Cluster Comput (2008) 11: 197–211
DOI 10.1007/s10586-008-0052-0

A regression-based analytic model for capacity planning of multi-tier applications

Qi Zhang · Ludmila Cherkasova · Ningfang Mi · Evgenia Smirni

Received: 8 February 2008 / Accepted: 6 March 2008 / Published online: 25 March 2008

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 two case studies, 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.

Keywords Regression-based model · Capacity planning · Multi-tier systems · Changing workload

Q. Zhang (✉)
Microsoft, One Microsoft Way, Redmond, WA 98052, USA
e-mail: qizha@microsoft.com

L. Cherkasova
Hewlett-Packard Labs, Palo Alto, CA 94304, USA

N. Mi · E. Smirni
College of William and Mary, Williamsburg, VA 23187, USA

1 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) [12], 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 queuing networks [21–23]. 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 queuing 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 queuing 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 [22]. 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 provides an effectively "compacted" information on workload demands within a few model parameters only. Simplicity is an additional benefit of this solution as this entire 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 two case studies:

- a detailed set of experimentation using the TPC-W e-commerce suite [20], and
- an HP Open View Service Desk application (OVSD).

We present sensitivity analysis of the 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 in TPC-W, 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 TPC-W testbed is presented in Sect. 2. Section 4 shows how one can compress a session-based workload with a transaction-based one. The statistical regression-based approach is presented in Sect. 5. Section 6 gives the simple queuing network model that is used to model TPC-W under changing workloads. Approach limitations are discussed in Sect. 7. Section 8 presents the second case study with OVSD. Related work is given in Sect. 9. Conclusions and future work are outlined in Sect. 10.

2 Experimental environment

We built a test-bed of a multi-tier application using the *three-tier architecture* paradigm that has now 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 [20]. A high-level overview of the experimental set-up is illustrated in Fig. 1, and specifics of the software/hardware used are given in Table 1.

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, 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 concurrent client sessions such that the multi-tier system can support without violating pre-defined limits in average transaction response times.

According to the TPC-W specification, the number of concurrent sessions (i.e., customers) or emulated browsers

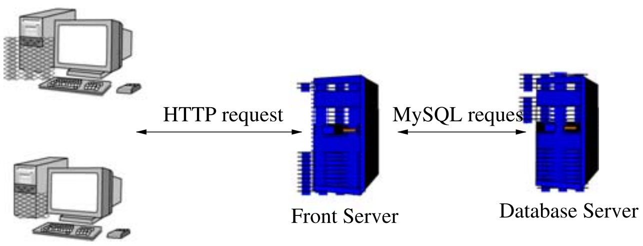
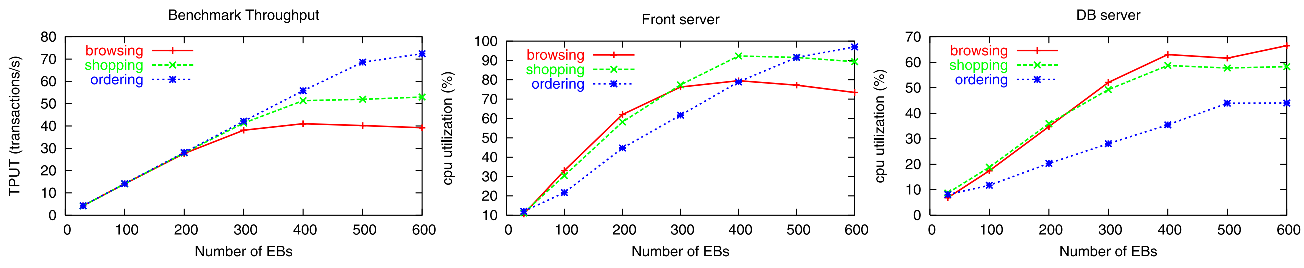


Fig. 1 E-commerce experimental environment

Table 1 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



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

Fig. 2 Three TPC-W Transaction Mixes: (**a**) Throughput; (**b**) Average CPU utilization of the front server; and (**c**) Average CPU utilization of the DB server

Table 2 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

Table 3 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%

(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 2.

One way to capture the navigation pattern within a session is through the *Customer Behavior Model Graph (CMBG)* [14], 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 3 gives the 5 most popular transaction types of each workload mix.

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. Each experiment ran for 5 hours. The first 20 minutes and the last 20 minutes are considered as 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 (i.e., 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/resource is the bottleneck: the average CPU utilization of both front and database tiers reaches 65–70%. It is not uncommon especially under bursty workload conditions [11, 24] for the system to become overloaded although average system utilization remains moderate. Here, average utilizations of both front and database servers are within the 65–80% range and ad-

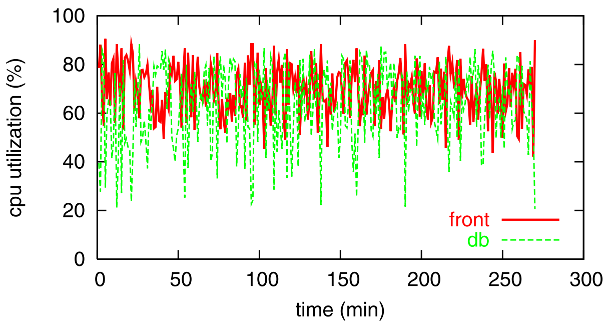


Fig. 3 Browsing Mix: Utilization of front and DB servers at 1 min granularity under 400 EBs

ditional performance measures show that I/O (either at the disk or network) is not the system bottleneck either.

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 [11, 24]. 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.

3 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, despite the fact that accurate CPU consumption estimates are required for effective model parameterization. To address this problem, we define a *transaction* as a combination of *all* processing activities at the server side to deliver an entire web page requested by a client, i.e., generate the main HTML file as well as retrieve embedded objects, and perform related database queries.

4 Session-based versus transaction-based systems

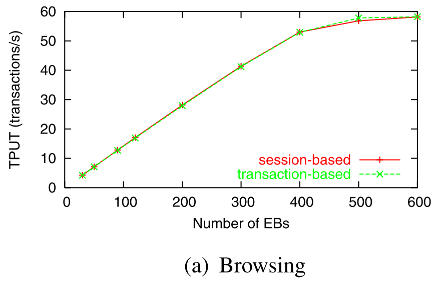
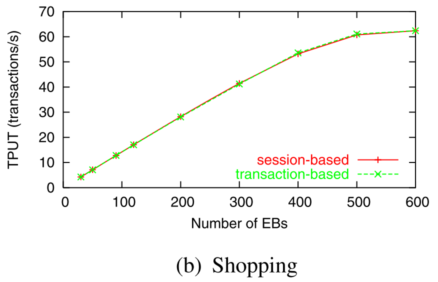
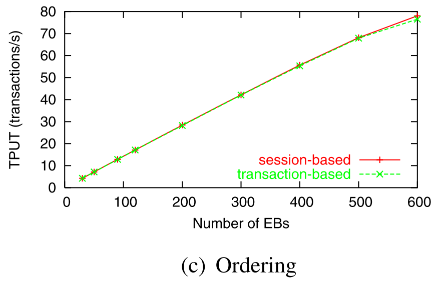
While it is well-accepted that a workload of e-commerce and enterprise sites is more accurately described at the level of sessions [6, 10], we focus on whether a simplified workload model that is only based on the probabilistic transaction mixes can be used for performance modeling of such sites.

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.

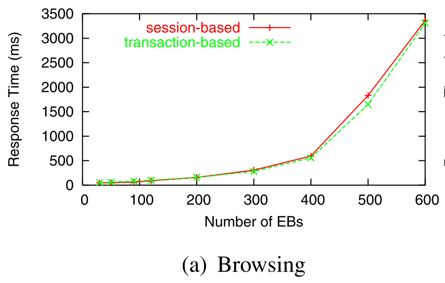
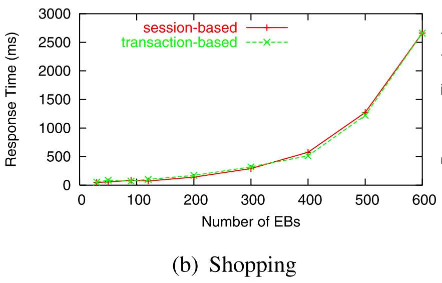
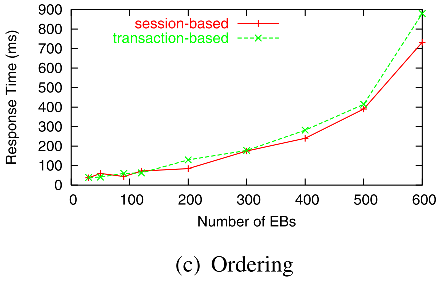
Assume that there is a total of N transaction types processed by the server. We use the following notation:

[1]This is the case for TPC-W implementation that uses PHP web-scripting/application development language [15], and it is common for many production systems that are built in a similar way.

(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

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

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

- let $\pi = [\pi_1, \pi_2, \ldots, \pi_N]$ 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 **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 **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 [18].

For performance comparison we use the *browsing*, *shopping*, and *ordering* workloads in TPC-W as defined in Sect. 2. Figures 4 and 5 present the simulation results for these workloads modeled as session-based versus transaction-based systems. Figures 4 and 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 that answer capacity planning and resource requirement questions.

5 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 of every tier of the evaluated system.

5.1 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.

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

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

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

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 $0 \le i \le N$. Then, an approximated utilization $U'_{\text{CPU},n}$ can be calculated as

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

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:

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

or the squared error:

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

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

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

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

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 of $O(u^3 v)$ 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 intensities on the accuracy of the regression solution.

5.2 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 Sect. 2) to validate the accuracy of the proposed regression-based method.

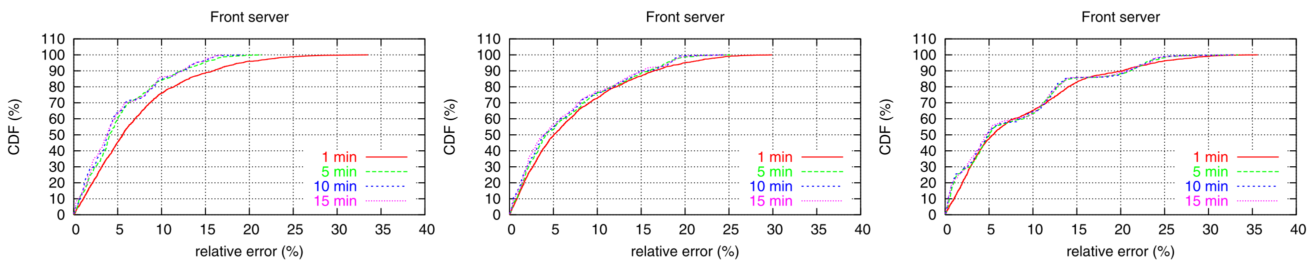
Every minute, we monitor and record the following:

- the average CPU utilization $U_{\text{CPU},n}$ at each n-th 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.

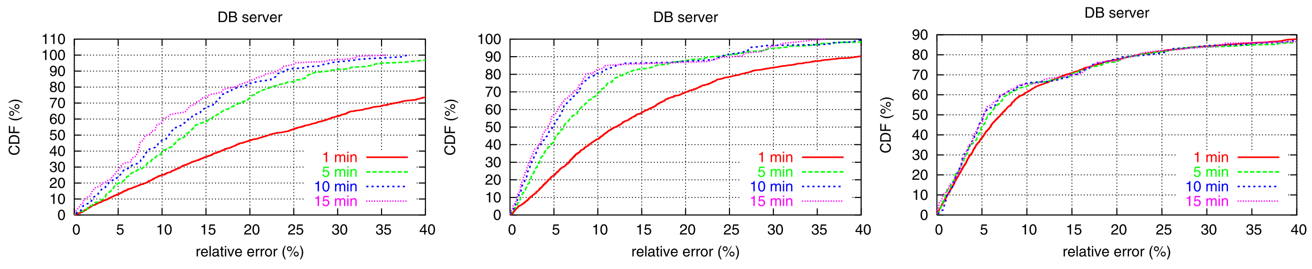
Using the aggregated values of N_1 to N_{14} and $U_{\text{CPU},n}$ for each monitoring window T we obtain a set of equations in a form of (2) to approximate the CPU processing cost of transaction i at the n-th tier, i.e., the *front*-tier and *db*-tier in our experiments. Then, using the non-negative LSQ, one can solve this set of equations for $C_{i,n}$ ($1 \le i \le 14$) in order to estimate an approximation of the CPU processing cost

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



(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

of all transaction types across all tiers. After this step, the approximated $U'_{\mathrm{CPU},n}$ (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

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

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 the shopping and ordering mixes as shown in Figs. 7 (b), (c), while at the front server, the relative errors are lower for the browsing mix, see Fig. 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 CPU service times of high variability 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%;

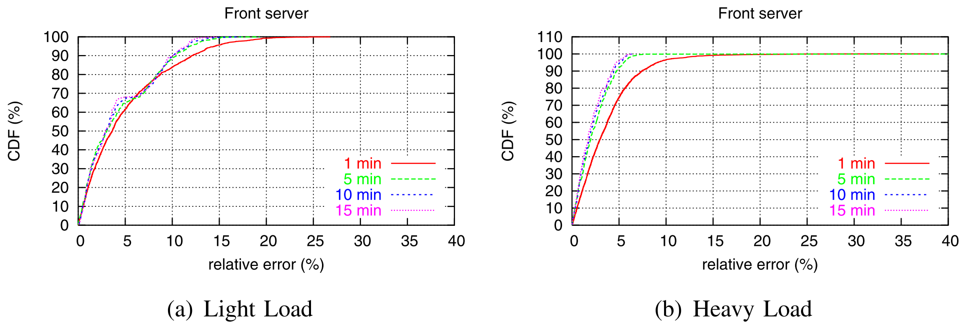


Fig. 8 Front Server: CDF of relative error of regression results under light and heavy system loads

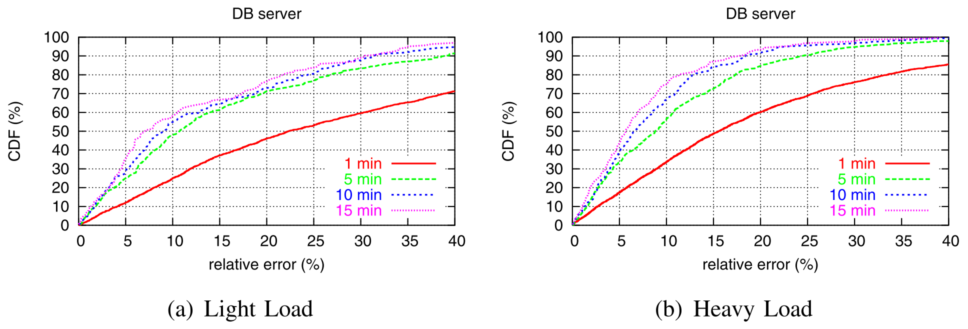


Fig. 9 DB Server: CDF of relative error of regression results under light and heavy system loads

- 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.

5.3 Sensitivity of regression results to workload intensity

Might be load dependent, we evaluate this conjecture by splitting the 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 heavy 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 and 9 present the combined CDF of relative errors across the three TPC-W mixes: browsing, shopping, and ordering, under light load and heavy 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 intensities. This observation holds for both front and database servers.
- The approximation of CPU transaction cost is less accurate under the "light" workload intensities. 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.

The above observations imply that in this modeling exercise one can use the transaction cost as a function of load, e.g., use two-values transaction cost under conditions of light and/or heavy load. 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 intensity (or system load) taken into account.

6 Analytic model and modeling results

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.

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 Fig. 10.

[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.

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, i.e., after spending some time at Q_0. One could argue that since some of the requests are satisfied in at the front 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 the aggregated service time spent in each tier by a transaction.

This model can be efficiently solved using Mean-Value Analysis (MVA) [13], a classic algorithm for solving closed product-form queuing 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 Sect. 4), the overall service demands at tier n is given by

$$S_n = \sum_{i=1}^{14} \pi_i \cdot C_{i,n}. \qquad (5)$$

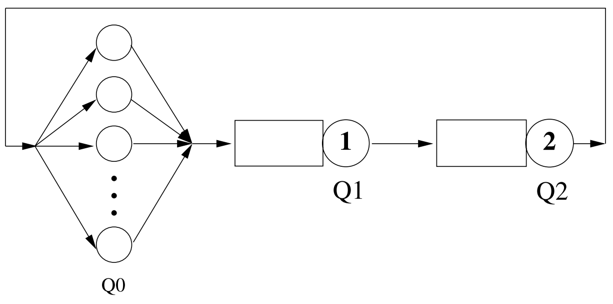


Fig. 10 The queuing model of the TPC-W environment

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.

We also evaluate an accuracy and performance of our transaction-based *simulation model* introduced and used in Sect. 4. 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.

Figure 11 compares 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 Sect. 2. 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.

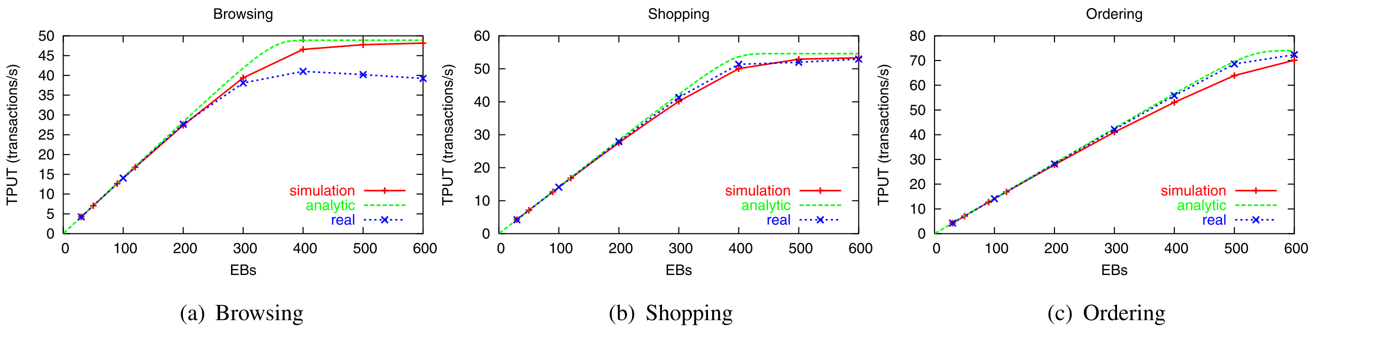


Fig. 11 Comparing analytic model results with simulation model and real system: average throughput under the three TPC-W transaction mixes

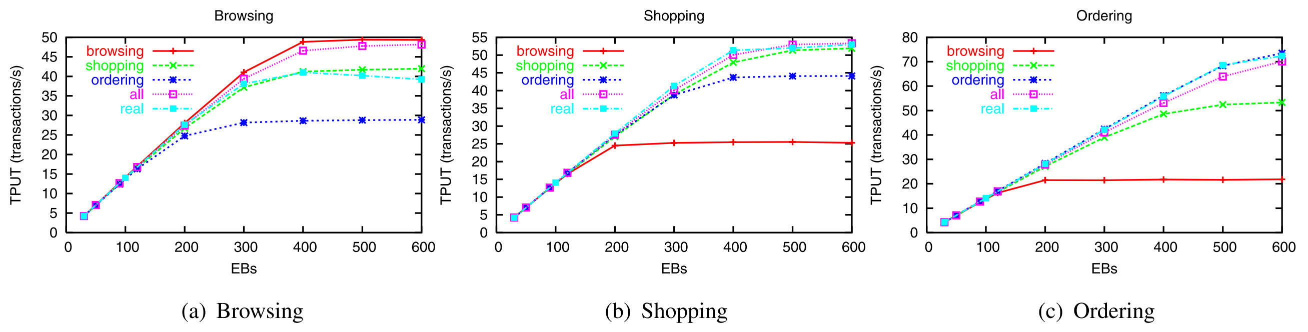


Fig. 12 Comparing performance results when the analytic model is parameterized with different CPU transaction cost models

7 Approach limitations

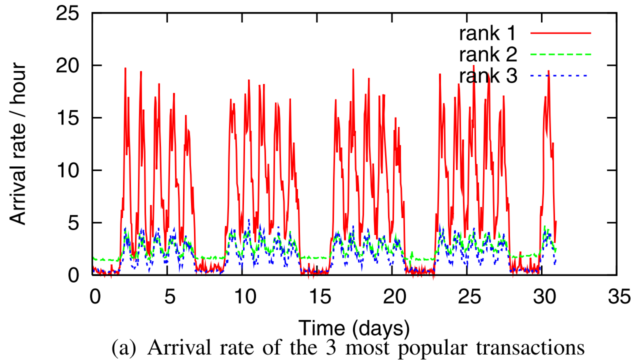
Once we approximated the CPU cost of different client transactions at different tiers, then we could use these cost functions for evaluating the resource requirement of scaled or modified transaction workload mix, in order to accurately size a future system. Ideally, one would like to use the CPU cost 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:

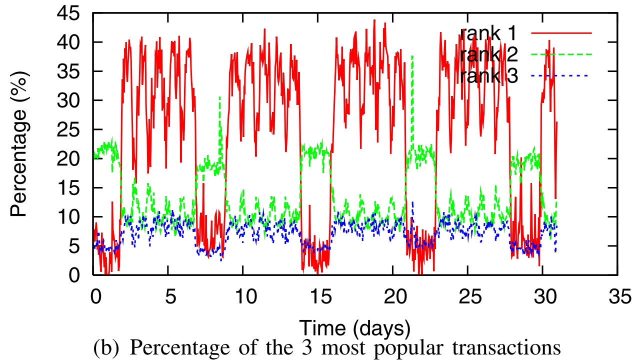
- the line labeled "*browsing*" ("*shopping*" or "*ordering*") means that the model is parameterized with CPU transaction costs derived with regression from the system that is processing the browsing mix (shopping or ordering mix respectively);
- the line labeled "*all*" means that the model is parametrized 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 obtained by the profile of a stationary 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, especially when this cost function is derived from a stationary mix. The complexity of deriving transaction cost function from the stationary mix is related to a high probability of correlation between some of the transaction types as well as chances that some groups of transactions are being not well represented. In both situations, the derived transaction cost can be "0". A transaction type is significant if its value is non-zero. 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 further examination. Note that shopping and ordering mixes have 80–20% and 50–50% of transactions of the browsing and ordering types respectively (see Sect. 2), so transactions from both classes are represented well in the overall mix (compared to 95–5% ratio in the browsing mix). When we derive CPU transaction costs from stationary browsing mix, most of the transactions of the ordering type result in "0" value in the solution, and hence they represent "non-significant" transactions in the browsing mix. The portion of the ordering transactions is greatly increased in shopping and ordering mixes. If we evaluate performance of these mixes using "0" cost function for the ordering transactions, it may result in a high prediction error. To avoid inaccurate predictions, one should apply the transaction cost estimates only to the transaction mixes with a similar set of significant transactions to the original mix used in the regression.
- The cost function "*all*" obtained from the aggregate profile of all the workload mixes gives excellent results for a diverse set of workloads. Under this approach, we solve the set of equations that represent all three workload mixes: browsing, shopping, and ordering. In such a way, we imitate the system with non-stationary workload: there are time periods when this system operates under varying browsing/shopping/ordering mixes. 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 (see also Sect. 2 and Sect. 6).



(a) Arrival rate of the 3 most popular transactions



(b) Percentage of the 3 most popular transactions

Fig. 13 Arrival rate and percentage of the first 3 most popular transactions in the overall workload across time

As the system and its workload evolve over time, continuously aggregated measurements like the ones used in cost function "*all*" allow to accurately approximate the cost function with respect to a broader set of transactions and significantly improve model prediction.

8 Applying regression to a production system with live workload

In this section, we apply the regression-based methodology that we developed earlier to evaluate the performance of a production system. We use a 1-month trace collected from the heterogeneous application servers at the HP Open View Service Desk (OVSD) business portal during July 2006. This trace has a detailed information about each processed request, including its arrival and departure time, request URL, and client session ID.

In our analysis, we consider a reduced trace that contains only transaction URLs as discussed in Sect. 3, and omit all embedded images, style sheets, and other format-related primitives. Moreover, we further distinguish a set of unique *transaction types* and a set of client accesses to them. For static web pages, the URL uniquely defines a file accessed by clients. For dynamic pages the requests from different users to the same web page URL may appear as requests to different URLs due to the client-specific extension or a corresponding parameter list. We carefully filter out these client-specific extensions in the reduced trace.

Figure 13(a) shows the arrival rates of the transactions for the 3 most popular transactions over time, and Fig. 13(b) shows the percentages of these transaction types in the workload mix over time. Each point in these figures corresponds to the statistics of one hour. It reflects a typical enterprise diurnal access pattern, i.e., high workload intensities during work hours and low intensities during nights and weekends. The figure shows that the transaction mix is not stationary over time. For example, the most popular transaction can constitute 15% to 40% of the workload depending on the hour of the day. Similar observations apply to transactions of lower popularity.

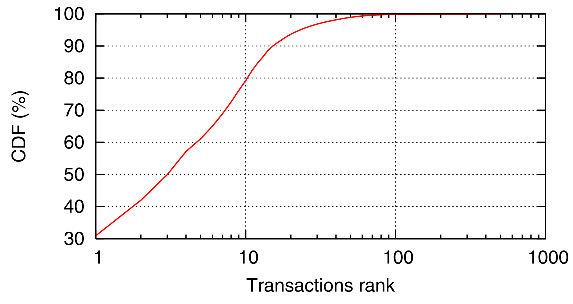


Fig. 14 CDF of the transaction popularity ranks

Traditional capacity planning methodologies usually examine peak loads and system utilization to conclude on the number of clients that can be handled by the system. These methods aim to accommodate variations in load (i.e., workload intensity) while assuming that the set of workload transactions is stationary, i.e., that the distribution of different transaction types is fixed. Many industry benchmarks are built using this principle [19, 20] but real workloads rarely exhibit this feature as shown by the analysis above. Therefore, instead of focusing only on loads, a robust capacity planning methodology must also consider the changing workload mix since the system capacity directly depends on the types of user activities.

Overall, in the reduced trace, there are 984,505 client requests that correspond to accesses of 756 different unique transactions (or transaction types). Figure 14 shows the cumulative distribution function (CDF) of client accesses to different transaction types ranked by their popularity. The transaction of *rank 1* corresponds to the most popular transaction type. Figure 14 shows that the studied workload exhibits a very high degree of reference locality: i.e., a small subset of site transactions is responsible for a very high percentage of client accesses, e.g.,

- the top 10 transaction types accumulate 79.1% of all the client accesses;

Table 4 An example of transaction profile in server 1

Time (hour)	N_1	N_2	N_3	N_4	$\cdots$	N_{756}	$U_{\text{CPU}}(\%)$
1	21	15	21	16	$\cdots$	0	13.3201
2	24	6	8	5	$\cdots$	0	8.4306
3	18	2	5	4	$\cdots$	0	7.4107
4	22	2	4	7	$\cdots$	0	6.4274
5	38	5	6	7	$\cdots$	0	7.5458
...							

- the top 20 transaction types are responsible for 93.6% of the site accesses;
- the top 100 transaction types account for 99.8% of all site accesses.

We use this one-month trace from the production system to evaluate the accuracy of the regression-based method described in Sect. 5. We limit our validation to the application server tier because we could not get relevant CPU utilization measurements at the database tier. For each 1-hour time window, we collected the average CPU utilization as well as the number of transactions N_i for the i-th transaction type, where $1 \le i \le M$. The OVSD trace profile has the format shown in Table 4.

In Sect. 5, we introduced and applied the regression-based technique for evaluating the transaction cost in TPC-W. There were only 14 different transaction types in TPC-W. The analysis of OVSD workload revealed that the real workloads often have a much higher number of transaction types, e.g., the OVSD workload operates with over 756 different transaction types. In order to apply the regression technique to OVSD workload we would need to collect more than 756 samples of 1-hour measurements. Such a collection would require to observe this workload for more than 1-month before we would collect enough "equations" for evaluating the OVSD transaction cost.

Our workload analysis above shows that the studied workload exhibits a very high degree of reference locality, i.e., a small subset of transactions is responsible for a very high percentage of client accesses, e.g., the 100 most popular transactions already cover 99.8% of all client accesses. This characterization is consistent with earlier works [2–4] that have demonstrated that web server and e-commerce workloads exhibit a high degree of reference locality. In addition, there is a high percentage of transactions that are rarely accessed, i.e., so called "one-timers". After dividing the 1-month trace in two halves and after examining each half individually, we found that there are 203 transactions that are accessed only once in the first half of the trace, and which are not accessed in the second half of the trace. Similarly, there are 189 transactions that are accessed only once in the second half of the trace, and which are not accessed in the first half of the trace. The non-negative LSQ regression used in this paper returns "0" as a typical value for "rare" variables, since there is not enough information in the original set of equations to produce an accurate solution.

So, the question is whether accurate performance results can be obtained by approximating the CPU cost of a much smaller set of popular (*core*) transactions. In other words, if we use regression to find the CPU cost of a small number of *core* transactions, can this small set be useful for an accurate evaluation of the future CPU demands in the system?

Following this idea, we only use the columns N_1 to N_K and U_{CPU} in Table 4 to approximate C_i for $1 \le i \le K$. The approximated U'_{CPU} of every hour is then computed by these N_1 to N_K and C_1 to C_K values.

We also consider the results produced by the non-negative LSQ regression method when K is equal to 10, 20, 60 and 100 transactions respectively. We divide the OVSD trace into two parts. The first half is used as a training set to solve for the CPU cost C_i using the non-negative LSQ regression. The second half is treated as a validation set. Because the administration jobs during weekends might introduce a significant noise to the CPU utilization, the training set for the regression consists of data from workdays only.

Regression produces similar results for the two heterogeneous application servers (referred as server 1 and server 2 in the text) in the system. Figures 15 and 16 show the CDF of the relative errors for the training and validating sets for servers 1 and 2, respectively. The results can be summarized as follows:

- Overall, the non-negative LSQ regression achieves good results for all examined values of K, i.e., when the regression method is applied to approximate the CPU cost of the top 10, 20, 60, or 100 most popular transactions. For the training set, at least 60% of the points have relative errors less than 10%, and at least 90% of the points have relative errors less than 20% (see Figs. 15(a) and 16(a)). The method's accuracy for the validating set is only slightly worse (see Figs. 15(b), 16(b)).
- Larger K achieves a higher accuracy for the training set. However, this improvement is not significant: for $K = 100$ there is only a 4% improvement compared to the results with the top 10 transactions.
- The larger values of K, e.g., $K = 100$, show worse prediction accuracy for the validating set compared to K equal to 10 or 20 core transactions as shown in Figs. 15 and 16. These results again can be explained by the workload properties. While we consider 100 most popular transactions, the last 80 of them only responsible for 6% of the client requests. These transactions have an irregular access pattern. Some appear only in the first or second half of the trace (while not being a "one-timer"). As a result, computing the individual cost of these transactions does

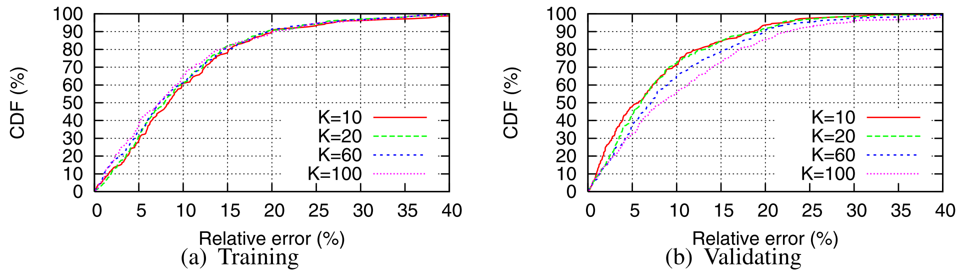


Fig. 15 CDF of relative errors for Server 1 under a different number of core transactions

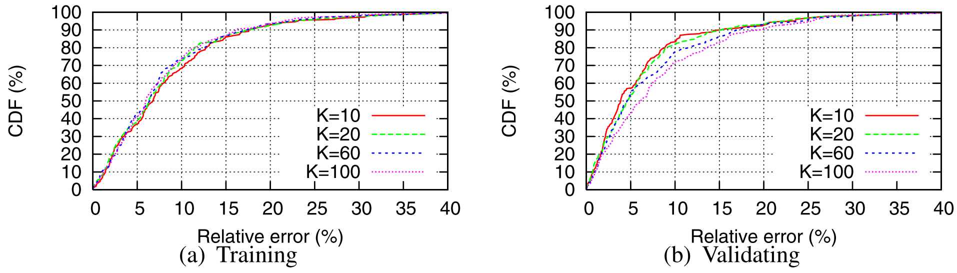


Fig. 16 CDF of relative errors for Server 2 under a different number of core transactions

not help to evaluate the future CPU demands, and introduces a higher error compared to the regression based on a smaller transaction set.

Regression produces the best results when a representative set of core transactions is used, and rarely accessed transactions are omitted. Since some of the rarely accessed transactions might only appear in the first half of the trace, while some different rarely accessed transactions may only appear in the second half of the trace, it is beneficial to use only core transactions in linear regression as well as in the overall capacity planning exercise. The additional CPU overhead that is due to the rarely accessed transactions is "absorbed" by the CPU cost of the core transactions. Consequently, a small additional CPU usage by the distinct and rarely accessed transactions is accounted via the CPU cost of the most frequently and consistently accessed core transactions.

We conclude that considering the top 20 *core* transactions (i.e., $K = 20$) leads to the most accurate results. Note that the top 20 transactions are responsible for 93.6% of the total transactions in the analyzed trace. Therefore, selecting the top K transactions that account for 90–95% of all client accesses for the regression method results in a good representative subset of the entire workload. The regression solver produces a solution for 200 equations with 20 variables only in 8 millisecond. In general, the common least squares algorithms have polynomial time complexity of $O(u^3v)$ when solving v equations with u variables, and hence, can be efficiently used as a part of an on-line resource evaluation method [1]. Combining the knowledge of workload properties with statistical regression provides a powerful solution for performance evaluation of complex production systems with real workloads.

9 Related work

Performance evaluation and capacity planning of software and hardware systems is a critical part of the system design process [13]. There is a number of capacity planning techniques proposed for different popular applications [5, 14, 21].

Among these techniques, queuing theory is a widely used methodology for modeling a system behavior and answering capacity questions [21–23]. 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 bottle-necked tier only: in [23], only the application tier for the e-commerce systems are modeled by a M/GI/1/PS queue; similarly in [16] 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 [21, 22]. 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 use in production environments. Direct measurements in [22] 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 [22] 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 [22], but in contrast to [22] 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 [7, 13], 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 [17] 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 [8, 9] the authors focus on transaction mix performance models. Based on the assumption that transaction response times mostly consist of service times rather than queuing 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 because 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.

10 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 the 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. The second case study with data from HP OVSD service identifies a set of additional challenges due to more complex and diverse behavior of real systems than the synthetic TPC-W benchmark. To efficiently deal with a significantly increased set of transactions in real systems, we use the set of the most frequent transactions in the regression analysis. This approach provides a practical, flexible and accurate solution for answering capacity planning and performance questions for multi-tier production systems with real workloads.

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.

Acknowledgements This work was originated and mostly completed while Qi Zhang worked in HPLabs during the summer 2006 and Ningfang Mi worked in HPLabs during the summer 2007. E. Smirni has been partially supported in part by the National Science Foundation under grants ITR-0428330 and CNS-0720699, and by a gift from Hewlett-Packard.

References

1. Ari, B., Giivenir, H.A.: Clustered linear regression. Knowl.-Based Syst. **15**(3) (2002)
2. Arlitt, M., Williamson, C.: Web server workload characterization: the search for invariants. In: Proc. of the ACM SIGMETRICS ’96 Conference, Philadelphia, PA, May 1996
3. Almeida, V., Bestavros, A., Crovella, M., de Oliveira, A.: Characterizing reference locality in the WWW. Technical Report, Boston University, TR-96-11 (1996)
4. Arlitt, M., Krishnamurthy, D., Rolia, J.: Characterizing the scalability of a large web-based shopping system. J. ACM Trans. Internet Technol. **1**(1) (2001)
5. Capacity Planning for WebLogic Portal: URL http://edocs.bea.com/wlp/docs81/capacityplanning/capacityplanning.html
6. Cherkasova, L., Phaal, P.: Session based admission control: a Mechanism for Peak Load Management of Commercial Web Sites. IEEE J. Trans. Comput. **51**(6) (2002)
7. Kachigan, T.M.: A multi-dimensional approach to capacity planning. In: Proc. of CMG Conference, Boston, MA (1980)
8. Kelly, T.: Detecting Performance Anomalies in Global Applications. Second Workshop on Real, Large Distributed Systems (WORLDS’2005) (2005)
9. Kelly, T., Zhang, A.: Predicting performance in distributed enterprise applications. HPLabs Tech Report, HPL-2006-76, May 2006
10. Krishnamurthy, D., Rolia, J., Majumdar, S.: A synthetic workload generation technique for stress testing session-based systems. IEEE Trans. Softw. Eng. **32**(11) (2006)
11. Mi, N., Zhang, Q., Riska, A., Smirni, E., Riedel, E.: Performance impacts of autocorrelated flows in multi-tiered systems. Perform. Eval. **64**(9–12), 1082–1101 (2007)
12. Lampman, D.: Building the Next Generation of IT. URL www.hpl/hp.com/news/2006/apr-jun/technology.html
13. Menasce, D., Almeida, V., Dowdy, L.: Capacity Planning and Performance Modeling: from Mainframes to Client-Server Systems. Prentice Hall, New York (1994)
14. Menasce, D., Almeida, V.: Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning. Prentice Hall, New York (2000)
15. PHP HyperText preprocessor: www.php.net
16. Ranjan, S., Rolia, J., Fu, H., Knightly, E.: QoS-driven server migration for Internet data centers. In: Proc. of IWQoS’2002, Miami (2002)
17. Rolia, J., Vetland, V.: Correlating resource demand information with ARM data for application services. In: Proc. of the ACM Workshop on Software and Performance (1998)
18. Schwetman, H.: Object-oriented simulation modeling with C++/CSIM. In: Proc. of 1995 Winter Simulation Conference, Washington, DC (1995)
19. The Workload for the SPECweb96 Benchmark: URL http://www.specbench.org/osg/web96/workload.html
20. TPC-W Benchmark: URL http://www.tpc.org
21. Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P.: Dynamic provisioning of multi-tier Internet applications. In: Proc. of the 2nd IEEE International Conference on Autonomic Computing (ICAC-05), Seattle, June 2005
22. Urgaonkar, B., Pacifici, G., Shenoy, P., Spreitzer, M., Tantawi, A.: An analytical model for multi-tier Internet services and its applications. In: Proc. of the ACM SIGMETRICS’2005, Banff, Canada, June 2005
23. Villela, D., Pradhan, P., Rubenstein, D.: Provisioning servers in the application tier for E-commerce systems. In: Proc. of IWQoS’04, Montreal, Canada (2004)
24. Zhang, Q.: The effect of workload dependence in systems: experimental evaluation, analytic models, and policy development. PhD thesis, College of William and Mary, December 2006

Qi Zhang currently is a software engineer in Windows Server Performance team at Microsoft. She received her Ph.D. degree in Computer Science from College of William and Mary, Williamsburg, VA, USA, in December 2006. She got the BS degree in computer science from Huazhong University of Science and Technology, Hubei, China, in 1998, and the MS degree in computer science from University of Science and Technology of China, Anhui, China, in 2001, respectively. Her research interests include performance evaluation, scheduling and load balancing policies, workload characterization and queuing modeling of multi-tiered systems, and departure processes. Qi Zhang is a member of ACM and IEEE.

Ludmila Cherkasova is a senior scientist in the Enterprise Software and Systems Laboratory at HPLabs, Palo Alto. She joined Hewlett-Packard Laboratories in 1991. Before joining HPLabs, she was a senior researcher at Institute of Computing Systems, Russia, and adjunct associate professor at Novosibirsk State University. Her current research interests are in distributed systems, Internet technologies and networking, performance measurement and monitoring, characterization of next generation system workloads and emerging applications in the large-scale enterprise data centers.

Ningfang Mi received her BS degree in Computer Science from Nanjing University, China, in 2000, and her MS degree in Computer Science from University of Texas at Dallas, in 2004. She is currently a Ph.D. candidate in the Department of Computer Science, College of William and Mary, Williamsburg, Virginia 23187-8795 (ningfang@cs.wm.edu). Her research interests include resource allocation policies, performance analysis of multi-tiered systems, workload characterization, and analytic modeling. She is a student member of the ACM and IEEE.

Evgenia Smirni Evgenia Smirni is the Wilson and Martha Claiborne Stephens Associate Professor at the College of William and Mary, Department of Computer Science, Williamsburg, VA (esmirni@cs.wm.edu). She received her Diploma in Computer Engineering and Informatics from the University of Patras, Greece, in 1987, and her M.S. and Ph.D. in Computer Science from Vanderbilt University in 1993 and 1995, respectively. From August 1995 to June 1997 she had a postdoctoral research associate position at the University of Illinois at Urbana-Champaign. Her research interests include analytic modeling, stochastic models, Markov chains, matrix analytic methods, resource allocation policies, Internet systems, workload characterization, and modeling of distributed systems and applications. She has served as program co-chair of QEST05 and of ACM SIGMETRICS/Performance06. She is a member of ACM, IEEE, and the Technical Chamber of Greece.