values of m, $Var[N^{(m)}]$ tends to m^{-1}, which implies in a slope of –1 in the VTP. For long-range dependences (LRD),

$$Var[N^{(m)}] \sim m^{-\beta} \qquad m \to \infty, 0 < \beta < 1.$$

So, the more β deviates from 1, the more LRD the series is.

4. Compute the Hurst parameter H, commonly used to measure the intensity of LRD processes, defined as H = 1 - β/2, ½ < H < 1. LRD sequences tend to have Hurst parameter values closer to 1.

The results of our experiments showed strong long-range dependence, indicated by a Hurst parameter (H) of .9966, i.e., a value of $\beta = 0.0068$.

6. Concluding Remarks

The work presented here provides an overview of the TPC benchmark for e-commerce and some introductory results from a test bed developed to implement the benchmark framework and a workload generator.

The e-commerce site was built and designed around the specifications outlined by TPC-W. The workload generator was built to not only meet the workload characteristics required by TPC-W, but also to expand them to allow for more realistic workloads. Experiments using the TPC-W workload characteristics showed very predictable results when compared with theoretical models. Using a model more reflective of real-world e-commerce workload characteristics, we found that the traffic at the e-commerce site exhibited strong self-similarity; a common and important factor in today's e-commerce workloads.

The testbed described in this paper was used to test methods designed to dynamically adjust the QoS of E-commerce sites [9].

Acknowledgements

This research was conducted at the E-Center for E-business at George Mason University, with the support of Virginia's Center for Innovative

Technology,under award number INF-00-22, and of the TRW Foundation.

References

[1] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira. “Characterizing Reference Locality in the WWW,” In *IEEE Conference on Parallel and Distributed Information Systems*, Miami Beach, Florida, December 1996.

[2] M. Arlitt, R. Friedrich, and T. Jin, “Workload Characterization of a Web Proxy in a Cable Environment,” *ACM Performance Evaluation Review*, 27 (2), Aug. 1999, pp. 25--36.

[3] V. Almeida, D. A. Menascé, R. Riedi, F. Pelegrinelli, R. Fonseca, and W. Meira, Jr., "Characterizing and Modeling Robot Workload on E-Business Sites," *Proc. 2001 ACM Sigmetrics Conference,* Cambridge, MA, June 16-20, 2001

[4] V. Almeida, D. A. Menascé, R. Riedi, F. Pelegrinelli, R. Fonseca, and W. Meira, Jr., "Analyzing Web Robots and their Impact on Caching," *Proc. Sixth Workshop on Web Caching and Content Distribution*, Boston, MA, June 20-22, 2001.

[5] M. Arlitt and C. Williamson, “Web Server Workload Characterization: the Search for Invariants,” *Proc. 1996 ACM Sigmetrics Conf. Measurement & Modeling of Computer Systems*, Philadelphia, PA, May 23-26, pp. 126—137.

[6] Paul Barford and Mark Crovella, “Generating Representative Web Workloads for Network and Server Performance Evaluation,” Computer Science Department Boston University, Dec 1997.

[7] M. Crovella and A. Bestravos, “Self-Similarity in Word Wide Web Traffic: evidence and possible causes,” *Proc. 1996 SIGMETRICS Conf. Measurements Compt Syst. ACM,* Philidelphia, May 1996.

[8] Denning, P. J. and J. P. Buzen, “The Operational Analysis of Queuing Network Models,” Computing Surveys, vol. 10, no. 3, Sept. 1978, pp. 225-261.

[9] D. A. Menascé, D. Barbara, and R. Dodge, “Preserving QoS of E-commerce Sites Through Self-Tuning: A Performance Model Approach,” Proc. 2001 ACM Conference on E-commerce, Tampa, FL, October 14-17, 2001.

[10] D. A. Menascé and V. A. F. Almeida, *Scaling for E-Business: technologies, models, performance, and capacity planning*, Prentice Hall, 2000.

[11] D. A. Menascé and V. A. F. Almeida, “Challenges in Scaling E-Business Sites," *Proc. 2000 Computer Measurement Group Conference*, Orlando, FL, December 10-15, 2000.

[12] D. A. Menascé, V. Almeida, R. Riedi, F. Pelegrinelli, R. Fonseca, and W. Meira Jr, “In Search of Invariants for E-Business Workloads," *Proc. Second ACM Conference on Electronic Commerce,* Minneapolis, MN, October 17-20, 2000

[13] D. A. Menascé, V. Almeida, R. Fonseca, and M. A. Mendes, "A Methodology for Workload Characterization of E-commerce Sites," *Proc. First ACM Conference on Electronic Commerce*, Denver, CO, November 3-5, 1999.

[14] G Zipf, “Human behaviour and the Principle of Least Effort,” Addison-Wesley, Cambridge, MA, 1949.