WAM – The Weighted Average Method for Predicting the Performance of Systems with Bursts of Customer Sessions

Diwakar Krishnamurthy, Jerry Rolia, Min Xu
HP Laboratories
HPL-2008-66

Keyword(s):
Heavy-tailed distributions, Monte Carlo simulation, Distributed applications, Operational analysis, Queuing network models, Session-based systems

Abstract:
Predictive performance models are important tools that support system sizing, capacity planning, and systems management exercises. We introduce the Weighted Average Method (WAM) to improve the accuracy of analytic predictive performance models for systems with bursts of concurrent customers. WAM considers the customer population distribution at a system to reflect the impact of bursts. The WAM approach is robust with respect to distribution functions, including heavy-tail-like distributions, for workload parameters. We demonstrate the effectiveness of WAM using a case study involving a multi-tier TPC-W benchmark system. To demonstrate the utility of WAM with multiple performance modeling approaches we developed both Queuing Network Models and Layered Queuing Models for the system. Results indicate that WAM improves prediction accuracy for bursty workloads for QNMs and LQMs by 10% and 12%, respectively, with respect to a Markov Chain approach reported in the literature.

External Posting Date: June 7, 2008 [Fulltext] Approved for External Publication

Internal Posting Date: June 7, 2008 [Fulltext]

WAM – The Weighted Average Method for Predicting the Performance of Systems with Bursts of Customer Sessions

Diwakar Krishnamurthy[1], Jerry Rolia[2], and Min Xu[1]
[1]University of Calgary, Calgary, AB, Canada
[2]HP Labs, Bristol, U.K.
{dkrishna@ucalgary.ca,jerry.rolia@hp.com,m.xu@ucalgary.ca}

Abstract -- **Predictive performance models are important tools that support system sizing, capacity planning, and systems management exercises. We introduce the Weighted Average Method (WAM) to improve the accuracy of analytic predictive performance models for systems with bursts of concurrent customers. WAM considers the customer population distribution at a system to reflect the impact of bursts. The WAM approach is robust with respect to distribution functions, including heavy-tail-like distributions, for workload parameters. We demonstrate the effectiveness of WAM using a case study involving a multi-tier TPC-W benchmark system. To demonstrate the utility of WAM with multiple performance modeling approaches we developed both Queuing Network Models and Layered Queuing Models for the system. Results indicate that WAM improves prediction accuracy for bursty workloads for QNMs and LQMs by 10% and 12%, respectively, with respect to a Markov Chain approach reported in the literature.**

Index Terms –Heavy-tailed distributions, Monte Carlo simulation, Distributed applications, Operational analysis, Queuing network models, Session-based systems

I. INTRODUCTION

The purpose of this work is to improve the accuracy of predictive performance modeling techniques so that they may be more reliably applied in system sizing, capacity planning, and systems management exercises. In particular, we focus on the impact of high variability and heavy-tail distributions on the accuracy of predictions for system responsiveness.

The systems we consider are session-based systems, e.g., e-commerce and enterprise application systems. A *session* is defined a sequence of related customer requests that accomplish some business purpose. For example, in an e-commerce system, requests may be for a `Home` page or a `Buy` transaction. The number of requests submitted by a session is defined as the *session length*. We argue that for systems with bursts in the number of concurrent sessions, knowing the mean or maximum number of concurrent customer sessions is not generally sufficient to enable accurate performance predictions. It is necessary to consider the distribution of concurrent customer sessions. We refer to this distribution as the session *population distribution*.

Several studies have indicated that multi-tier session-based systems experience bursty workloads and that burstiness can adversely affect performance [18] [19]. Techniques have been proposed to reflect the impact with heavy-tail distributions in performance models [3][9], but they are not general enough for the problem we consider. Hybrid models that combine Markov-chain birth-death processes with Queuing Network Models (QNM) have been proposed to reflect load

dependent behaviour on mean response times [28]. However, these techniques are not adequate for capturing the impact of complex underlying workload parameter distributions that can contribute to burstiness in population distribution.

In this paper we propose and evaluate a new approach to estimate population distribution called the Weighted Average Method (WAM). WAM is motivated by the hybrid approach mentioned previously but does not rely on a birth-death model. Instead it exploits a fast Monte Carlo simulation to estimate population distribution. The primary advantage of the newly proposed method is that it is more robust with respect to the distributions that contribute to bursty behaviour. For example, it permits the study of arbitrary distribution functions for workload parameters such as session inter-arrival time, think time, and session length in a straightforward way.

We demonstrate the WAM technique in a study involving a multi-tier TPC-W [25] benchmark system. The system was subjected to controlled workloads to explore its responsiveness when subjected to bursty behaviour. We developed both QNMs and extended QNMs called Layered Queuing Models (LQM) [7][8] for the system. The results indicate that modeling approaches that only consider the mean number of concurrent sessions produce very poor estimates of mean response time for systems with bursty workloads. The average prediction error for bursty workloads is nearly 24% and 21% for the QNM, and the LQM, respectively. Furthermore, for bursty workloads, using the QNM and LQM models in combination with a Markov birth-death model does not improve prediction accuracy significantly. In contrast, the WAM approach significantly improves the accuracy of mean response time predictions. For bursty workloads, average prediction errors dropped by 12% and 10% for LQMs and QNMs, respectively, as compared to the Markov birth-death approach. Moreover, the LQM-based WAM approach had much lower average error and range of errors than the QNM-based WAM approach.

The remainder of the paper is organized as follows. Section II describes related work. Section III describes the WAM approach in detail. A measurement study for the multi-tier TPC-W system is presented in Section IV. The section provides insights into the impact of burstiness on session-based systems. Section V presents a QNM and a LQM for the multi-tier system. Section

VI investigates the accuracy of WAM in predicting the mean response times for the experiments described in Section IV. Summary and concluding remarks are offered in Section VII.

II. BACKGROUND AND RELATED WORK

Several studies have indicated that multi-tier session-based systems experience bursty workloads and that burstiness can adversely affect performance. Menasce *et al.* [19] characterized the workloads observed at an e-commerce system and an auction system. The authors found that both systems were characterized by bursty arrivals of requests over several timescales. They invoke the properties of the well-known ON-OFF process [20] to argue that the bursts observed at fine timescales, i.e., several dozen seconds, were due to the heavy-tailed nature of the session length distributions observed at the systems and the presence of think t imes in sessions. Vallamsetty *et al.* also noticed similar burstiness in the arrival of requests at another real e-commerce system [18]. The authors attribute this phenomenon to the highly variable service times for requests at the backend tiers of the system, i.e., the application servers and the database server. Krishnamurthy *et al*. [21] showed for a multi-tier system that distributions which cause highly variable session lengths, think times, and request resource demands result in high variability in the customer population distribution and hence burstiness in request arrivals at fine timescales. This suggests that modeling customer population distribution may be helpful for modeling the impact of burstiness. In addition to such fine timescale burstiness, burstiness has also been observed at coarser timescales, e.g., hours, days, in real-session based systems [22].

Burstiness can have a big impact on how predictable or repeatable a system's behaviour is in response to similar workloads. Crovella and Lipsky showed that the steady-state values for performance measures from multiple statistically identical simulation runs that use heavy-tailed distributions can result in very different measures for each run [24]. Krishnamurthy *et al.* [21] confirmed this for multi-tier software systems. For a TPC-W system servicing bursty workloads, the authors found that multiple statistically identical measurement runs with the same mean resource demands and same throughput resulted in significantly different mean response time measurements. These results suggest the need for modeling approaches that characterize a range of possible system behaviours under bursty workloads.

There are many examples of predictive performance models for systems. These include Markov chains [4] and related models such as Stochastic Activity Networks (SAN) [5] and Petri-Nets [6],

QNMs [1][2][3], and extended-QNMs such as LQMs [7][8]. The Markov chain related approaches enable very detailed models of systems. But in general, their solution efficiency does not scale to support even modestly sized systems of the kind we consider. Mean value analysis (MVA) for QNMs [2] offers a more restrictive modeling technique than Markov chains. Yet, it is a much more efficient technique for obtaining exact, and approximate [10][11], solutions for QNMs and as a result it can be used to study larger systems.

MVA and QNMs have been used to study computer system performance since the early 1970's. Several researchers have recently applied them directly to the study of multi-tier systems [13][14]. LQMs are based on QNMs, and were developed starting in the 1980's to consider the performance impact of software interactions in multi-tier software systems, e.g., systems that have contention for software resources such as threads. Tiwari *et al.* [15] report that layered queuing networks were more appropriate for modeling a J2EE application than a Petri-Net based approach [16] because they better addressed issues of scale. Balsamo *et al.* [17] conclude that extended QNM-based approaches, such as LQMs, are the most appropriate modeling abstraction for multi-tiered software environments.

MVA of QNMs and LQMs only consider the average customer population of a system. More complex techniques exist which could potentially be used for modeling the impact of burstiness. Classical queuing theory offers G/G/* queues [4] that can take into account the first and second moments of any arbitrary request inter-arrival time distribution. However, exact solution methods for mean response times do not exist for networks of such queues and reliable estimates from approximate solutions are difficult to obtain [4]. Furthermore, heavy-tail-like distributions require more than the first two moments for a proper characterization. Recently, Psounis *et al.* [9] considered a single multi-server queue that is subjected to heavy-tail-like distributions. However, the approach has not been extended to queuing networks.

Menasce and Almeida propose techniques that consider heavy-tailed distributions and bursty request arrivals for Web server systems [3]. Specifically, they describe a QNM that reflects the impact of a heavy-tailed file size distribution at Web servers serving static HTML pages. The authors argue that a multi-class model where the classes represent requests for files belonging to different file size ranges is more suited for capturing the impact of the heavy-tailed distribution

than a single class model. This technique is specific to systems that serve static files. It is not intended for transaction-oriented, session-based systems of the kind considered in this work. The authors also propose another heuristic technique that uses a QNM to reflect the impact of burstiness in request arrivals. The technique splits a given HTTP request log into equal sized time periods. It counts the number of time periods for which the average request arrival rate exceeded the request arrival rate observed over the entire log. This count is used to compute a burstiness factor which is in turn used to inflate the service demand of the bottleneck device in a QNM [3]. However, the technique was not validated with respect to measurements and was not proposed as a constructive technique that permits a performance analyst to assess the impact of distributions that contribute to burstiness on mean response time behaviour.

Menasce and Bennani [26] used a hybrid model that combines a Markov chain birth-death model with QNMs to capture the customer population distribution at multi-threaded servers. However, the authors' use of the population distribution was not for modeling the impact of burstiness. Instead they focused on load dependent behaviour. This is an alternative approach to using the threaded servers directly supported via residence time expressions in LQMs. The hybrid technique has also been used to study the performance of systems that are characterized by both open customer arrivals and closed customer circulations [28].

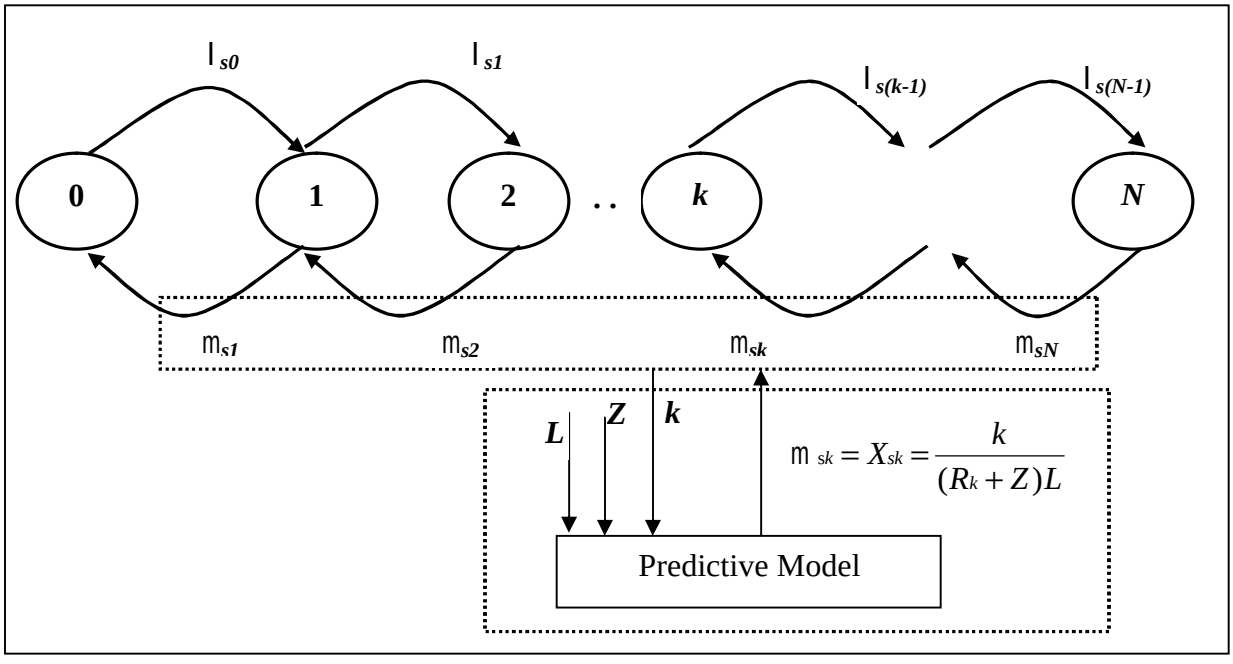

Figure 1: Hybrid Markov chain birth-death model for a session-based system

We apply the hybrid approach for a session-based system as shown in Figure 1. The hybrid approach is treated as a baseline approach for modeling session population distribution in the case study of Section VI. In Figure 1, the birth-death process has multiple states, S_k for k=0...N. Each state k denotes the number of concurrent sessions in the session-based system. Sessions arrive at the system from the outside world. Each session arrival causes the number of concurrent sessions to increase by 1. The rates at which such transitions occur are given by the state dependent session arrival rates λ_{sk}. A session submits L requests on an average, i.e., mean number of visits, where L is the mean session length. Z denotes the mean think time between successive requests in a session. Each session completion causes the number of concurrent sessions to decrease by 1. The rates at which such transitions occur are given by the session death rates μ_{sk}. At a given state k, k concurrent sessions are competing for the session-based system's resources. Consequently, as shown in (1) the death rate μ_{sk} can be calculated as the session throughput X_{sk} obtained by solving a closed QNM or LQM with a session population of k and mean think time of Z.

$$\mu_{sk} = X_{sk} = \frac{k}{(R_k + Z)L} \qquad (1)$$

In (1) R_k is the mean request response time obtained from the predictive model. Balance equations involving the birth and death rates can be solved to obtain the probability P_k of residing in each state k as follows:

$$P_0 = \frac{1}{1 + \sum_{k=1}^{N} \prod_{i=1}^{k} \frac{\lambda_{s(i-1)}}{X_{si}}} \qquad (2)$$

$$P_k = P_0 \prod_{i=1}^{k} \frac{\lambda_{s(i-1)}}{X_{si}} \qquad (3)$$

The probabilities P_k, for k=0…N defines the population distribution. Equations (2) and (3) can be written in terms of request arrival and completion rates. The request arrival rate λ_k is the session arrival rate λ_{sk} multiplied by the mean session length L. Similarly, the request throughput X_{sk} is the session throughput X_{sk} multiplied by the mean session length L. Using these relationships (2) and (3) can be rewritten in terms of request-level rates as follows:

$$P_0 = \frac{1}{1 + \sum_{k=1}^{N} \prod_{i=1}^{k} \frac{\lambda_{i-1}}{X_i}} \qquad (4)$$

$$P_k = P_0 \prod_{i=1}^{k} \frac{\lambda_{i-1}}{X_i} \qquad (5)$$

The mean request response time for the system is estimated using Little's law [27] as follows:

$$R_{mean} = \frac{\sum_{i=1}^{N} P_i X_i R_i}{\sum_{i=1}^{N} P_i X_i} \qquad (6)$$

We note that the hybrid technique's estimates of the population distribution are accurate only for workloads that cause the distributions of time spent at each population level to be exponentially distributed [4]. However, this is not expected to be the case for systems affected by burstiness.

The WAM approach that we present in this paper requires a trace of sessions to reflect the distributions under study. We exploit the SWAT method [21][29] to create such traces. SWAT permits the specification of a *workload mix* in terms of the ratio of different request types, e.g., `Home`, `Browse`, `Buy`, supported by the system under test. It also supports the specification of arbitrary session inter-arrival time, request think time, and session length distributions. SWAT uses a set of pre-existing and semantically correct sessions to create a workload for a test that satisfies a desired specification. The workload is modeled as a trace file of sessions *S*. We employ SWAT in Section IV to submit controlled workloads based on *S* to a TPC-W multi-tier system to obtain measurement results. SWAT also provides traces of sessions for WAM's estimation process for session population distribution. The following section describes the WAM approach.

III. WAM

This section describes WAM which is a method for improving the accuracy of performance predictions for systems with bursts in the number of concurrent customer sessions. The method is motivated by the approach in Figure 1 but uses a Monte Carlo simulation to quickly estimate the population distribution, i.e., per population level probabilities P_k for k=0…N, rather than relying on the closed formulae for the birth-death process from (4) and (5). The remainder of the

performance prediction method is similar to that shown in Figure 1. We now describe the population distribution estimation process. The algorithm for the population distribution estimator is summarized in Figure 2.

The WAM approach relies upon the following:

- A trace file of sessions *S*;
- A sequence *R* of mean request response time estimates R_k for k=0…N for the system – one for each concurrent session population level *k* as obtained by solving a predictive model with a mean customer think time of *Z* seconds; and,
- A sequence *X* of mean request throughputs X_k for k=0…N for the system – one for each concurrent session population level *k*.

A trace file *S* can be based on a historical session log from a real system, or it can be synthetically generated using a tool such as SWAT. Each session in the trace has an *arrival time*. Each request has a *session identifier*, a *start time* and *end time* such that (*end time – start time*) is the *response time* of the request, and a flag that indicates whether a request is the last request for its session. For all but the last request in a session, we define a request's *think time* as the time between its *end time* and the *start time* of the next request in the session. The first request of a session has a *start time* that is equal to its session's *arrival time*. The sequence *R* of response time estimates is obtained from a performance model for the system, e.g., a QNM or an LQM. The sequence *X* of throughput estimates are obtained from the trace *S* using the method described shortly.

The population distribution estimator operates as follows. When using a historical trace file *S*, per request response times are known so the sequence *R* of response time estimates is not needed to compute the population distribution. The population distribution estimator computes the P_k for k=0...N values by traversing the trace of sessions *S* noting when the first request of each session starts and the last completes. In this way it is able to keep track of and report the aggregate time that the system has spent at each session population level. When normalized with respect to the total simulated time this gives the population distribution P_k, k=0…N. Furthermore, as shown in Figure 2 the population distribution estimator also tracks the aggregate number of request completions observed at each session population level. Knowledge of these completions and the aggregate time spent at each session population level allows us to compute estimates of X_k for

k=0…N. The simulations are very quick, essentially requiring the time to traverse the trace file and are robust with respect to arbitrary workload parameter distributions.

When using a synthetically generated session trace file, the session arrival times, think times, and session lengths are known from the trace. However, only the first request's start time is known. The request end times and hence response times are not known. As the population distribution estimator traverses the session trace, each time it encounters a new request, it estimates the request response time as the mean request response time given by R based on the current estimate for the number of concurrent sessions. That is, if the current session population is k, then R_k is used to estimate the *end time* for the request as *start time* + R_k. The request's think time is recorded in the trace and could be from any desired distribution. The next request has a *start time* equal to the *end time* + *think time* from the previous request. We note that using the mean response time from a model is an approximation of the response time for the request. The effectiveness of this approximation is evaluated in Section VI. Finally, WAM computes the overall estimate for mean response time in the same manner as (6).

WAM can also be used to explore the predictability of a system's behaviour. By using different seeds for random number generation, SWAT can be used to generate multiple session trace files that match the desired workload parameters. Each trace may provide different estimates for the population distribution and may result in a different estimate for mean system request response time. As mentioned in Section II, this is expected for systems influenced by heavy-tailed distributions. Each execution is an example of how the system may behave. A range of estimated mean response times, from multiple simulations, provides information about how variable, i.e., un-predictable, we can expect a system's behaviour to be.

1. Create a Future Event List (*FEL*). *FEL* stores events in chronological order.
2. *Current_Population*=0
3. *State_Start_Time*=0
4. *State_End_Time*=0
5. Initialize elements of *Aggregate_State_Time* array to 0. This array has N_{max} elements where N_{max} is the maximum population.
6. Initialize elements of *Aggregate_State_Completions* array to 0. This array has N_{max} elements.
7. Obtain *Predictive_Model_Response_Time* array by solving a predictive model. This array has N_{max} elements.
8. Create request submission events corresponding to first requests of all sessions in trace *S*. Store the events in the *FEL*
9. **While** *FEL* is non-empty
    Select earliest event in *FEL*
    If event is submission of a request
        If request is first request in a session
            State_End_Time = start time of request
            Aggregate_State_Time[*Current_Population*]+=(*State_End_Time-State_Start_Time*)
            Completions=Request completions in the period (*State_Start_Time, State_End_Time*)
            Aggregate_State_Completions[*Current_Population*]+=*Completions*
            State_Start_Time=State_End_Time
            Current_Population+=1
        End If
        If *S* is a historical trace
            Response_Time = Get actual response time of request
        End If
        If *S* is a synthetic trace
            Response_Time = Predictive_Model_Response_Time[*Current_Population*]
        End If
        Create a request completion event at (*State_Start_Time*+*Response_Time*)
        Update *FEL* with the event
        Continue
    If event is completion of a request
        If request is last request in a session
            State_End_Time = end time of request
            Aggregate_State_Time[*Current_Population*]+=(*State_End_Time-State_Start_Time*)
            Completions=Request completions in the period (*State_Start_Time, State_End_Time*)
            Aggregate_State_Completions[*Current_Population*]+=*Completions*
            State_Start_Time=State_End_Time
            Current_Population-=1
            Continue
        End If
        Think_Time = Get think time of request
        Create a request submission event at (*State_Start_Time*+*Think_Time*)
        Update *FEL* with the event
        Continue

End While

10. Compute *Total_Time* as sum of elements of *Aggregate_State_Time*
11. Compute P_k values by dividing each element of *Aggregate_State_Time* by *Total_Time*
12. Compute X_k by dividing each element of *Aggregate_State_Completions* with the corresponding element of *Aggregate_State_Time*
13. Use equation (6) to compute mean response time

Figure 2: Algorithm for the population distribution estimator and WAM

We note that the WAM algorithm of Figure 2 does not take into account embedded requests for objects such as images and multimedia files. Such content is often hosted on external servers or content delivery networks as was the case with the real systems studied in [23] and [18]. We consider support for embedded requests as future work.

IV. MEASUREMENTS FOR A MULTI-TIER SYSTEM

This section applies the SWAT tool to perform controlled experiments on a multi-tier e-commerce system executing the TPC-W bookstore application. We consider a subset of the results of our earlier work [21] that was aimed at demonstrating the SWAT workload generator. We augment those results with additional measurement scenarios. In particular, we consider cases that cause greater disk demands and that have larger numbers of items in the TPC-W database, respectively. The additional cases explore a greater range of system behaviours. The measurements provide insights into the impact of burstiness on measured mean demands and mean response times for the session based system. The measurements are also used to obtain parameters for the predictive models in Section V and to assess the accuracy of WAM with QNMs and LQMs in Section VI.

A. Experiment Setup

The experimental setup consists of a client node, a Web and application server node and a database node connected together by a non-blocking Fast Ethernet switch, which provides dedicated 100 Mbps connectivity to each node. The *client* node is dedicated exclusively to execute an *httperf* Web request generator [30] that submits the synthetic workloads used in this study. The *Web/application server* node executes the Web and application server tiers. It implements the TPC-W application's business logic and communicates with the TPC-W database. The *database* node executes the database server which manages the TPC-W database. Finally, a performance monitoring utility is employed that collects a user-specifiable set of performance measures from both server nodes at regular specified sampling intervals.

The TPC-W application is deployed on Web, application, and database servers that are part of a commercial off-the-shelf software product. The name of the product has been withheld due to a non-disclosure agreement with the vendor. The system is configured to not serve images. Image requests were not submitted in any of our experiments. We note that that the experiments

presented in this study are not TPC-W benchmark runs. The TPC-W bookstore system merely serves as an example system for the study.

All our experiments employ HTTP 1.1 over SSL. Configuration parameters related to HTTP 1.1, e.g., persistent connection timeout, are chosen to force a single connection per session irrespective of session duration or the load on the system. This ensures that two workloads with the same number of sessions, mean session length, and mean think time impose the same connection establishment and connection shutdown overheads on the Web server. Consequently, any difference in performance between them is solely due to differences in the higher-level workload characteristics, i.e., session length distribution, think time distribution, and workload mix.

The number of server processes and the threading levels are set as follows. The number of Web server threads is set to be 1000. This was much greater than the maximum number of concurrent connections encountered in the experiments. The number of application server processes is fixed at 16, an upper limit imposed by the application. The number of database server threads for the database server was set to the upper limit of 32.

The primary performance metric of interest for the study is the user-perceived mean response time (R_{mean}) for the requests at the TPC-W system. This metric is of interest for system sizing, capacity planning, and service level management exercises. We define *response tim*e as the time between initiating a TCP connection for a HTTP request and receiving the last byte of the corresponding HTTP response. The measured response time is a good indicator of the delay suffered by the request at the TPC-W system, provided the network and the client workload generator node are not saturated.

B. Experiment Design

The following factors are considered for the experiments: a) session inter-arrival time distribution; b) session length distribution; c) think time distribution; d) workload mix; and, e) application settings.

For the session inter-arrival time distribution, we assume session arrivals are uncorrelated, which is consistent with several previous studies, e.g., [31]. Consequently, an exponential distribution

is used to generate session inter-arrival times for all experiments. The mean session inter-arrival time is chosen to achieve desired utilizations at the bottleneck resources. We note that a Poisson arrival process for sessions does not imply a non-bursty arrival of requests. As mentioned in Section II, the burstiness of the request arrival pattern depends on various attributes such as the distributions of session length and think time.

For the session lengths and think times, we consider two different distributions: empirical and bounded Pareto. We use these to represent the expected and worst cases for variability, respectively. The *empirical distributions* are obtained from workload data collected from a large e-commerce system [23]. Since that system did not serve requests for images embedded in Web pages, we use the request inter-arrival times within a session as measured at the system as an approximation of the think times within sessions. The bounded Pareto distribution [32], a "heavy-tail-like" distribution, is used to study the impact of distributions that have a *slightly heavier tail* relative to the empirical distribution. The distribution is characterized by three parameters: •, the tail index, which governs the rate at which the tail of the distribution decays, *k* the smallest possible observation and *p* the largest possible observation. These parameters are deduced as follows. The parameter *p* is set to the maximum observation obtained from the empirical distribution. We then choose *k* and • such that mean of the empirical distribution is matched. The • values chosen in this manner are 1.16 for the session length distribution and 1.10 for the think time distribution. Table 1 shows the minimum, maximum, and mean of observations obtained with the distributions for the synthetic workloads used in the study.

Table 1: Session length and think time statistics

		Empirical	**Bounded Pareto**
Session length (Requests/session)	Minimum	3	3
	Maximum	120	120
	Mean	9.44	9.44
Think time (s)	Minimum	0	12
	Maximum	900	900
	Mean	46.54	46.54

We consider three workload mixes with different levels of variability in request resource demands. Table 2 shows these mixes along with the mean "no-load" mean response times (R_{mean}) measured for each of the 14 TPC-W request types. The R_{mean} values are obtained when the number of concurrent sessions is set to one. Consequently, they reflect the end-to-end resource demands across all resources for request types for the TPC-W system. The TPC-W Shopping

mix [25] is used as a high demand variation mix (Hi-Mix) in this study. We also define a slightly different mix with *slightly lower variation in demand* (Med-Mix). To construct such a mix, we reduce the proportions of requests belonging to the top two resource intensive request types `Buy request` and `Buy confirm` and the non-resource intensive `Home` request type and cause a corresponding increase to the proportion of requests belonging to the `Product detail` request type, relative to the Hi-Mix. Finally, we also construct the Lo-Mix to reflect a mix that has a slightly lower mean demand and lower variation in demand than both the Hi-Mix and the Med-Mix. As shown in Table 2, this is achieved by eliminating certain resource intensive request types such as `Buy request` followed by a concomitant increase to the less resource intensive `Home` request type.

Table 2: Mean no-load response times of request types and workload mixes

	R_{mean}(s)	Hi-Mix	Med-Mix	Lo-Mix
`Home`	0.09	16.00%	**9.00%**	23.46%
`New products`	0.18	5.00%	5.00%	5.00%
`Best sellers`	0.18	5.00%	5.00%	5.00%
`Product detail`	0.23	17.00%	**27.80%**	17.00%
`Search request`	0.07	20.00%	20.00%	20.00%
`Search results`	0.13	17.00%	17.00%	17.00%
`Shopping cart`	0.24	11.60%	11.60%	11.60%
`Customer registration`	0.21	3.00%	3.00%	**0.00%**
`Buy request`	0.63	2.60%	**0.00%**	**0.00%**
`Buy confirm`	0.25	1.20%	**0.00%**	**0.00%**
`Order display`	0.18	0.66%	0.66%	**0.00%**
`Order inquiry`	0.05	0.75%	0.75%	0.75%
`Admin request`	0.09	0.10%	0.10%	0.10%
`Admin confirm`	0.14	0.09%	0.09%	0.09%
Mean R_{mean} (s)		0.16	0.16	0.14
COV of request response time		0.62	0.41	0.39

Table 2 also shows the mean no-load response time and the coefficient of variation of no-load request response time computed for the three mixes. These correspond closely to aggregate resource demand usage. It can be seen that our design causes the Med-Mix to have a lower coefficient of variation (COV) of request response time than the Hi-Mix while maintaining the same no-load R_{mean} for both mixes. Finally, both the no-load R_{mean} and the COV of request response time for the Lo-Mix are slightly lower than that for the other mixes.

To establish the robustness of our modeling technique, we conducted experiments with three different application settings Base, HighDiskU and BigDB. The Base setting corresponded to a TPC-W application configured with 1000 books in the database. For the workloads we studied with this setting, the Web server node CPUs were found to be the bottleneck. The HighDiskU setting differs from the Base setting in terms of database server configuration. Specifically, the

database server's main memory cache settings were modified to cause more database node disk I/Os for a given workload when compared to the Base setting. However, in spite of the increased I/Os, the Web server node CPUs were still the bottleneck for all the workloads explored for the HighDiskU setting. Finally, the BigDB setting corresponded to a TPC-W application with 100,000 books in the database. This configuration allowed us to verify the effectiveness of our approach when the bottleneck shifts from the Web server node CPUs to the database server node CPU. The HighDiskU and BigDB cases did not appear in our earlier work [21].

C. Experiment Methodology

Due to time constraints, we did not conduct a full-factorial investigation of the workload and application factors discussed in the previous section. Instead we used SWAT to create carefully controlled workloads designed to exhibit the performance impact of combinations of the factors considered. Table 3 lists the workloads that were created by SWAT. Each workload is described by four hyphen-separated tokens. The first token describes the session length and think time distribution of the workload. For each workload, the choice of distribution type, i.e., empirical or bounded Pareto, is always chosen to be the same for session length and think time distribution. BPSLZ indicates the use of the bounded Pareto distributions of Table 1 while EMPSLZ indicates the use of the empirical distributions of Table 1. The subsequent tokens describe the workload mix, the mean utilization of each processor in the Web/application server node (U_{WebCPU}) observed over the experiment duration, and the application settings, in that order.

From Table 3, eleven experiments are conducted for this study. Each experiment is designed to study the impact of a given workload. As shown in Table 3, several statistically independent replications are conducted for each experiment. To achieve this, SWAT is used with different random number generator seeds to create several session traces that are statistically identical with respect to the workload characteristics described in Section III.B. In each experiment replication 10,000 sessions are submitted to the TPC-W system. The duration of a replication varied from approximately 3 hours to 5 hours depending on the mean session inter-arrival time used. Each replication yielded around 95,000 response time observations. From Table 3, in total 38 experiment replications were conducted for this study.

Table 3: Response time and resource demand measurements from the case study

Workload	R_{mean} (s)	Mean R_{mean} (s)	$D_{Web,CPU}$ (ms)	$D_{Web,Disk}$ (ms)	$D_{DB,CPU}$ (ms)	$D_{DB,Disk}$ (ms)
BPSLZ-HiMix-77-HighDiskU	1.10	1.11	191.64	8.44	46.54	19.04
	0.93		190.65	8.49	46.80	18.94
	1.30		194.33	8.25	46.51	17.65
BPSLZ-HiMix-71-BigDB	2.02	2.09	189.53	8.95	110.76	6.67
	2.06		190.35	8.85	110.88	6.66
	1.63		189.54	9.38	111.42	7.08
	2.65		195.86	9.04	112.31	6.76
BPSLZ-HiMix-77-Base	1.03	1.06	191.02	8.27	39.11	5.48
	0.93		190.45	8.54	38.83	5.37
	1.22		193.57	8.13	38.95	5.36
EMPSLZ-HiMix-77-Base	0.85	0.94	189.45	8.42	39.57	5.86
	0.90		191.58	8.71	39.47	5.39
	0.97		191.15	8.43	39.60	5.76
	1.02		190.45	9.07	39.49	5.44
EMPSLZ-MedMix-77-Base	0.75	0.75	188.39	9.24	34.08	5.49
	0.75		191.57	8.52	36.80	5.60
	0.76		188.79	8.43	34.18	5.47
	0.74		186.59	8.41	34.20	5.49
BPSLZ-MedMix-77-Base	0.92	0.93	189.99	8.32	32.94	5.35
	0.86		190.17	8.14	33.97	5.52
	1.02		191.45	9.96	33.82	5.57
EMPSLZ-LoMix-77-Base	0.67	0.72	176.89	6.27	26.04	4.57
	0.79		179.09	6.61	25.95	4.84
	0.69		177.40	6.23	25.98	4.56
	0.71		177.18	6.41	25.96	4.91
BPSLZ-HiMix-71-Base	0.67	0.69	184.45	8.88	38.91	5.46
	0.70		185.72	9.10	39.07	5.57
	0.60		183.67	9.14	38.97	5.46
	0.78		186.19	9.35	39.05	5.55
EMPSLZ-MedMix-71-Base	0.56	0.55	183.09	9.33	36.26	5.32
	0.55		183.38	9.88	33.18	5.52
	0.57		183.89	8.97	33.99	5.44
	0.53		183.20	9.31	34.04	5.52
EMPSLZ-LoMix-71-Base	0.49	0.52	171.55	6.70	25.98	4.79
	0.52		172.11	6.84	25.94	4.96
	0.54		174.75	7.15	25.96	4.75
	0.52		173.59	8.10	26.06	4.72
EMPSLZ-MedMix-65-Base	0.43	0.44	178.09	10.97	34.06	5.70
	0.44		178.75	10.36	34.06	5.69

The following observations were consistent across all experiments. *httperf* provided highly reproducible results. When expected, multiple repetitions of an experiment replication yielded almost the same mean response time measures. Furthermore, there was very little difference between the achieved workload characteristics, as measured from *httperf* logs collected from experiment replications, and the specified workload characteristics. This verifies that the client node was not saturated in our study. The worst-case mean and peak network traffic during the experiments was only 0.40 Mbps and 0.83 Mbps, respectively. This is because the CPU intensive nature of HTTPS and application server processing limited request throughputs. The low network traffic indicates that the response time measured by *httperf* is likely to be dominated

by the delay encountered at the TPC-W system. The disks at both server nodes were very lightly utilized. Virtually no memory paging activity was observed at either server node. Finally, job flow balance was achieved for all experiments with the number of request completions equaling the number of request arrivals.

D. Overview of Results

Table 3 provides several sanity checks with regards to our experimentation. The table presents the average per-request demands in milliseconds placed on the CPUs and disk of the Web/Application server node, $D_{Web,CPU}$ and $D_{Web,Disk}$, respectively, and the database server node, $D_{DB,CPU}$ and $D_{DB,Disk}$, respectively. It also provides the mean response time of requests that were submitted in an experiment replication, R_{mean}, and the mean R_{mean} over all replications in an experiment. The following observations can be made from Table 3.

The demand values for an experiment's replications are always nearly identical. This confirms that statistically identical replications place similar demands on the system and that burstiness does not affect average demands.

For a given application setting, workloads with the same mix cause similar demands on system resources. This can for example be verified by comparing the demand measurements for the BPSLZ-HiMix-77-Base, EMPSLZ-HiMix-77-Base, and BPSLZ-HiMix-71-Base workloads.

The measurements show that the mixes chosen for the study behaved as intended. From Table 3, for a given application setting the MedMix workloads impose almost the same average demands on the system as the HiMix workloads (compare for example EMPSLZ-HiMix-77-Base and EMPSLZ-MedMix-77-Base). As expected, the LoMix workloads place slightly lower demands on the system than the HiMix and MedMix workloads.

The application settings explored also exhibited the intended behaviour. For example, the BPSLZ-HiMix-77-HighDiskU workload exerts more demand on the database server's CPU and disk when compared to the BPSLZ-HiMix-77-Base workload. Similarly, the database server CPU demand for the HTSLZ-HiMix-71-BigDB workload is significantly more than that of the HTSLZ-HiMix-71-Base workload.

Results pertaining to the Base application setting along with a detailed discussion can be found in our earlier publication [21]. We now briefly describe some of the salient findings of the results from a performance modeling perspective. As mentioned previously, we also present additional results pertaining to the other two new application settings.

Distributions that cause highly variable session lengths and think times can adversely impact system performance – This observation can be made by comparing the BPSLZ-HiMix-77-Base and EMPSLZ-HiMix-77-Base workloads in Table 3. These workloads only differ with respect to their session length and think time distributions. From Table 3, they place almost identical demands on the TPC-W system's resources. The CPUs and disks in the systems have similar utilizations for both workloads. However, from Table 3, the mean R_{mean} for the BPSLZ-HiMix-77-Base workload is about 13% higher than that of the EMPSLZ-HiMix-77-Base workload. Similarly, from Table 3, the mean R_{mean} for BPSLZ-MedMix-77-Base workload is about 24% higher than that of the EMPSLZ-MedMix-77-Base workload. These results suggest that the bounded Pareto session length and think time distributions are responsible for the performance degradation.

As mentioned in Section II, high variability in session lengths and think times impact performance since they can cause bursty request arrivals. Specifically, such distributions yield large numbers of very small and very large session length and think time values. Consequently, BPSLZ-like workloads will have larger numbers of very long duration and very short duration sessions than EMPSLZ-like workloads. As a result, for any given mean session inter-arrival time, the likelihood of observing very large and very small number of concurrent sessions is more with a BPSLZ workload than with a EMPSLZ workload. This is illustrated in Figure 3 which shows the cumulative distribution function (CDF) of number of concurrent sessions for BPSLZ-HiMix-77-Base and EMPSLZ-HiMix-77-Base workloads[1]. Since the number of requests that can arrive at the system is positively correlated with the number of concurrent sessions, this phenomenon causes a more uneven or bursty arrival of requests. This increase in burstiness can sometimes, as in our experiments, be significant enough to cause periods of heightened contention for system resources during which requests incur very long response times.

Mixes characterized by higher variability in request demands cause poorer performance – This conclusion can be verified from Table 3 by comparing the EMPSLZ-HiMix-77-Base and EMPSLZ-MedMix-77-Base workloads. Recalling from the previous sections, both these

[1] The CDF for a workload was obtained by combining data from all its experiment replications.

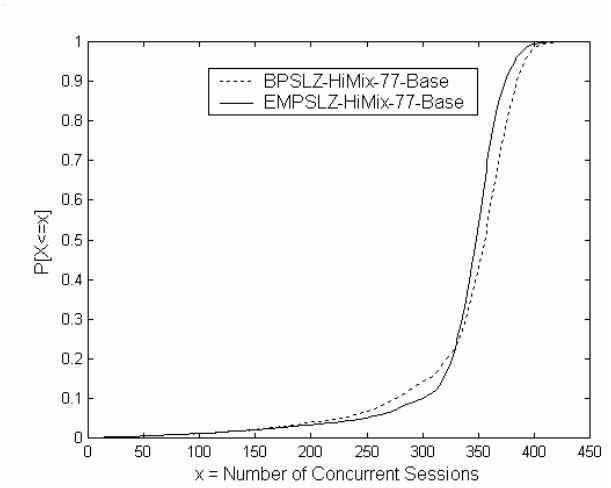

Figure 3: CDFs of number of concurrent sessions for BPSLZ-HiMix-77-Base and EMPSLZ-HiMix-77-Base

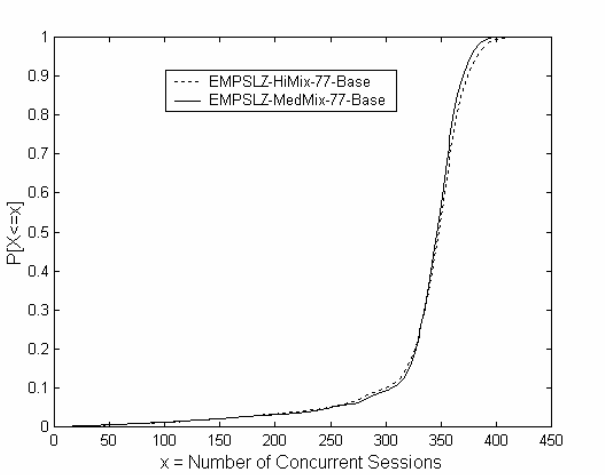

Figure 4: CDFs of number of concurrent sessions for EMPSLZ-HiMix-77-Base and EMPSLZ-MedMix-77-Base

workloads are similar in all respects except their workload mix. From Table 2, both workloads place the same mean aggregate demands on the system's resources. However, the HiMix workload is characterized by a slightly higher variability in request demands. Both workloads cause nearly identical utilizations of the CPUs and disks in the system. However, the mean R_{mean} for the EMPSLZ-HiMix-77-Base workload is about 25% higher than that of the EMPSLZ-MedMix-77 workload. Figure 4 plots the CDFs of number of concurrent sessions for the workloads. From Figure 4, it can be seen that the HiMix workload exhibits a slightly longer tail than the MedMix workload. The reason for this behaviour is again due to the increased variability of session durations; the larger proportions of resource intensive, e.g., Buy request, and non resource intensive, e.g., Home, requests within sessions of the HiMix workload increases the likelihood of very long duration and very short duration sessions. This leads to periods of increased contention among sessions leading to a higher mean R_{mean}.

Bursty workloads exhibit high variability in R_{mean} – As mentioned in Section II, workloads characterized by heavy-tailed distributions lead to unpredictability in system behaviour. This phenomenon can be observed for the BPSLZ-HiMix-71-BigDB workload. Recalling from the previous section, this workload caused the database server node CPU to be the bottleneck. The mean database server node CPU utilization over the duration of each replication was 84%. From

Table 3, the highest R_{mean} value of 2.65 seconds for this workload is 64% higher than the lowest R_{mean} value of 1.63 seconds. This is in spite of the fact that the experiment replications are statistically identical, cause near identical demands and utilizations on the system's resources,

and lasted for nearly 5 hours. Similar trends can be observed for the BPSLZ-HiMix-77-Base and BPSLZ-MedMix-77-Base workloads.

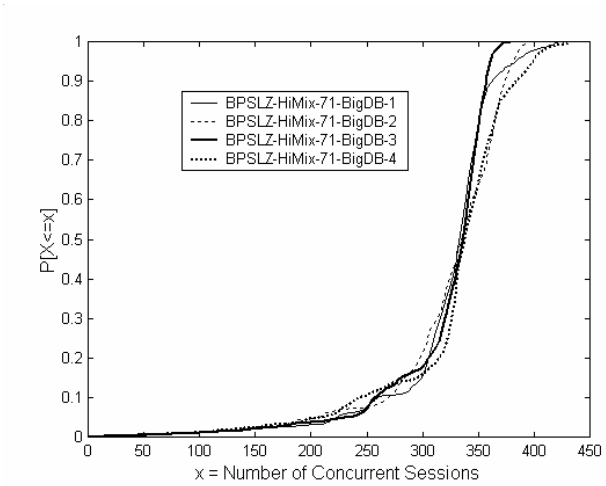

Figure 5: CDFs of number of concurrent sessions for BPSLZ-HiMix-71-BigDB

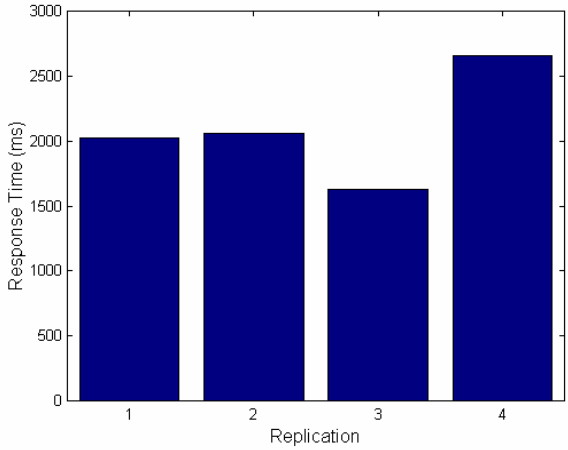

Figure 6: Measured R_{mean} values for BPSLZ-HiMix-71-BigDB

The reason for the variation in R_{mean} can again be explained in terms of the population distribution. Figure 5 plots the CDF of number of concurrent sessions for the four replications of the BPSLZ-HiMix-71-BigDB workload. Figure 6 plots the R_{mean} values for these replications. Figure 5 shows that the CDF is different for the different replications. In particular, replication 4's CDF exhibits the longest tail and results in the highest R_{mean} while replication 3's CDF has the shortest tail and causes the lowest R_{mean}. WAM can help performance analysts estimate the extent of variability in R_{mean} for bursty workloads by repeating the analysis multiple times with different session traces.

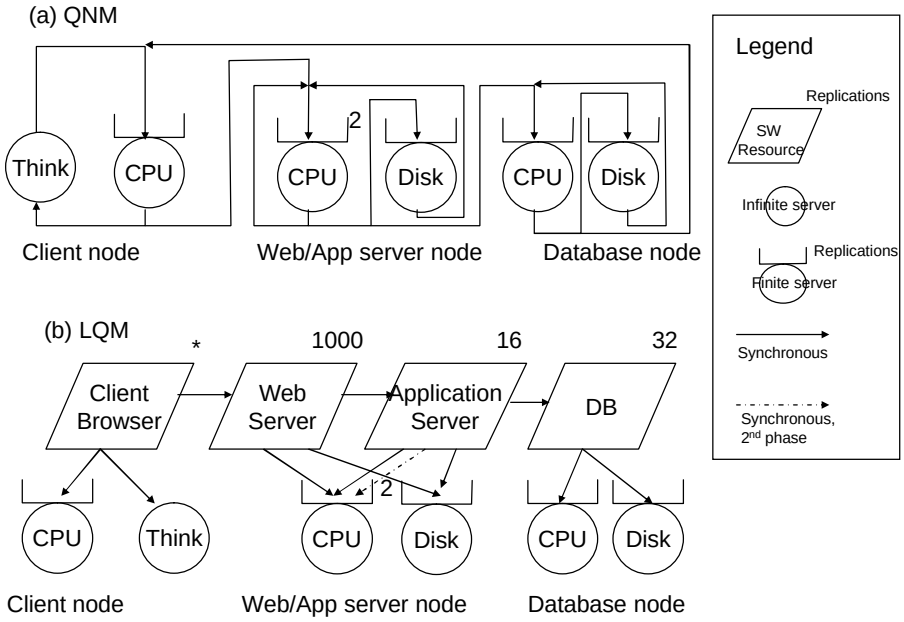

Figure 7: Predictive performance models for the TPC-W system

V. PREDICTIVE PERFORMANCE MODELS FOR THE MULTI-TIER SYSTEM

This section describes the QNM and LQM performance models developed for the TPC-W system described in Section IV. Section VI applies these models in combination with WAM to predict mean request response time for the cases considered in Table 3.

Figure 7 shows the two performance models. Both models take as input the average demands incurred by an HTTP request. Total per-request average CPU and disk demands are given in Table 3.

Figure 7(a) illustrates the QNM. It only includes a think time delay centre and queues for hardware resources, namely client node CPU, Web/application server node CPUs and disk, database server node CPU, and database server node disk. The value 2 shown to the upper right of the Web/App Node CPU indicates that the server has two CPUs. Other hardware resources were very lightly loaded so they were not included in the model. The number of customers corresponds to the number of concurrent sessions. Customers flow from queue to queue. After visiting a CPU, a customer may have one or more alternative queues to visit. Routing choices do not depend on the state of the system, are random, and have probabilities such that the desired

ratio of demands is incurred at the resources. A customer that flows from the client node CPU through to the database server node CPU and back to the client node CPU completes a HTTP request.

Figure 7(b) shows the LQM for the TPC-W system. LQMs are extended QNMs that include information about logical resources such as threading levels for application servers and software request-reply relationships. The LQM for the TPC-W system includes the same think time delay centre and hardware resources. The logical resources in the model are the client browsers, Web server threads, application server threads and database server threads. Threading levels other than one are shown by placing a value near the upper right hand side of an icon. In this model, we have blocking requests between software resources and between software resources and hardware resources.

From Figure 7(b), there is one *client browser* for each concurrent session using the system. A customer using a client browser may visit its node's CPU or may think. A HTTP request causes a blocking call to the *Web server*. If a Web server thread is available then the request is accepted. The thread uses some CPU resource from the Web /application server node CPUs and then makes a request to the *application server*. If an application server thread is available then the request is accepted. The application server thread uses some CPU resource from the Web /application server node CPUs and then makes a request to the *database server*. If a database server thread is available then the request is accepted. The thread uses some CPU and disk resource from the database server node and releases the calling thread. The released calling thread from the application server can then complete its *first phase* of work and release the calling thread from the Web server.

From Figure 7(b), after finishing its first phase and releasing the calling thread from the Web server the application server thread continues on to a *second phase* of service. The second phase of service keeps the application server thread busy so that it cannot service another calling thread. However at the same time the calling thread from the Web Server that was released after the first phase of service can complete its work and release the calling thread from the client browser. This completes an HTTP request. The reasons for modeling the request-reply relationship of the application server in this manner are discussed shortly.

During an HTTP request, if a thread is not available when a server is called, the calling thread blocks until a thread becomes available. Once a thread completes its work it is available to serve another caller. Such threading can lead to software queuing delays in addition to any contention for hardware resources that are incurred by active threads. The numbers of threads used for each tier in the model reflect the application settings as described in Section IV.

To obtain resource demand values, for each experiment replication we measured the CPU utilizations for the Web server threads, application server threads, and the database server threads. We also measured the CPU and disk utilizations for the Web/application server node and the database server node, the elapsed time of the run, and the number of request completions. This enables us to compute the average resource demand per request for the Web server threads, application server threads, database server threads, and for the Web/application server node and database server node as a whole. The aggregate demand values used in the models are given in Table 3. We note that there was a very small difference between the utilization of a node and the sum of the utilizations of software processes running on that node. We modeled this as background load in the LQM.

We note that the demand values per hardware resource are identical for the QNM and the LQM. Moreover, both models handle the dual Web/application server node CPUs by making use of residence time expressions developed for multiprocessor resources [8]. The only difference in the models relate to whether software interactions, i.e., threading and two phase processing, are reflected in the model or not.

Finally, we observed from measurement runs with one concurrent session that mean response times were often *lower* than the aggregate demand upon the hardware resources. This is an indication of two phases of processing at a server. We reflected this in the LQM by placing 25% of the application server thread demands in a second phase of service [7][8]. This modeling choice was found to produce good model predictions. For the application settings considered, the per session population level mean response time predictions from the LQM closely matched the corresponding per session population level mean response times observed from the measurements in Section IV.

VI. RESULTS OF QNMs AND LQMs WITH WAM

This section applies the QNM and LQM models of Section V with WAM to predict the mean response times for the experiments of Section IV. These results are further compared with the straightforward application of QNMs and LQMs and the hybrid Markov chain birth-death approach of Section II for session based systems.

Table 4 shows the four different modeling approaches that are explored for both QNMs and LQMs. The "MEAN" approaches ignore the distribution of number of concurrent sessions. They solve a predictive model for only one customer population, namely, the mean number of concurrent sessions observed during an experiment replication. The "MBD" methods use the Markov birth-death approach to estimate the population distribution and R_{mean}. The birth-death model used a constant, state-independent birth rate[2] that equals the mean session arrival rate observed during a measurement experiment replication. The "WAMEMP" methods predict R_{mean} for an experiment replication by using WAM in conjunction with the empirical population distribution as measured during the replication. It does not use the population distribution estimation technique illustrated in Figure 2. The "WAMMC" method uses the SWAT trace corresponding to an experiment replication and Monte Carlo simulation as per the algorithm in Figure 2 to estimate the measured population distribution. The estimated population distribution is used to compute R_{mean}.

In general all the methods yielded good throughput estimates. The absolute errors for throughput were within 2% for the WAMMC methods and the MBD methods. The accuracy of the MEAN methods was slightly poorer. The throughput estimates of the MEAN-LQM and MEAN-QNM methods were within 3.5% and 4.0% of measured values, respectively.

However, there are significant differences in prediction accuracy for R_{mean} across the different methods. Three different error metrics are used to characterize the R_{mean} prediction accuracy of the modeling approaches. The mean absolute error (*ABS_ERROR*) is defined as

[2] Models with state-dependent birth rates were also tried but their accuracy was poorer than the state-independent approach.

$$\text{ABS\_ERROR} = 100 * \frac{\sum_i |e_i|}{\sum_i y_i}$$ where e_i is the difference between the measured and predicted mean response time and y_i is the measured response time for the i^{th} replication in a set of replications. The maximum of the absolute *e* values, expressed as a percentage, calculated for a set of replications is denoted as the maximum absolute error (*MAX_ERROR*). The trend error (*TRND_ERROR*) is an indicator of the range of errors obtained with a modeling approach. It is defined as the difference between the largest *e* value and the smallest *e* value, expressed as a percentage, for a set of replications. Table 4 shows the error measures for models pertaining to the entire set of thirty nine replications described in Table 3. The table gives results for the `MEAN`, `MBD`, `WAMEMP`, and `WAMMC` cases.

Table 4: Accuracy of modeling approaches for predicting R_{mean} over all workloads

Modeling Approach	***ABS_ERROR* (%)**	***MAX_ ERROR* (%)**	***TRND_ERROR* (%)**
`MEAN-LQM`	15.20	32.37	42.50
`MEAN-QNM`	17.22	42.56	63.75
`MBD-LQM`	13.67	32.56	45.09
`MBD-QNM`	16.71	42.68	66.01
`WAMEMP-LQM`	5.79	15.50	28.23
`WAMEMP-QNM`	9.77	26.10	44.94
`WAMMC-LQM`	7.18	18.44	33.17
`WAMMC-QNM`	12.14	30.69	57.30

First we consider the `MEAN` cases. These are the only cases that do not take population distribution into account. From Table 4, the *ABS_ERROR* is lower for the `MEAN-LQM` approach than the `MEAN-QNM` approach. The `MEAN-LQM` approach also does better in terms of *MAX_ERROR* and *TRND_ERR*. The improved prediction accuracy is due to the LQM taking into account the performance impacts of finite server thread pools and two phases of application server processing. However, the *ABS_ERROR* of about 15% and the *MAX_ERROR* of about 32% are still quite large for the `MEAN-LQM` approach. These errors are large despite individual per session population level R_{mean} predictions from the LQM agreeing well with the corresponding measured values. This suggests there will be benefits from considering the population distribution.

We now consider the Markov Chain birth-death approach `MBD`. From Table 4, results show only slight improvements in *ABS_ERROR*. For example, the technique when used in conjunction with the LQM (`MBD-LQM`) achieves a reduction in *ABS_ERROR* of only about 1.5% when compared to `MEAN-LQM`.

WAM with the empirical population distribution from a historical trace with measured response times, WAMEMP, improves accuracy a great deal. From Table 4, the *ABS_ERROR* drops by nearly 10% with WAMEMP-LQM when compared to MEAN-LQM. Moreover, *MAX_ERROR* and *TRND_ERROR* drop by about 17% and 14%, respectively when compared to MEAN-LQM. Similar improvements are noticed when comparing WAMEMP-QNM and MEAN_QNM. This confirms the importance of population distribution.

Finally, we consider WAM with a population distribution estimated using the algorithm of Figure 2, WAMMC. The results from Table 4 show the effectiveness of the approach. The WAMMC methods performed nearly as well as their corresponding WAMEMP methods. For example, from Table 4 the error metrics for WAMMC-LQM are very similar to that of WAMEMP-LQM. However, WAMMC has an advantage over WAMEMP. It allows the WAM method to be applied in a constructive manner to predict the performance of systems when varying workload parameters and when a historical trace with measured response times is simply not available.

The WAMMC results validate the population distribution estimator's use of the R_{mean} prediction from a predictive model for the current population level as an approximation of the response time of an individual request. We suggest that the approach works well for these cases because the think times encountered in the synthetic workloads used for the study are much longer, on the order of tens of seconds, than the response times which are on the order of hundreds of milliseconds or seconds. As a result the population distribution is governed more by the session length, think time, and session inter-arrival time distributions than the response time distribution for each population level. We note than an analysis of the empirical think time distribution of Table 1 indicates that the assumption of think times being much longer than response times is likely to be valid for real session-based workloads.

Table 5: Accuracy of modeling approaches for predicting R_{mean} for bursty workloads

Modeling Approach	***ABS_ERROR* (%)**	***MAX_ERROR* (%)**	***TRND_ERROR* (%)**
MEAN-LQM	21.20	32.37	28.03
MEAN-QNM	23.88	42.56	45.11
MBD-LQM	19.24	32.56	30.06
MBD-QNM	22.55	42.68	46.79
WAMEMP-LQM	4.93	13.80	25.53
WAMEMP-QNM	9.03	18.84	29.40
WAMMC-LQM	7.06	18.44	31.17
WAMMC-QNM	12.43	30.69	41.57

We now consider several subsets of the results in more detail. Results are discussed for the following cases: bursty workloads; higher and lower contention for the bottleneck; higher and

medium coefficients of variation for request resource demands; non-bursty workloads; and, workloads with heavy-tail-like distributions. Finally, a case is presented that demonstrates the constructive capability of WAMMC.

The WAM approach is particularly effective for bursty workloads - Table 5 summarizes the error measures for only those seventeen experiment replications that employed the bounded Pareto session length and think time distributions of Table 1. For bursty workloads using just the mean population provides very poor R_{mean} estimates. From Table 4, the *ABS_ERROR* is nearly 21% for the MEAN-LQM approach. From Table 4 and Table 5, the MEAN-LQM approach applied to these workloads results in 6% greater *ABS_ERROR* than overall for all workloads. For these workloads the WAM method results in a greater reduction in *ABS_ERROR* than overall for all workloads. For example, from Table 4 the *ABS_ERROR* for WAMEMP-LQM is about 16% lower than that for MEAN-LQM. This represents about 6% more reduction in error than when considering all the workloads. A similar trend can be noticed with WAMEMP-QNM. The WAMMC-LQM and WAMMC-QNM methods result in slightly increased *ABS_ERROR* when compared to their counterparts that use the empirically measured population distribution. However, the errors are still significantly less than those obtained with the MEAN and MBD methods.

Table 6: Comparison of gains from WAM for two different bottleneck device utilizations

BPSLZ-HiMix-77			
Modeling Approach	***ABS_ERROR* (%)**	***MAX_ERROR* (%)**	***TRND_ERROR* (%)**
MBD-LQM	21.59	27.99	13.91
WAMEMP-LQM	5.80	10.59	8.47
WAMMC-LQM	5.90	9.90	10.48
BPSLZ-HiMix-71			
Modeling Approach	***ABS_ERROR* (%)**	***MAX_ERROR* (%)**	***TRND_ERROR* (%)**
MBD-LQM	12.94	21.24	18.73
WAMEMP-LQM	6.48	11.74	16.61
WAMMC-LQM	6.97	12.72	17.73

The gains from WAM are significant when there is higher contention for the bottleneck resource - To illustrate this effect Table 6 compares the error metrics for the BPSLZ-HiMix-77 and BPSLZ-HiMix-71 replications. For the sake of clarity results are shown only for the MBD-LQM, WAMEMP-LQM, and WAMMC-LQM methods. From the table, when $U_{Web,CPU}$ is 71% WAM results in an improvement of about 6%, 9%, and 1% in *ABS_ERROR*, *MAX_ERROR*, and *TRND_ERROR*, respectively. These numbers increase to 15.5%, 18%, and 3.5% when $U_{Web,CPU}$ is 77%. Previous studies have shown that the burstiness induced by heavy-tails becomes more pronounced at higher utilizations [33]. Consequently, the BPSLZ-HiMix-77 workload is more

bursty than the BPSLZ-HiMix-71 workload. The increased gain in accuracy for the BPSLZ-HiMix-77 workload provides further evidence that WAM is very effective for predicting the behaviour of bursty workloads.

WAM is particularly effective for mixes characterized by higher variability in request resource demands -Table 7 compares the MBD-LQM and WAM methods for the BPSLZ-HiMix-77 and BPSLZ-MedMix-77 workloads. Recall from section IV that the HiMix workload exhibits more variability in demands than the MedMix workload since it has a greater percentage of resource intensive and resource non-intensive requests. From Table 7, the MBD-LQM method results in an *ABS_ERROR* of 13.55% and a *MAX_ERROR* of 17.33% for the MedMix workload. The method performs even poorer for the HiMix workload with the *ABS_ERROR* and *MAX_ERROR* increasing to 21.59% and 27.99%, respectively. From Table 7, the WAM methods are significantly more accurate than MBD-LQM for both workloads. The gains in *ABS_ERROR* while using the WAMMC-LQM method over the MBD-LQM method are nearly 11.5% for the MedMix workload and 15.5% for the HiMix workload.

Table 7: Comparison of gains from WAM for HiMix and MedMix workloads

BPSLZ-HiMix-77			
Modeling Approach	***ABS_ERROR* (%)**	***MAX_ERROR* (%)**	***TRND_ERROR* (%)**
MBD-LQM	21.59	27.99	13.91
WAMEMP-LQM	5.80	10.59	8.47
WAMMC-LQM	5.90	9.90	10.48
BPSLZ-MedMix-77			
Modeling Approach	***ABS_ERROR* (%)**	***MAX_ERROR* (%)**	***TRND_ERROR* (%)**
MBD-LQM	13.55	17.33	8.54
WAMEMP-LQM	1.98	3.49	4.33
WAMMC-LQM	1.90	3.48	5.31

WAM is effective for cases with non-bursty workloads -Table 8 summarizes the error measures for those experiment replications that did not use the bounded Pareto session length and think time distributions. From Table 8 and Table 4, the MEAN-LQM and MBD-LQM approaches have a much lower *ABS_ERROR* for these workloads than overall for all workloads. The *ABS_ERROR* for WAMEMP-LQM is comparable to those of MEAN-LQM and MBD-LQM. However, WAMEMP-LQM method results in a smaller range of errors when compared to the other two approaches. For example, the *MAX_ERROR* and *TRND_ERROR* for WAMEMP-LQM are about 9.5% and 7.5% lower, respectively than those of MEAN-LQM. Furthermore, the WAMMC methods result in almost similar errors to those of their counterpart WAMEMP methods. These results show that the WAM technique can provide more robust performance estimates than the other approaches and is suitable for both bursty as well as non-bursty workloads.

Table 8: Accuracy of modeling approaches for predicting R_{mean} for non-bursty workloads

Modeling Approach	***ABS_ERROR* (%)**	***MAX_ERROR* (%)**	***TRND_ERROR* (%)**
MEAN-LQM	6.91	24.92	35.05
MEAN-QNM	8.03	22.32	43.51
MBD-LQM	5.99	23.17	35.70
MBD-QNM	8.66	23.32	44.34
WAMEMP-LQM	6.98	15.50	28.23
WAMEMP-QNM	10.78	26.10	38.73
WAMMC-LQM	7.33	15.79	30.52
WAMMC-QNM	11.75	26.61	40.73

WAM captures the complex effects of heavy-tail-like distributions - Figures 8 to 11 show the probability distribution function (PDF) of number of concurrent sessions for the BPSLZ-HiMix-71-BigDB experiment replications estimated using WAMMC-LQM and MBD-LQM. Figure 12 compares the measured R_{mean} values for this workload with those predicted using WAMMC-LQM and MBD-LQM. As discussed in Section IV, the R_{mean} values measured for this case varied by up to a factor of 1.63 even though the measured demands and device utilizations were nearly identical for all the replications.

Figures 8 to 11 reveal that the MBD-LQM method does not capture the differences in measured PDFs among the replications. The PDFs estimated by MBD-LQM are nearly identical for all the replications. In contrast, WAMMC-LQM closely tracks the changes in PDFs. The PDFs estimated through simulation are very close to their counterpart measured PDFs. Consequently, from Figure 12 the R_{mean} values predicted by MBD-LQM are nearly the same for all the replications. In contrast, the R_{mean} values predicted by WAMMC-LQM closely track the measured R_{mean} values. From Figure 12, MEAN-LQM also suffers from the same limitation as MBD-LQM and predicts almost the same R_{mean} for all the replications. We note that for non-bursty workloads there is less concentration of probability mass towards very large and very small populations. As a result the accuracy obtained with the other methods starts to approach that obtained with WAM.

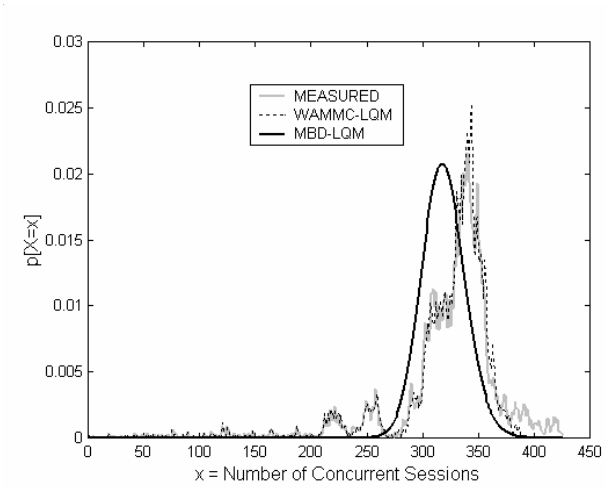

Figure 8: PDF of number of concurrent sessions for BPSLZ-HiMix-71-BigDB-1

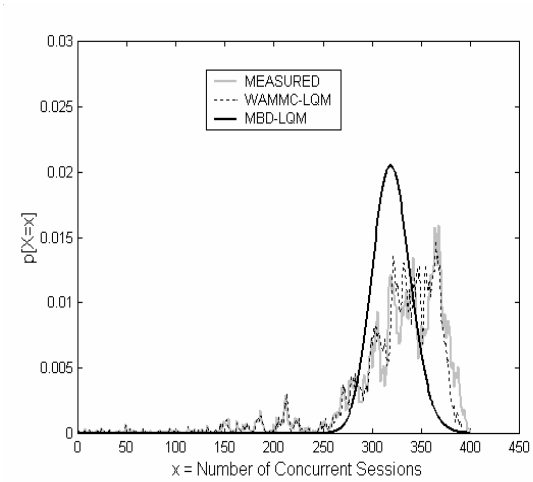

Figure 9: PDF of number of concurrent sessions for BPSLZ-HiMix-71-BigDB-2

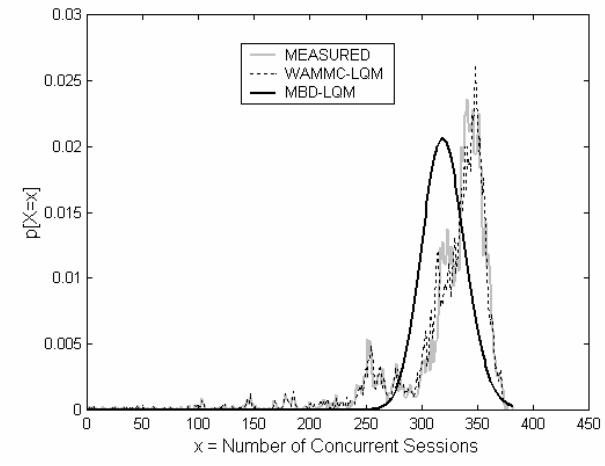

Figure 10: PDF of number of concurrent sessions for BPSLZ-HiMix-71-BigDB-3

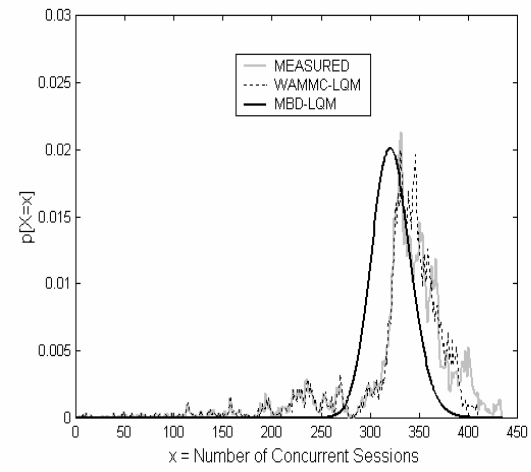

Figure 11: PDF of number of concurrent sessions for BPSLZ-HiMix-71-BigDB-4

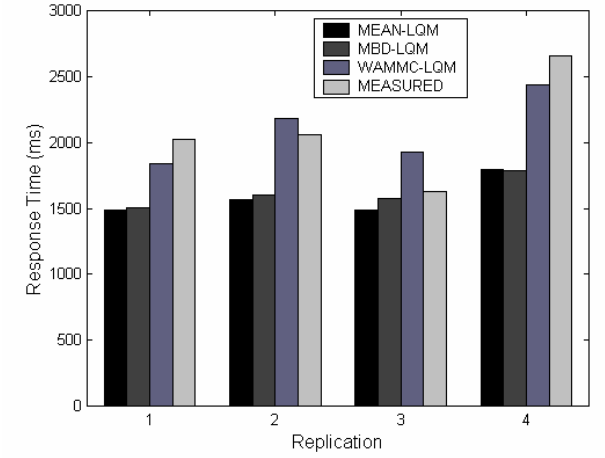

Figure 12: Predicted and measured R_{mean} values for BPSLZ-HiMix-71-BigDB

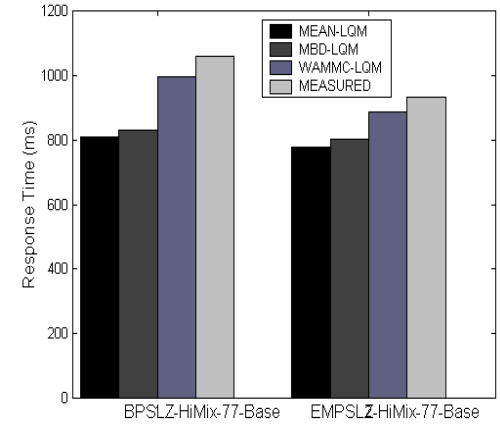

Figure 13: Predicted and measured mean R_{mean} values for BPSLZ-HiMix-77-Base and EMPSLZ-HiMix-77-Base

Another consequence of the ability to accurately estimate the population distribution is that WAM can assess the predictability of performance. As shown in Figure 12, WAM is the only method able to capture the variation in measured R_{mean} values for the statistically identical replications of the BPSLZ-HiMix-71-BigDB workload. The results are similar for the other bursty cases, though less pronounced.

Finally, we show that WAM is better suited for predicting the impact on system performance of changes in workload characteristics than the other methods. Figure 13 plots the mean of measured R_{mean} values over all replications for both the BPSLZ-HiMix-77 and EMPSLZ-HiMix-77 workloads. It also shows the mean of the predicted R_{mean} values over all replications for both workloads while employing the MEAN-LQM, MBD-LQM, WAMEMP-LQM, and WAMMC-LQM methods. From the figure, WAM is able to capture the increase in the measured mean R_{mean} that is caused by increased heavy-tail behaviour for session lengths and think times in the BPSLZ-HiMix-77 workload. While the measured increase is approximately 125 ms the increase predicted by WAMMC-LQM is about 110 ms. In contrast, the MEAN-LQM and MBD-LQM methods do not reflect the impact of the changes and offer almost identical results for both workloads.

VII SUMMARY AND CONCLUSIONS

In this paper, we introduce a new technique called the Weighted Average Method (WAM) for improving the accuracy of predictive models for systems with bursty customer populations. The technique is appropriate for session-based systems such as e-commerce systems and enterprise application systems. Others have shown that real session based systems exhibit such bursty behaviours so sizing, capacity planning, and on-going management exercises should benefit from WAM.

The technique was motivated by the well-known hybrid method that combines a Markov birth-death process and QNMs. We apply the general approach but replace the closed expression for estimating population distribution with a fast Monte Carlo simulation technique that lets us take into account arbitrary distributions that affect burstiness for request arrivals. Furthermore, we consider both QNMs and LQMs. A measurement based study for a TPC-W system permits us to compare the effectiveness of all these methods at predicting the mean request response time for

the TPC-W system. The system is subjected to both bursty and non-bursty workloads including workloads with heavy-tail-like distributions.

The results indicate that modeling approaches that only consider the mean number of concurrent customers produce very poor estimates of mean response time for systems with bursty workloads. The average prediction error for bursty workloads is nearly 24% and 21% for the QNM, and the LQM, respectively. Furthermore, for bursty workloads, using the QNM and LQM models in combination with a Markov birth-death model does not improve prediction accuracy significantly. In contrast, the WAM approach significantly improves the accuracy of mean response time predictions. For bursty workloads, prediction accuracy improved by 12% and 10% for LQMs and QNMs, respectively, as compared to the Markov birth-death approach. Moreover, the LQM-based WAM approach had much lower average error and range of errors than the QNM-based WAM approach. Furthermore, WAM also enabled the prediction of very different mean response times reported by multiple statistically identical runs for cases that include heavy-tail-like distributions. In effect, WAM can be used to assess whether a system has unpredictable behaviour by reporting a range of possible behaviours.

Others using MVA based techniques for modeling multi-tier session based systems [12][13][14] have not been considering burstiness that is inherent in these systems [18] [19]. To the best of our knowledge these are the first results that accurately predict mean request response times for such complex systems with bursty behaviour in such a straightforward way. The accuracy of WAM's predictions for the system studied is due both to WAM's approach for estimating customer session population distribution and the benefits obtained from using LQMs rather than QNMs.

The results we present likely benefit from the relatively large think times between requests. The think times were on the order of 46 seconds with response times typically less than a second. However, the think times chosen were realistic since they were based on empirical measurements from a large e-commerce site [23].

Our future work includes extending the technique to consider multi-class models, load dependent service rates, and embedded requests for images. We also intend to apply WAM for studying a time varying customer session arrival process and "flash crowd" scenarios. Techniques will be

developed to ensure the efficiency of WAM, in particular for multi-class models. Future work will apply and validate these techniques for other multi-tier software systems, including enterprise application systems.

References

[1] J.P. Buzen, "Computation algorithms for closed queuing networks with exponential servers," *Communications of the ACM*, vol. 16, no. 9, pp. 527-531, September 1973.

[2] M. Reiser, "A queuing network analysis of computer communication networks with window flow control," *IEEE Transactions on Communications*, vol. 27, no. 8, pp. 1201-1209, August 1979.

[3] D. Menasce and V. Almeida, "Capacity planning for Web services: metrics, models and methods," Prentice Hall Inc., September 2001.

[4] L. Kleinrock, "Queuing systems volume 1: theory," John Wiley & Sons Inc., 1975.

[5] W. H. Sanders, W. D. Oball II , M. A. Qureshi , and F. K. Widjanarko, "The UltraSAN modeling environment," *Performance Evaluation*, vol. 24, no.1-2, pp.89-115, November 1995.

[6] M. K. Molloy, "Performance analysis using stochastic Petri nets," *IEEE Transactions on Computers*, vol. 31, no. 9, pp. 913-917, 1982.

[7] M. Woodside, J. E. Nielsen, D. C. Petriu, and S. Majumdar, "The stochastic rendezvous network model for performance of synchronous client-server-like distributed software," *IEEE Transactions on Computers*, vol. 44, no. 1, pp, 20-34, January 1995.

[8] J.A. Rolia and K.C. Sevcik, "The method of layers," *IEEE Transactions on Software Engineering*, vol. 21, no. 8, pp. 689-700, August 1995.

[9] K. Psounis , P. Molinero-Fernández, B. Prabhakar and F. Papadopoulos, "Systems with multiple servers under heavy-tailed workloads," *Performance Evaluation*, vol. 62, no. 1-4, pp, 456-474, October 2005.

[10] K.M. Chandy and D. Nuese, "Linearizer: A heuristic algorithm for queuing network models of computer systems," *Communications of the ACM*, vol. 25, no. 2, pp. 126-133, February 1982.

[11] G. Casale, "An efficient algorithm for the exact analysis of multiclass queuing networks with large population sizes," *Proceedings of Joint ACM SIGMETRICS/Performance Conference*, pp. 169-180, 2006.

[12] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "Analytic modeling of multitier Internet applications," *ACM Transactions on the Web*, vol. 1, no. 1, article 2, May 2007.

[13] Y. Chen, S. Iyer, X. Liu, D. Milojicic, and A. Sahai, "SLA Decomposition: Translating service level objectives to system level thresholds," *Hewlett Packard Labs Technical Report*, 2007.

[14] Q. Zhang, L. Cherkasova, and E. Smirni, "A regression-based analytic model for dynamic resource provisioning of multi-tier applications," *4th International Conference on Autonomic Computing*, pp. 27-27, June 2007.

[15] N. Tiwari and P. Mynampati, "Experiences of using LQN and QPN tools for performance modeling of a J2EE application," *International Computer Measurement Group (CMG) Conference*, pp. 537-548, 2006.

[16] S. Kounev and A. Buchmann, "Performance modeling of distributed e-business applications using queuing Petri Nets", *IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 145-153, March 2003.

[17] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni, "Model-based performance prediction in software development: A survey," *IEEE Transactions on Software Engineering*, vol. 30, no. 5, pp, 295-310, May 2004.

[18] U. Vallamsetty, K. Kant, and P. Mohapatra, "Characterization of e-commerce traffic," *Electronic Commerce Research*, vol. 3, no. 1-2, pp. 167-192, 2003.

[19] D. Menasce, V. Almeida, R. Reidi, F. Pelegrinelli, R. Fonesca, and W. Meira Jr., "In search of invariants in e-business workloads," *ACM Conference on Electronic Commerce*, pp. 56-65, October 2000.

[20] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1-15, 1995.

[21] D. Krishnamurthy, J. Rolia, and S. Majumdar, "A synthetic workload generation technique for stress testing session-based systems," *IEEE Transactions on Software Engineering*, vol. 32, no. 11, November 2006.

[22] V. Almeida, M. Arlitt, and J. Rolia, "Analyzing a web-based system's performance measures at multiple time scales," ACM SIGMETRICS Performance Evaluation Review, vol. 30, no. 2, pp. 3-9, September 2002.

[23] M. Arlitt, D. Krishnamurthy, and J. Rolia, "Characterizing the scalability of a large web-based shopping system," *ACM Transactions on Internet Technology*, vol. 1, no. 1, pp. 44-69, 2001.

[24] M. E. Crovella and L. Lipsky, "Long-lasting transient conditions in simulations with heavy-tailed workloads,"

Proceedings of the Winter Simulation Conference, pp.1005 – 1012, 1997.
[25] TPC-W benchmark, http://www.tpc.org/tpcw/default.asp
[26] D. Menasce and M. Bennani, "Analytic performance models for single class and multiple class multithreaded software servers," *Proceedings of the International Computer Measurement Group (CMG) Conference*, pp. 475-482, 2006.
[27] R. Jain, "The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling," John Wiley & Sons Inc., April 1991.
[28] D. Menasce and V. Almeida, "Capacity planning and performance modeling: From mainframes to client-server systems," Prentice Hall, 1994.
[29] D. Krishnamurthy, "Synthetic workload generation for stress testing session-based systems," PhD thesis, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, January 2004.
[30] D. Mosberger and T. Jin, "httperf - a tool for measuring Web server performance," *Workshop on Internet Server Performance*, pp. 59-67, 1998.
[31] V.Paxon and S. Floyd, "Wide area traffic: The failure of Poisson modeling," *IEEE/ACM Transactions on Networking*, vol. 3, no. 3, pp. 226-244, 1995.
[32] M. Harchol-Balter, M. Crovella, and C. Murta. "On choosing a task assignment policy for a distributed server system," *Journal of Parallel and Distributed Computing*, vol. 59, no. 2, pp. 204-228, November 1999.
[33] K. Park, G.T. Kim, and M. Crovella, "On the relationship between file sizes, transport protocols, and self-similar network traffic," *Proceedings of the International Conference of Network Protocols*, pp. 171-180, October 1996.