A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications

Qi Zhang
College of William and Mary
Williamsburg, VA 23187, USA

Ludmila Cherkasova
Hewlett-Packard Labs
Palo Alto, CA 94304, USA

Evgenia Smirni
College of William and Mary
Williamsburg, VA 23187, USA

***Abstract*— The multi-tier implementation has become the industry standard for developing scalable client-server enterprise applications. Since these applications are performance sensitive, effective models for dynamic resource provisioning and for delivering quality of service to these applications become critical. Workloads in such environments are characterized by client sessions of interdependent requests with changing transaction mix and load over time, making model adaptivity to the observed workload changes a critical requirement for model effectiveness. In this work, we apply a *regression-based* approximation of the CPU demand of client transactions on a given hardware. Then we use this approximation in an analytic model of a simple network of queues, each queue representing a tier, and show the approximation's effectiveness for modeling diverse workloads with a changing transaction mix over time. Using the TPC-W benchmark and its three different transaction mixes we investigate factors that impact the efficiency and accuracy of the proposed performance prediction models. Experimental results show that this *regression-based* approach provides a simple and powerful solution for efficient capacity planning and resource provisioning of multi-tier applications under changing workload conditions.**

I. Introduction

Effective models of complex enterprise systems are central to capacity planning and resource provisioning. As multi-tiered architectures are now established as the industry standard that allows for integration of new, agile web applications with legacy (e.g., database) systems, the need for effective models of such systems becomes prevalent. Self-adaptive resource provisioning in such systems requires swift responses to workload changes. The need of fast response necessitates the use of analytic models that can quickly supply performance numbers, which then can drive system provisioning. In Next Generation Data Centers (NGDC) [7], where server virtualization provides the ability to slice larger, underutilized physical servers into smaller, virtual ones, fast and accurate performance models become instrumental for enabling applications to automatically request necessary resources and support design of utility services.

Our thesis is that effective analytic models can enable powerful and simple solutions for dynamic resource provisioning. The need for swift changes and timely performance predictions argues against the use of traditional simulation models and is in part responsible for the revival of classic analytic techniques for performance prediction that are based on simplified queueing networks [16], [17], [15]. The advantage of analytic models relates to their ability of providing a contained abstraction of the system by considering flows of customers (requests) in the queueing network (tiers). The effectiveness of the modeling ability of the queuing network relates to whether this abstraction is done properly. If salient characteristics of the system workload are captured well within the abstraction, then simple queueing network models can be effective in predicting the performance of complex systems. Naturally, more detailed workload models that capture multi-class behavior (i.e., the resource demands of different classes of customers) can be more effective than single class workloads where different user behaviors are aggregated into a single one.

A further challenge is the sensitivity of analytic models to their parameterization. Measurements in real systems cannot provide accurate workload "demands" (i.e., execution times without *any* delays due to queuing) in each tier/server (i.e., queue). Approximate workload demands are extrapolated using measurements at very low utilization levels or at nearly 100% utilization [16]. Variability across different customer behaviors further exacerbates the problem by requiring measurements of a large number of flows to accurately model the workload. An additional point relates to the fact that the workload is session-based rather than transaction-based. Each user session consists of an assortment of transactions, which in turn consist of processing many smaller objects and database queries. Consequently, detailed measurements, although necessary to increase model accuracy, become totally impractical.

In this work, we provide a practical solution to the above problems by laying out a theoretical framework which illustrates how to use information at the transaction level to effectively model session-based workloads. The effectiveness of the proposed framework is based on a *regression-based* methodology to approximate CPU demands of transactions on a given hardware. This regression-based solution can "absorb" some level of *uncertainty* or *noise* present in real-world data by effectively "compacting" information on workload demands within a few model parameters only. An additional benefit is simplicity: the methodology is not intrusive and is based on monitoring data that are typically available in enterprise production environments.

We illustrate the effectiveness of the methodology via a detailed set of experimentation using the TPC-W e-commerce suite [14]. We present sensitivity analysis of the

This work was largely completed in the summer of 2006 when Qi Zhang did an internship at HPLabs. Q. Zhang and E. Smirni are supported in part by the National Science Foundation (ITR-0428330). Currently, Qi Zhang is employed by MicroSoft and can be reached at the following address: qizha@microsoft.com.

proposed modeling approach with respect to the regression window as well as with respect to *a continuously* changing workload and transaction mix over time, that essentially behaves like a "live" system. Our experiments show that for the majority of cases, the model is in excellent agreement with experimental data. Even for the more challenging case where there is a continuous bottleneck switch, errors remain contained within 15%, providing a close answer to the fundamental problem of how many simultaneous sessions can be concurrently supported by the system.

This paper is organized as follows. The experimental testbed is presented in Section II. Section IV shows how one can compress a session-based workload with a transaction-based one. The statistical regression-based approach is presented in Section V. Section VI gives the simple queueing network model that is used to model TPC-W under changing workloads. Approach limitations are discussed in VII and related work is given in VIII. Conclusions and future work are outlined in Section IX.

II. Experimental Environment

We built a test-bed of a multi-tier application using the *three-tier architecture* paradigm that has become the industry standard for implementing scalable client-server applications. This allows to conduct experiments under different settings in a controlled environment, which then allows to evaluate the proposed modeling approach.

In our experiments, we use a testbed of a multi-tier e-commerce site that simulates the operation of an on-line bookstore, according to the classic TPC-W benchmark [14]. A high-level overview of the experimental set-up is illustrated in Figure 1, and specifics of the software/hardware used are given in Table I.

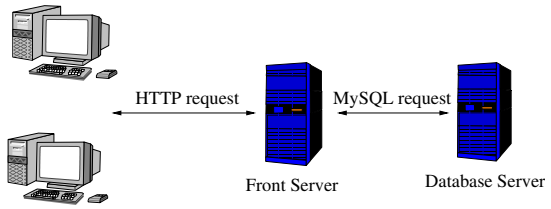


Fig. 1. E-commerce experimental environment.

TABLE I
TESTBED COMPONENTS

	Processor	RAM
Clients (Emulated-Browsers)	Pentium III / 1 GHz	2 GB
Front Server - Apache2.0/Tomcat4.0	Pentium III / 1 GHz	3 GB
Database Server - MySQL4.0	Pentium III/ 1 GHz	3 GB

Typically, a client access to a web service occurs in the form of a *session* consisting of a sequence of consecutive individual requests. In an e-commerce site, placing an order through the web site involves selecting a product, providing shipping information, arranging payment agreement, and finally receiving a confirmation. Thus, for a customer trying to place an order, or a retailer trying to make a sale, the real performance measure of such a web service is its ability to process the entire sequence of individual transactions needed to complete a higher-level business transaction. The capacity of the system is measured by the number of such concurrent client sessions that a multi-tier system can support without violating pre-defined limits in average transaction response times. Therefore, the workload of e-commerce and enterprise sites is best described at the level of sessions.

According to the TPC-W specification, the number of concurrent sessions (i.e., customers) or emulated browsers (EBs) is kept constant throughout the experiment. For each EB, the TPC-W benchmark statistically defines the user session length, the user think time, and the queries that are generated by the session. To better simulate the behavior of a real system, there is a time-out period (uniformly distributed between 5 and 15 minutes) that is associated with each EB. If a time-out occurs, then the session ends and a new session starts immediately. The database size is determined by the number of items and the number of customers. In our experiments, we use the default database setting, i.e., the one with 10,000 items and 1,440,000 customers.

TPC-W defines 14 different transactions which are roughly classified as either of browsing or ordering types as shown in Table II:

TABLE II
BASIC 14 TRANSACTIONS AND THEIR TYPES IN TPC-W

Browsing Type	**Ordering** Type
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

One way to capture the navigation pattern within a session is through the *Customer Behavior Model Graph (CMBG)* [9], which describes patterns of user behavior, i.e., how users can navigate through the site, and where arcs connecting states (transactions) reflect the probability of the next transaction type. TPC-W defines the set of probabilities that drive user behavior from one state to another at the user session level.

According to the weight of each type of activity in a given traffic mix, TPC-W defines 3 types of traffic mixes as follows:

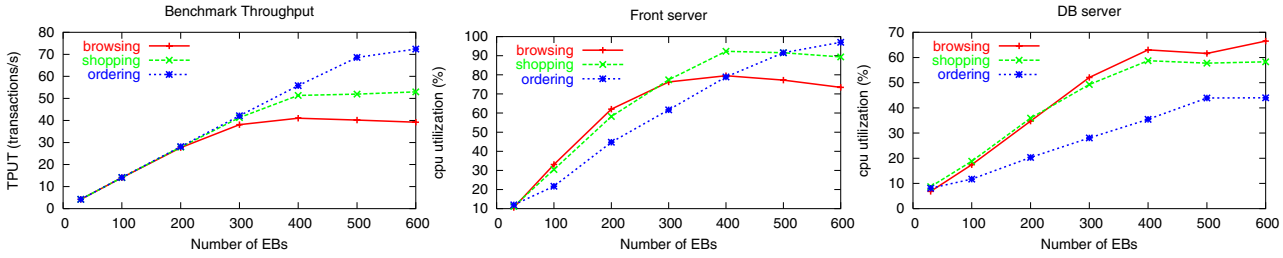
- the *browsing mix*: 95% browsing and 5% ordering;
- the *shopping mix*: 80% browsing and 20% ordering;
- the *ordering mix*: 50% browsing and 50% ordering.

Table III gives the 5 most popular transaction types of each workload mix.

TABLE III
TOP 5 TRANSACTION TYPES OF EACH WORKLOAD MIX

Browsing Mix	**Shopping** Mix	**Ordering** Mix
Home 29%	Search Request 20%	Search Request 15%
Product Detail 21%	Product Detail 17%	Shopping Cart 14%
Search Request 12%	Search Results 17%	Search Results 13%
New Products 11%	Home 16%	Customer Reg. 13%
Best Sellers 11%	Shopping Cart 12%	Buy Request 13%

For each workload mix, we ran a set of experiments with the number of EBs equal to 30, 100, 200, 300, 400, 500, and 600 respectively. Each experiment ran for 5 hours. The first 20 minutes and the last 20 minutes are considered as



(a) TPC-W Throughput (b) Front Server (c) DB Server

Fig. 2. Three TPC-W Transaction Mixes: a) Throughput; b) Average CPU Utilization of Front Server, and c) Average CPU Utilization of DB Server.

warm-up and cool-down periods, thus omitted in our analysis. Figure 2 presents a summary of these experiments. Figure 2(a) shows that the system becomes overloaded with 300 EBs, 400 EBs, and 500 EBs under the browsing mix, shopping mix and ordering mix, respectively. System throughput asymptotically flattens with higher EBs due to the effect of a closed-loop system, i.e., there is a constant number of EBs (customers) that circulate in the system at all times. Figures 2 (b) and (c) show the average CPU utilization at front and database servers respectively under the three workloads. From these results it is apparent that the front server is a bottleneck when the system is processing shopping and ordering transaction mixes (CPU utilization of the front tier is reaching 90-98%, while CPU utilization of the database tier is under 40-60%). However, for the browsing mix under high loads it is not obvious which tier and resource is responsible for the bottleneck: the average CPU utilization of the front and database tiers reaches 65-70%. It is not uncommon especially under bursty workload conditions [18] for the system to become overloaded. Here, average utilizations of both front and database servers are within the 65-80% range and additional performance measurements show that I/O (either at the disk or network) is not the system bottleneck either.

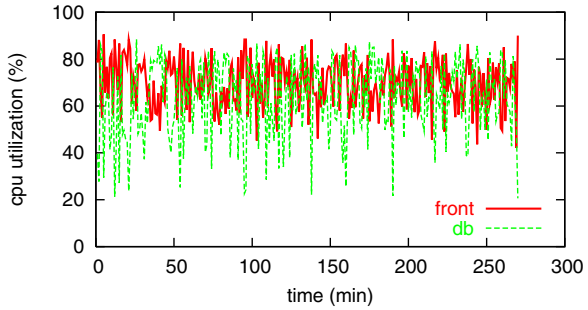


Fig. 3. Utilization of Front and DB Servers at 1 min granularity while processing browsing transaction mix under high load (400 EBs).

Figure 3 shows the CPU utilization of the front and database servers across time (at 1 min granularity) for the browsing mix with 400 EBs. The figure shows that there is a continuous bottleneck switch between the front and database servers over time. If switching of bottlenecks occurs across time, one can observe increased client response times and violations of service level agreements, while server utilizations on individual components remain modest [18]. This non stable behavior is difficult to model. Traditional analytic and simulation models assume that system is capable of higher throughput. We return to this phenomenon in the future sections with modeling results.

III. Transaction as a Unit of Client/Server Interaction

Since service providers are interested in dynamic resource provisioning methods for their production systems under live, real workloads, we must first understand which are the most important properties of these workloads to incorporate in analytic models. To this end, we first focus on what is required at the server side to generate a reply in response to a web page request issued by a client.

Typically, a client communicates with a web service (deployed as a multi-tier application) via a web interface, where the unit of activity at the client-side corresponds to a download of a web page. In general, a web page is composed of an HTML file and several embedded objects such as images. A browser retrieves a web page by issuing a series of HTTP requests for all objects: first it retrieves the main HTML file and after parsing it, the browser retrieves all embedded images. Thus, at the server side, a web page retrieval corresponds to processing multiple smaller objects that can be retrieved either in sequence or via multiple concurrent connections. It is common that a web server and application server reside on the same hardware, and shared resources are used by the application and web servers to generate main HTML files as well as to retrieve page embedded object [1]. Additionally, the main HTML file may be built via dynamic content generation where the page content is generated on-the-fly to incorporate customized data retrieved via multiple queries from the back-end database.

Since the HTTP protocol does not provide any means to delimit the beginning or the end of a web page it is very difficult to accurately measure the aggregate resources consumed due to web page processing at the server side. There is no practical way to effectively measure the service times for *all* page objects, although accurate CPU consumption estimates are required for effective model parameterization. To address this problem, we define a *transaction* as a combination of *all* the processing activities at the server side to deliver an entire web page requested by a client, i.e.,

[1]This is the case for TPC-W implementation that uses PHP web-scripting/application development language [10], and it is common for many production systems that are built in a similar way.

generate the main HTML file as well as retrieve embedded objects, and perform related database queries.

IV. Session-Based versus Transaction-Based Systems

While it is well-accepted [2], [6] that a workload of e-commerce and enterprise sites is more accurately described at the level of sessions, we focus on whether a simplified workload model that is only based on the probabilistic transaction mixes can be used for performance modeling of such sites.

A *session* is defined as a sequence of interdependent individual transactions. Therefore, effective system provisioning requires to evaluate the amount of system resources needed to support a targeted number of concurrent client sessions without violating a negotiated upper limit on the transaction response time. There are explicit transaction dependencies in session-based systems, e.g., "an order" cannot be submitted to an e-commerce system unless the previous transactions have resulted in "an item being ordered". Therefore, the *session-based system* is not stateless since the next client transaction explicitly depends on the previous ones. Such transaction dependency in the client behavior limits the opportunity for an efficient analytical model design. Because we aim at simple analytic models, we focus on simplifying the workload such that all transaction dependencies are ignored.

We refer to systems that do not have inter-request dependencies as *transaction-based systems*. The *question* we would like to answer is whether we can model well resource requirements of a session-based system by evaluating the resource requirements of its simplified transaction-based equivalent.

Assume that there is a total of N transaction types processed by the server. We use the following notation:

- let $p_{i,j}$ be the probability of the transaction type i following the transaction type j in the same client session, where $1 \leq i, j \leq N$;
- let $\mathbf{P}$ be the probability matrix of the transition probabilities of all the transaction types, i.e.,

$$\mathbf{P} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,N-1} & p_{1,N} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,N-1} & p_{2,N} \\ \cdots & & & & \\ p_{N-1,1} & p_{N-1,2} & \cdots & p_{N-1,N-1} & p_{N-1,N} \\ p_{N,1} & p_{N,2} & \cdots & p_{N,N-1} & p_{N,N} \end{bmatrix} \quad (1)$$

- let $\pi = [\pi_1, \pi_2, \cdots, \pi_N]$ be the vector of stationary probabilities of the transactions, i.e., $\pi P = \pi$ and $\pi \mathbf{e} = 1$, where $\mathbf{e}$ is a column vector of 1s with the appropriate dimension.

Vector π represents the steady-state probability all transactions, i.e., π_i gives the overall percentage of transactions of type i in the workload.

In order to compare performance of session-based versus transaction-based system, we designed and implemented a simulation model of session-based system and its transaction-based equivalent as follows:

- *session-based model*: we simulate the *real* session behavior of each client. The transaction type is determined when a client sends out the request to the system (according to the pre-defined transition probability matrix $\mathbf{P}$), and this transaction type generates the appropriate sequence of requests to the other tiers in the modeled multi-tier system. The next client transaction in the session is generated according to the transaction probability matrix $\mathbf{P}$.
- *transaction-based model*: each tier has the same transaction mix as the session-based system. However, the transaction type in each tier is selected according to the stationary probabilities π.

This simulation model is implemented using C++Sim [13].

For performance comparison we use the *browsing*, *shopping*, and *ordering* workloads in TPC-W as defined in Section II. Figures 4-5 present the simulation results for these workloads modeled as session-based versus transaction-based systems. Figure 4 and Figures 5 show system throughput and average transaction response time, respectively, for the three workload mixes. Simulation results confirm that performance and resource requirements of session-based systems in multi-tier environment can be efficiently modeled by their simplified transaction-based equivalent.

Under the transaction-based workload, each transaction arriving in the system is totally independent of other transactions while the overall transaction distribution is the same as in the system with session-based behavior. Such transaction distribution can be easily monitored for an existing production system. If we find a way to approximate the service time of each transaction type in the workload, then we can evaluate the average service time for the entire system under changing workload conditions (i.e., under varying transaction mix and load conditions over time) and design compact and efficient analytical models answering capacity planning and resource requirement questions.

V. CPU Cost of Transactions

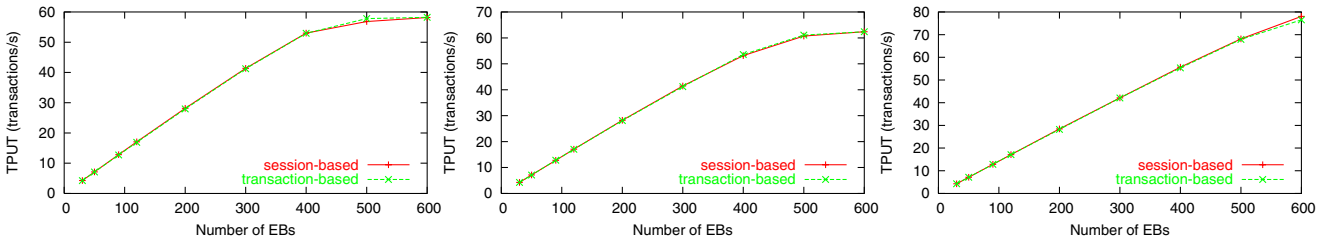
In this section, we propose a statistical regression-based approach for an efficient approximation of CPU demands of different transaction types. With the knowledge of CPU demands of transactions one can easily compose the resource requirement of scaled or modified transaction mixes. Thus, this methodology can be directly applied to production systems operating under live, real workloads, and can be used for explaining large-scale system behavior and predicting future system performance.

Prerequisite to applying regression is that a service provider collects the following:

- the application server access log that reflects all processed client transactions (i.e., client web page accesses);
- the CPU utilization if every tier of the evaluated system.

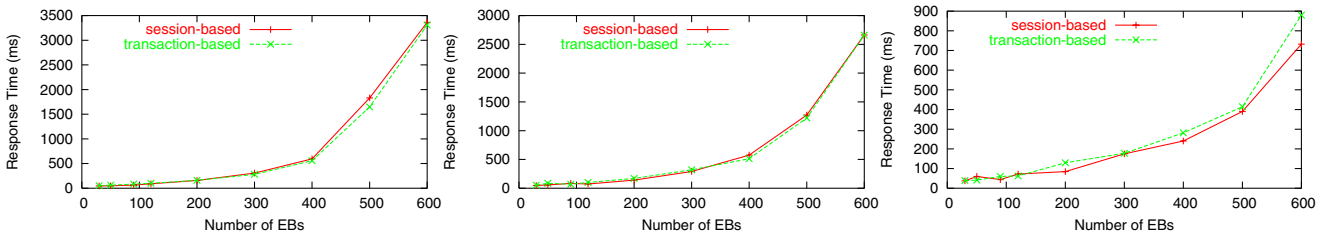
A. Regression Methodology

To capture the site behavior across time we observe a number of different transactions over *monitoring windows* of fixed length T. The transaction mix and system utilization are recorded at the end of each monitoring window.



(a) Browsing (b) Shopping (c) Ordering

Fig. 4. Session-Based versus Transaction-Based Model: Average Throughput under Three TPC-W Transaction Mixes.



(a) Browsing (b) Shopping (c) Ordering

Fig. 5. Session-Based versus Transaction-Based Model: Average Response Time under Three TPC-W Transaction Mixes.

Assuming that there are totally N transaction types processed by the server, we use the following notation:

- T is the length of the monitoring window;
- N_i is the number of transactions of the i-th type, where $1 \leq i \leq N$;
- $U_{CPU,n}$ is the average CPU utilization at n-tier during this monitoring window;
- $D_{i,n}$ is the average service time of transactions of the i-th type at the n-tier of the system [2], where $1 \leq i \leq N$.

From the utilization law, one can easily obtain Eq. (2) for each monitoring window.

$$\sum_i N_i \cdot D_{i,n} = U_{CPU,n} \cdot T. \tag{2}$$

Because it is practically infeasible to get accurate service times $D_{i,n}$, let $C_{i,n}$ denote the approximated CPU cost of $D_{i,n}$ for $0 \leq i \leq N$. Then, an approximated utilization $U'_{CPU,n}$ can be calculated as

$$U'_{CPU,n} = \frac{\sum_i N_i \cdot C_{i,n}}{T}. \tag{3}$$

To solve for $C_{i,n}$, one can choose a regression method from a variety of known methods in the literature. A typical objective for a regression method is to minimize either the absolute error:

$$\sum_j |U'_{CPU,n} - U_{CPU,n}|_j$$

or the squared error:

$$\sum_j (U'_{CPU,n} - U_{CPU,n})^2_j,$$

[2]This value is defined for all transactions and for all tiers. If there is no processing activity for transaction i at n-tier, then $D_{i,n} = 0$.

where j is the index of the monitoring window over time.

Finding the ideal regression method for the above problem is outside of the scope of this paper. In all experiments, we use the Non-negative Least Squares Regression (Non-negative LSQ) provided by MATLAB to obtain $C_{i,n}$. This non-negative LSQ regression minimizes the error

$$\epsilon = \sqrt{\sum_j (U'_{CPU,n} - U_{CPU,n})^2_j} \quad ,$$

such that $C_{i,n} \geq 0$.

This regression solver produces a solution for 200 equations with 14 variables only in 7 millisecond. In general, the common least squares algorithms have polynomial time complexity as $O(u^3 v)$ when solving v equations with u variables, and hence, can be efficiently used as a part of on-line resource evaluation method [1].

In the next two subsections, we explore the impact of monitoring window size and workload rates on the accuracy of the regression solution.

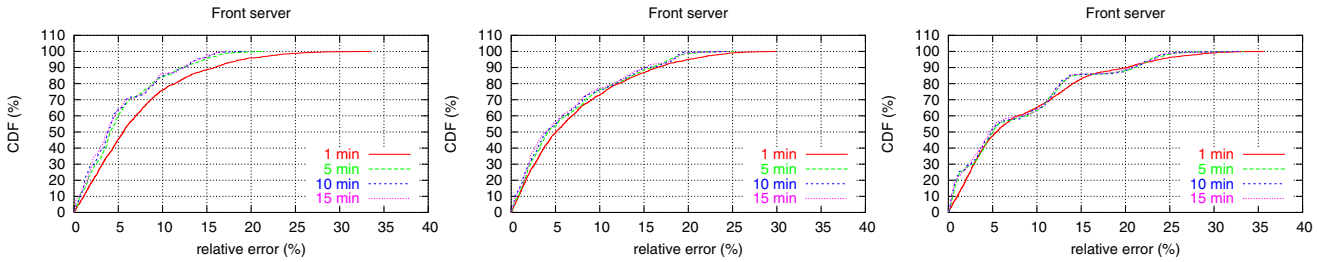
B. Sensitivity of Regression to Monitoring Window Size

We use the traces collected from the TPC-W experiments under the three workload mixes (i.e., browsing, shopping, and ordering mixes as described in Section II) to validate the accuracy of the proposed regression-based method.

Every minute, we monitor and record the following:

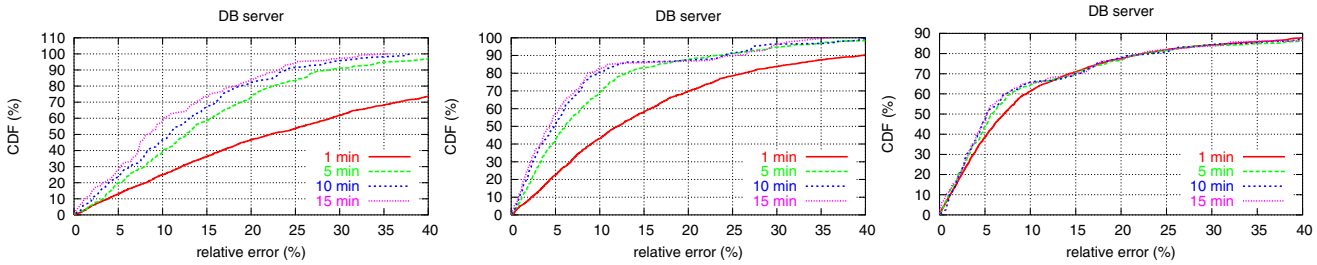
- the average CPU utilization $U_{CPU,n}$ at each n-tier in the system, and
- the number N_i of processed transactions of the i-th transaction type in the total 14 unique transaction types.

We then examine the sensitivity of the regression results to the length T of the monitoring window, i.e., T equal to 1 minute, 5 minutes, 10 minutes, and 15 minutes.



(a) Browsing (b) Shopping (c) Ordering

Fig. 6. Front Server: CDF of Relative Error of Regression Results under Different Monitoring Window Size.



(a) Browsing (b) Shopping (c) Ordering

Fig. 7. DB Server: CDF of Relative Error of Regression Results under Different Monitoring Window Size.

Using the aggregated values of N_1 to N_{14} and $U_{CPU,n}$ for each monitoring window T we obtain a set of equations in a form of Eq. (2) to approximate the CPU processing cost of transaction i at the n-tier, i.e., the $front$-tier and db-tier in our experiments. Then, using non-negative LSQ can solve this set of equations $C_{i,n}$ $(1 \leq i \leq 14)$ in order to estimate an approximation of the CPU processing cost of all transaction types across all tiers. After this step, the approximated $U'_{CPU,n}$ (we call it *fitted*) of every monitoring window is computed by using the original N_1 to N_{14} and the computed $C_{1,n}$ to $C_{14,n}$ values.

We use the relative error of the approximated CPU utilization with respect to the originally measured CPU utilization as a metric to validate the regression accuracy. For every monitoring window, the relative error of the approximated utilization is defined as

$$Error_R = \frac{|U'_{CPU,n} - U_{CPU,n}|}{U_{CPU,n}}. \quad (4)$$

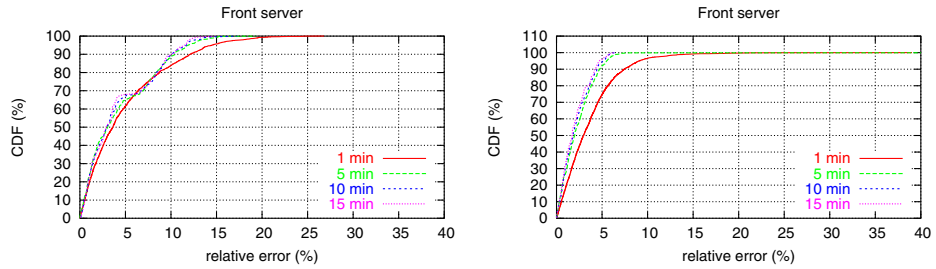
Figures 6 and 7 show the CDF of the relative errors for the front server and the database server under different lengths of monitoring window and the three TPC-W transaction mixes: browsing, shopping, and ordering. These performance results can be summarized as follows:

- *The approximation of CPU transaction cost at the front server is of higher accuracy than that at the database server.*
  For the three TPC-W transaction mixes, the relative errors of the CPU cost approximation at the database server is higher than that at the front server. Partially, this reflects a higher variance in the CPU service time at the database tier for the same transaction type. The relative errors of the CPU cost approximation at the database server is lower for shopping and ordering mixes as shown in Figures 7 (b), (c), while at the front server, the relative errors are lower for the browsing mix, see Figure 6 (a);
- *Larger T achieves higher accuracy.*
  The larger monitoring windows T work better, especially at the database server. For example, for browsing and shopping mixes, with T=1 min, the percentage of monitoring windows at the database server that show less than 20% of relative errors are 50% and 70%, respectively. With T=15 min, the percentage of monitoring windows at the database server with the same relative errors (less than 20%) increases to 83% and 89%, respectively. Larger T allows us to find a better "average" approximation for a variable CPU service time for the same transaction type.
  A larger monitoring window T has less impact at the front server. However, for the browsing mix, it still provides a reasonable improvement: with T=15 min 87% of monitoring windows show less than 10% of relative error compared to 77% of windows in the same error range when T=1 min.

By considering "worst" / "best" numbers across the three transaction mixes and using a larger monitoring window T=15 min, we can summarize the accuracy of regression results for approximating the CPU transaction cost as follows:

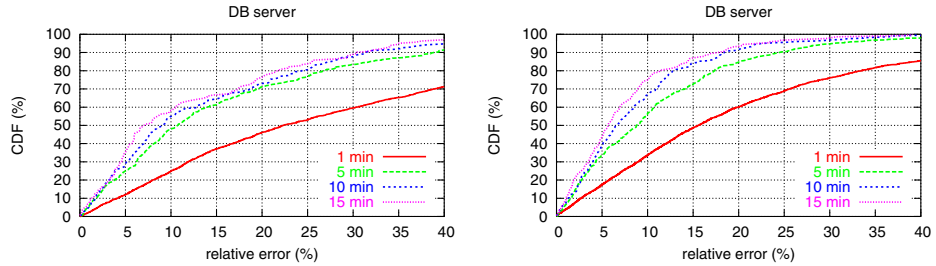
- at the front server: 87% - 98% of monitoring windows have relative errors less than 15%;
- at the database server: 79% - 89% of monitoring windows show relative errors less than 20%.

Now, we turn our attention to the impact of workload type on the accuracy of regression.



(a) Light Load (b) Steady Load

Fig. 8. Front Server: CDF of relative error of regression results under light and steady loads.



(a) Light Load (b) Steady Load

Fig. 9. DB Server: CDF of relative error of regression results under light and steady loads.

C. Sensitivity of Regression Results to Workload Rate

As our more detailed analysis indicates that the CPU service time might be load dependent, we evaluate this conjecture by splitting regression equations into two sets according to their corresponding loads. Measurements from experiments with less than or equal to 200 EBs are used to get CPU costs under light load, and data from experiments with larger than 200 EBs are used to get the costs under steady load. Here, we do not partition equations and results according to different workload mixes, but rather present the overall (combined) impact of all mixes on the accuracy of CPU transaction cost approximation.

Figures 8–9 present the combined CDF of relative errors across the three TPC-W mixes: browsing, shopping, and ordering, under light load and steady load. Comparing these figures, we can summarize the observations as follows:

- The approximation of CPU transaction cost is much more accurate when the regression is done separately for different workload rates. This observation holds for both the front and the database servers.
- The approximation of CPU transaction cost is less accurate under the "light" workload rates. Partially, it is due to a smaller number of transactions per monitoring window, and at the same time, higher variance of processing time in a lightly loaded system.

It means that in the modelling exercise one can use the transaction cost as a function of load, e.g., use two-values transaction cost for low and steady load areas.

Overall, we demonstrated that regression provides a simple and powerful solution to accurately approximate CPU transaction costs, especially with appropriately tuned monitoring window size and with the workload rates (or system load) taken into account.

VI. Analytic Model

Our next step is to use the results of the regression method to parameterize an analytic model of queues to enable dynamic evaluation of required system resources under changing workload conditions [3]. In this section, we explore this idea and perform a detailed performance study comparing the accuracy of our analytic model for resource usage evaluation with the real system results.

A. Analytic Model

Because of the upper limit on the number of simultaneous connections at a web server (which is reflected by the fixed number of EBs in the TPC-W benchmark), the system can be modeled as a closed system with a network of queues, see Figure 10.

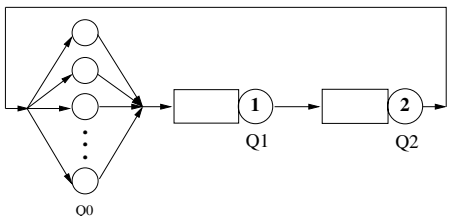


Fig. 10. The queuing model of the TPC-W environment.

The number of clients in the system is fixed and circulate in the network. When a client receives the response from the server, it issues another request after certain think time,

[3]For the TPC-W benchmark and most production multi-tier services CPU is a typical system bottleneck. However, in practice, when one needs to make a projection of the maximum achievable system throughput, additional "back of the envelope" computations for estimating memory and network requirements under the maximum number of concurrent clients are required to justify this maximum throughput projection.

i.e., after spending some time at Q_0. One could argue that since some of the requests are satisfied in at the front server tier, i.e., Q_1, therefore there must be direct flow from Q_1 back to Q_0. This is not needed here since we do not model each single visit at each tier, but an aggregated service time spent in each tier by a transaction.

This model can be efficiently solved using Mean-Value Analysis (MVA) [8], a classic algorithm for solving closed product-form networks. This model takes as inputs the think time in Q_0 and the service demands of Q_1 and Q_2, and provides average system throughput, average transaction response time, and average queue length in each queue. The think time in Q_0 is defined by the TPC-W benchmark as exponentially distributed with mean equal to 7 seconds, this is the value used in all experiments here. In production systems this value can be measured on-line or extracted by analyzing historic data. The average service demand at tier n is computed as follows. First, the CPU cost $C_{i,n}$ is obtained by regression for all i and all n. After calculating the transaction mix distribution vector π (see Section IV), the overall service demands at tier n is given by

$$S_n = \sum_{i=1}^{14} \pi_i \cdot C_{i,n} \tag{5}$$

The above value is used by the MVA model to evaluate the maximum achievable system throughput for the three TPC-W transaction mixes: browsing, shopping, and ordering.

B. Simulation Model

We also evaluate an accuracy and performance of our transaction-based simulation model introduced and used in Section IV. Here, we briefly describe its basic functionality. After a certain think time (exponential distributed), the client sends a transaction to the front server. The transaction type i is randomly selected according to the stationary probabilities π of the browsing, shopping, or ordering mixes. Then, the front server processes this transaction with an exponentially distributed service time with mean is equal to $C_{i,front}$ of the front server, i.e., the approximated CPU cost of transaction type i as given by regression. If this transaction type issues a query to the database server then the database server processes it and sends the reply back to the client. The service time at the database server is exponentially distributed with mean equal to $C_{i,db}$, this value is also provided by regression.

C. Modeling Results

Figure 11 compare the analytic results with the simulation of the detailed session-based model and experimental measurements of the real system. The results of the analytic model perfectly match the experimental results for the shopping and ordering mixes. The results also validate the simplified transaction-based model: its performance results are also in excellent agreement with experimental values.

For the browsing mix, both analytic and simulation models predict higher system throughput than the measured one. The reason that the two models do not do as well relates to the bottleneck switching behavior for browsing mix under higher loads: we discussed and demonstrated this phenomenon in Section II. However, even for this challenging case with a continuous bottleneck switch, the error remains contained within 15%, providing a close answer to the fundamental problem of how many simultaneous sessions can be concurrently supported by the system.

VII. Approach Limitations

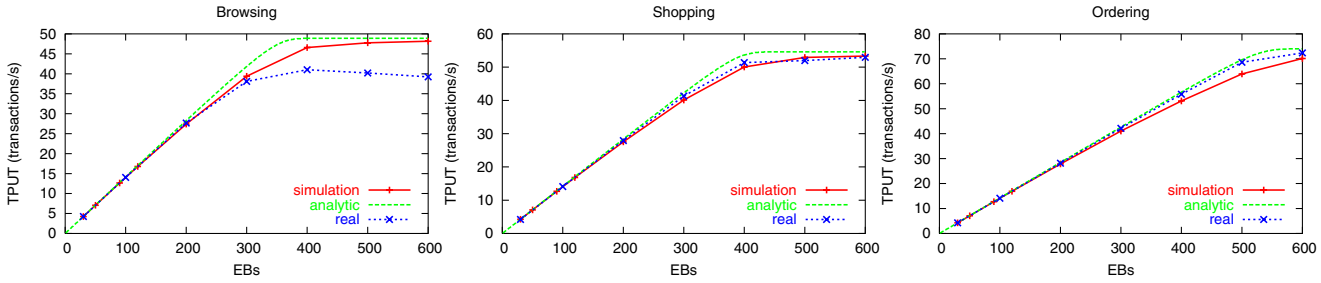
Once we approximated the CPU cost of different client transactions at different tiers, then we could use these cost functions for evaluating the resource requirement of scaled or modified transaction workload mix, in order to accurately size a future system. Ideally, one would like to use the CPU cost function obtained with the regression method under *WorkloadMix_1* to predict the system behavior under a different *WorkloadMix_2*. In this section, we try to assess the accuracy of performance predictions under drastic changes in the workload using the analytic model.

Figures 12 (a)-(c) present the system average throughput under different workload mixes. The lines on the graphs have the following meaning:

- the line labeled "*browsing*" ("*shopping*" or "*ordering*") means that the model is parametrized with CPU transaction costs derived with regression from the system that is processing the browsing mix (shopping or ordering mix respectively);
- the line labeled "*all*" means that the model is parametrized with CPU transaction costs derived with regression from the aggregate profile with all three mixes. It mimics the situation when the workload mix is changing and varying over time, i.e., when the system is processing over different periods of time either browsing, or shopping, or ordering transaction mixes;
- the line labeled "*real*" reflects measured performance of the real system.

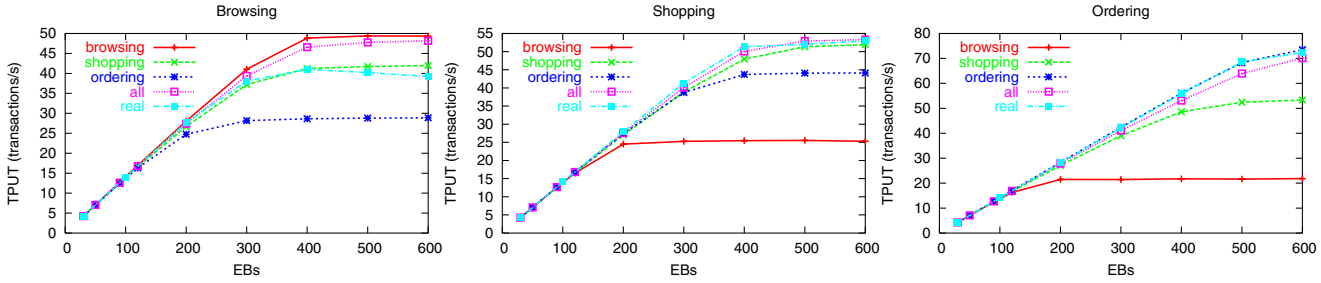
The observations from the modeling results can be summarized as the follows.

- The cost function "*all*" obtained from the aggregate profile of all the workload mixes gives excellent results for a diverse set of workloads. The maximum error with this cost function occurs when it is used to approximate system performance under the browsing mix. For the browsing mix, the model overestimates performance by 15%. The reason that the product form model does not do we well here is the bottleneck switching behavior that we discussed in Section II and Section VI-C.
- The cost function obtained by the profile of a stable workload mix gives excellent accuracy for the same workload mix. The relative error is under 2% when using the cost function from the shopping (or ordering) profile into the shopping (or ordering) simulation.
- The transaction cost function should not be applied to a very different workload mix compared to the mix it was derived from. For example, the relative error of the average throughput reaches 80% when the cost function from the browsing mix profile is used to simulate the ordering mix. This observation deserves



(a) Browsing (b) Shopping (c) Ordering

Fig. 11. Comparing Analytic Model Results with Simulation Model and Real System: Average Throughput under Three TPC-W Transaction Mixes.



(a) Browsing (b) Shopping (c) Ordering

Fig. 12. Comparing Performance Results when Analytic Model is Parametrized with Different CPU Transaction Cost Models.

further examination. In general, modeling results related to the browsing mix appear less accurate and this is explained by the continuous bottleneck switching at higher loads. Instead, when we used the "shopping" cost function to approximate system throughput under the ordering mix and vice versa, the error is contained within 20% – in these cases the errors come from the (inevitably) inaccurate approximations of the cost. Note that shopping and ordering mixes have 80%-20% and 50%-50% of transactions of the browsing and ordering types respectively (see Section II), so transactions from both classes are represented well in the overall mix (compared to 95%-5% ratio in the browsing mix). To obtain the accurate approximation of CPU transaction cost, the regression method requires a representative number of these transactions in the workload.

As the system and its workload evolve over time, continuously aggregated measurements like the ones used in cost function "*all*" allow to "adjust" the cost function and significantly improve model prediction.

VIII. Related Work

Performance evaluation and capacity planning of software and hardware systems is a critical part of the system design process [8]. There is a number of capacity planning techniques proposed for different popular applications.

Among these techniques, queuing theory is a widely used methodology for modeling a system behavior and answering capacity questions [17], [15], [16]. Modeling of a single-tier system, such as a simple HTTP server, has been studied extensively. Even for a multi-tier structure which is employed ubiquitously for most servers, the system is usually abstracted as the most bottlenecked tier only: in [17], only the application tier for the e-commerce systems are modeled by a M/GI/1/PS queue; similarly in [11] the application tier with N node cluster is modeled by a G/G/N queue. Recently B. Urgaonkar et al. proposed analytic models for both open and closed multi-tier systems [15], [16]. These models are validated by synthetic workloads running in real systems. However the expense of accurately estimating model parameters, i.e., service times and visit ratios, from each server log makes this model difficult to apply in production environments. Direct measurements in [16] do not characterize transactions as we do in this paper. Moreover, existing capacity planning methods are based on evaluating the system capacity for a fixed set of typical user behaviors. Once the service time is estimated, it is consistent throughout the planning procedure. This approach does not consider the fact that a changing workload for the same system has different service times and may result in different system capacity. Our experiments show that such techniques as those in [16] may fail to model a real system because of its dynamic nature.

In this paper, we use a similar closed multi-tier model as in [16], but in contrast to [16] or other examples in the existing literature of capacity planning, we propose a methodology that does not need a controlled environment for analytic model parameterization. Instead of characterizing the overall service time of every server, we use a statistical regression method to approximate the service

cost of individual transactions. This CPU cost function together with the transaction mix help to approximate the system service time that varies with the changing transaction mix.

The use of statistical methods in capacity planning has been proposed in the early 80's [3], [8], but the focus was on a single machine/cluster that is much simpler than current large-scaled multi-tiered systems. Recently statistical methods are getting more attention in computer performance analysis and system performance prediction. In [12] the authors use multiple linear regression techniques for estimating the mean service times of applications in a single-threaded software server. These service times are correlated with the Application Response Measurement package (ARM) data to predict system future performance. In [4],[5] the authors focus on transaction mix performance models. Based on the assumption that transaction response times mostly consist of service times rather than queueing times they use the transaction response time to approximate the transaction service demand. The authors use linear regression to identify performance anomalies in past workloads and to scrutinize their causes. We do not use measured transaction response times to derive CPU transaction demands (this approach is not applicable to the transactions that themselves might represent a collection of smaller objects).

The contribution of our paper is that it illustrates how a multi-tier system with a complex session-based workload can be modeled with a transaction-based mix. This approximation reduces the number of requisite parameters in a workload and further allows for the use of regression to derive the model parameters from direct measurements that are available at any production system, making a step toward a practical way to effectively model complex, live system with few parameters only.

IX. Conclusion

Predicting and controlling the issues surrounding system performance is a difficult and overwhelming task for IT administrators. With complexity of enterprise systems increasing over time and customer requirements for QoS growing, effective models for quick and automatic evaluation of required system resources in production systems processing diverse real workloads become a priority item on the service provider's "wish list".

In this work, we develop a practical solution to above problem by providing a theoretical framework which enables the resource evaluation of complex session-based systems through the performance modeling of their transaction-based equivalent. Once dealing with "stateless" transaction-based workloads, we design an analytic model for evaluating multi-tier system performance that is based on a network of queues representing the different tiers. This model is capable of modeling diverse workloads with changing transaction mix over time. The effectiveness of the proposed framework is based on a regression-based methodology to approximate the CPU demands of customer's transactions on a given hardware along all the tiers in the system. The statistical regression works very well for estimating the CPU demands of transactions that themselves might represent a collection of smaller objects and where direct measurement of the cost of each object is not feasible.

We illustrate the effectiveness of this methodology via a detailed set of experiments under different settings in the controlled TPC-W e-commerce suite. Our experiments show that for the majority of cases, the analytic model provides an accurate performance prediction compared to experimental data. However, the regression results should be used with care. The CPU transaction demands that are derived from workload mix which is very different from the one that is used in prediction might lead to inaccurate performance projections. The accuracy of regression significantly improves when the CPU transaction demands are derived from the extensive, aggregate workload profile that incorporates these possible different behaviors.

While this paper concentrates on evaluating the CPU capacity required for support of a given workload, we believe that regression methods can be efficiently applied for evaluating other shared system resources. We plan to exploit this avenue in our future work.

References

[1] B. Ari and H. A. Giivenir. Clustered Linear Regression. Knowledge-Based Systems, v. 15, No. 3, 2002.

[2] L. Cherkasova, P. Phaal. Session Based Admission Control: a Mechanism for Peak Load Management of Commercial Web Sites. IEEE J. Transactions on Computers, v. 51, No. 6, 2002.

[3] T. M. Kachigan. A Multi-Dimensional Approach to Capacity Planning. In Proc. of CMG Conference 1980. Boston, MA, 1980.

[4] T. Kelly. Detecting Performance Anomalies in Global Applications. Second Workshop on Real, Large Distributed Systems (WORLDS'2005), 2005

[5] T. Kelly, A. Zhang. Predicting Performance in Distributed Enterprise Applications. HP Labs Tech Report, HPL-2006-76, May 2006.

[6] D. Krishnamurthy, J. Rolia, S. Majumdar. A Synthetic Workload Generation Technique for Stress Testing Session-Based Systems. IEEE Trans. Software Eng. 32(11), 2006.

[7] D. Lampman. Building the Next Generation of IT. URL `www.hpl/hp.com/news/2006/apr-jun/technology.html`

[8] D. Menasce, V. Almeida, L. Dowdy. Capacity Planning and Performance Modeling: from mainframes to client-server systems. Prentice Hall, 1994.

[9] D. Menasce and V. Almeida. Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning. Prentice Hall, 2000.

[10] PHP HyperText preprocessor. `www.php.net`.

[11] S. Ranjan, J. Rolia, H. Fu, E. Knightly. QoS-Driven Server Migration for Internet Data Centers. Proc. of IWQoS'2002, Miami, 2002.

[12] J. Rolia, V. Vetland. Correlating Resource Demand Information with ARM Data for Application Services. In Proc. of the ACM Workshop on Software and Performance, 1998.

[13] H. Schwetman. Object-oriented simulation modeling with C++/CSIM. In Proc. of 1995 Winter Simulation Conference, Washington, D.C., 1995.

[14] TPC-W Benchmark. URL `http://www.tpc.org`

[15] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal. Dynamic Provisioning of Multi-tier Internet Applications. In Proc. of the 2nd IEEE International Conference on Autonomic Computing (ICAC-05), Seattle, June 2005.

[16] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. An Analytical Model for Multi-tier Internet Services and its Applications. In Proc. of the ACM SIGMETRICS'2005, Banff, Canada, June 2005.

[17] D. Villela, P. Pradhan, D. Rubenstein. Provisioning Servers in the Application Tier for E-Commerce Systems. In Proc. of IWQoS'04, Montreal, Canada, 2004.

[18] Qi Zhang. The Effect of Workload Dependence in Systems: Experimental Evaluation, Analytic Models, and Policy Development. PhD Thesis, College of William and Mary, December 2006.