Resource Demand Modeling for Multi-Tier Services

Jerry Rolia
Hewlett Packard Labs
Bristol, UK
jerry.rolia@hp.com

Amir Kalbasi
University of Calgary
Calgary, Canada
akalbasi@ucalgary.ca

Diwakar Krishnamurthy
University of Calgary
Calgary, Canada
dkrishna@ucalgary.ca

Stephen Dawson
SAP Research
Belfast, UK
stephen.dawson@sap.com

ABSTRACT
We present a new technique for predicting the resource demand requirements of services implemented by multi-tier systems. Accurate demand estimates are essential to ensure the efficient provisioning of services in an increasingly service-oriented world. The demand estimation technique proposed in this paper has several advantages compared with regression-based demand estimation techniques, which many practitioners employ today. In contrast to regression, it does not suffer from the problem of multicollinearity, it provides more reliable aggregate resource demand and confidence interval predictions, and it offers a measurement-based validation test. The technique can be used to support system sizing and capacity planning exercises, costing and pricing exercises, and to predict the impact of changes to a service upon different service customers.

Categories and Subject Descriptors
I.6.5 [**Simulation and Modeling**]: Model Development-modeling methodologies

General Terms
Measurement, Performance, Experimentation

Keywords: Resource demand prediction, analytic performance models, benchmarking, statistical regression

1. INTRODUCTION
Software services can be very complex. A service may have many capabilities, each corresponding to a business level concept such as a business process. A customer that requires a service may actually exploit only a fraction of the service's capabilities. Each capability may use many different software functions that cause demands on a possibly distributed or multi-tier set of resources such as CPUs. The purpose of this work is to transform a specification for a customer's use of capabilities to an estimate for the service's corresponding resource demands. We assume that the relationship between capabilities and software functions is known in advance so that given a customer's specific throughputs for capabilities the corresponding desired *workload mix* of software functions is easily deduced.

Demand prediction can be difficult because computer measurement systems do not always offer resource usage measurements at the desired abstraction. For example, total server CPU utilization over some time interval and operating system process CPU utilization over some time interval may be available whereas the utilization of a CPU by a particular software function is not. Software level monitoring and logging facilities often provide counts for the number of times software functions are invoked and even measures of response times. However relating these to demands is difficult, particularly in distributed and multi-tier environments. In these environments, application servers are often multi-threaded, execute on hosts that often have multiple CPUs, may execute on virtualized hosts, and have many layers of caching that frequently delay input-output activity.

Regression techniques have been widely used [2][17][18][19][22][23][26] to support demand prediction by estimating per-function demands. Given throughputs for a set of capabilities, the corresponding expected workload mix of software function executions can be computed. For each resource, the resource demand is estimated as the sum of corresponding estimated per-function demands weighted by this mix. Many regression techniques suffer from the well-studied problem of multicollinearity [8] which can lead to unreliable predictions for demands and very wide confidence intervals for predicted demands. Furthermore, previous studies [18][19][23] have shown that the accuracy of regression techniques suffer if the per-function demands for a system are *not* deterministic, which is generally the case for computer systems. Finally, regression can provide incorrect confidence intervals for its demand predictions if assumptions exploited by the confidence interval calculations are not valid.

Our proposed solution is a Demand Estimation with Confidence (DEC). DEC requires the preparation and execution of a number of benchmarks under controlled conditions. A *benchmark* submits a semantically correct sequence of requests to a system under study. Resource demands are measured for each benchmark separately. A linear combination of the resource demands from the benchmark runs is then reused to offer service demand estimates for a desired workload mix. In contrast to regression, DEC does not rely on estimating per-function demands to estimate the overall demand for a customer. Consequently, it does not suffer from multicollinearity, is robust to non-determinism in per-function demands, and provides robust confidence interval predictions. Additionally, DEC provides for a measurement based test that, if desired, can be used to validate its demand predictions. Results from our TPC-W [24] case study show that DEC performs better overall than commonly used regression techniques such as Least Squares regression (LSQ) and Least Absolute Deviations regression (LAD). DEC does significantly better for cases with multicollinearity and in estimating confidence intervals for predicted demands.

WOSP/SIPEW'10, January 28–30, 2010, San Jose, California, USA.

Predictions for resource demands support sizing and capacity planning exercises. Such exercises typically rely on performance models, e.g., Queuing Network Models [8], which require as input parameters the demands for various resources in a system. Furthermore, demand predictions can also support costing and pricing exercises in pay as you go environments [1]. They can help software service providers estimate the quantity and hence cost of resources needed to provide a service based on a customer's specific expected use of a system. Accurate confidence intervals are essential for providers to properly assess risks related to poor performance. In addition, our solution provides a mechanism to predict the impact of changes to a system upon each of many customers that use it and upon an aggregate of customers that share a service platform.

Section 2 describes related work including LSQ and LAD. Section 3 introduces DEC and discusses the qualitative features of each approach. Section 4 provides a brief description of the DEC technique. Section 5 presents a case study that evaluates the effectiveness of DEC as compared to LSQ and LAD. Section 6 offers summary and concluding remarks.

2. RELATED WORK

Bard and Shatzoff studied the problem of characterizing the resource usage of operating system functions in the 1970's [2]. The system under study did not have the ability to measure the resource demands of such functions directly. The execution rates of the functions and their aggregate resource consumption were measurable and were recorded periodically. This data served as inputs to a regression problem. Bard used the least squares technique to successfully estimate per-function resource consumption for the study.

Equation 1 formalizes the LSQ regression problem for demand estimation. The problem has data from N measurement intervals. For each interval i there is a resource usage measurement Y_i and function execution count measurements for M functions, namely $F_{1i}...F_{Mi}$. Y_i is the *dependent variable*, $F_{1i}...F_{Mi}$ are the *independent variables*, and E_i is a random error associated with measurement of Y_i. $D_1...D_M$ are defined as *coefficients* of the independent variables. From a computer systems perspective, Y_i represents the aggregate demand on a resource due to executing the M functions as per $F_{1i}...F_{Mi}$. The coefficients represent estimates of per-function resource demands on some resource, e.g., a CPU. Accordingly, the contribution of a function k towards the aggregate demand Y_i is estimated as D_kF_{ki}. LSQ finds values for coefficients $D_1...D_M$ such that the objective function O is minimized. Furthermore, the coefficients $D_1...D_M$ are constrained to be positive since they represent demands. It should be noted that the regression model shown does not have a y intercept term. This is because aggregate demands should be zero for intervals where no functions are executed. For a desired customer mix of functions $F_1...F_M$, $\hat{Y}$ is an estimate of the resource demand for the mix.

$$
\begin{aligned}
&Y_i = D_1F_{1,i} + D_2F_{2,i} \cdots + D_MF_{M,i} + E_i, i = 1,2,\cdots N \\
&D_i \geq 0, i = 1..N \\
&O(D_1,\cdots D_M) = \sum_i \left(Y_i - D_1F_{1,i} - D_2F_{2,i} \cdots - D_MF_{M,i}\right)^2 \\
&\hat{Y} = D_1F_1 + D_2F_2 \cdots + D_MF_M
\end{aligned}
\qquad (1)
$$

LSQ is referred to as an l_2 method because it solves for coefficients such that the mean square of predictive errors with regard to the dependent variable is minimized. The regression model presented has several assumptions. First is the assumption of *linearity* between the dependent variable, i.e., resource usage, and the independent variables, i.e., function counts. Second, it is assumed that values for the independent variables are uncorrelated and *known* without error. Several assumptions relate to measurement errors. Measurement errors for the dependent variable should be *independent*, i.e., not be serially correlated. Furthermore, errors should have a constant variance, i.e., *homoscedasticity*, with respect to time, the dependent variable, and the independent variables. Finally, the measurement errors should follow the Normal distribution.

The effectiveness of least squares regression can be impacted if the assumptions are violated. If the linearity assumption is violated, then the regression model is likely to act as a poor predictor of the dependent variable. Failure of the linearity assumption can also cause the normality of errors to be violated. Non-normally distributed errors can arise even when the linear assumption is true when there are sources of variation in the system other than the random measurement error associated with the dependent variable. In particular, normality can be violated if the per-function demands are not deterministic. Various techniques can be used to assess whether the assumptions of regression hold [8]. Confidence interval calculations for predictions from LSQ rely on the assumption of normality of errors. Consequently, predicted confidence intervals can be unreliable if the normality assumption is violated.

LAD regression is less sensitive to outliers for the dependent variable than the LSQ technique. It has similar assumptions to LSQ but assumes measurement errors have a *Laplacian* distribution. The confidence interval calculations for predictions from LAD are different from those of LSQ due to the different distributional assumption made regarding the errors. LAD is referred to as an l_1 method because it solves for coefficients to minimize the sum of the absolute difference between predictions for the dependent variable and the measured values for the dependent variable. The problem statement for LAD is given in equation 2. As with LSQ, LAD can perform poorly if regression assumptions are violated.

$$
\begin{aligned}
&Y_i = D_1F_{1,i} + D_2F_{2,i} \cdots + D_MF_{M,i} + E_i, i = 1,2,\cdots N \\
&D_i \geq 0, i = 1..N \\
&O(D_1,\cdots D_M) = \sum_i \left|Y_i - D_1F_{1,i} - D_2F_{2,i} \cdots - D_MF_{M,i}\right| \\
&\hat{Y} = D_1F_1 + D_2F_2 \cdots + D_MF_M
\end{aligned}
\qquad (2)
$$

Many regression techniques suffer from the problem of multicollinearity. This arises when some tuples of "independent" variables are in fact correlated in the input data and these correlations are stronger than their correlations with the dependent variable. For example, if $F_{1,i}$ and $F_{2,i}$ have correlated values for i= $1..N$ then their impact on the Y_i will be confounded and indistinguishable to the solution for the coefficients D_1 and D_2. In this scenario, demand predictions for a different mix of F_1 and F_2 than is present in the input data may yield poor predictions for the dependent variable. Furthermore, the reported confidence

intervals for the predictions may be too wide to be of practical use.

Stewart *et al*. [22] suggest that multicollinearity is not as likely when observing the natural diversity of workload mixes in production systems as it is when observing synthetic benchmark workloads that typically produce a single "average" workload mix, e.g., [23][26]. Yet, Pacifici *et al.* show real evidence of significant multicollinearity in workload mixes of production systems [17].

Traditionally, multicollinearity is overcome by replacing groups of correlated "independent" variables with a single variable, for example by using a Principle Component Analysis (PCA) [8] or other reduction techniques [17]. However this reduces the expressive power of the prediction model because it reduces the number of variables that can describe a workload mix. This is problematic if a desired workload mix does not include such correlations or includes different correlations. Ridge regression techniques [5][12] can be applied to mitigate the impact of multicollinearity without decreasing expressive power. However, the effectiveness of ridge regression depends on the value selected for the so-called ridge parameter. In practice, an appropriate value for this parameter can only be determined by trial and error. In addition, the confidence interval calculations are only approximate and are more complex than those of LSQ. Furthermore such techniques still assume that predicted demands are deterministic. Section 5 shows that the DEC method does not suffer from the problem of multicollinearity and provides robust confidence interval calculations.

Sun [23] considered the sensitivity of predicted values for resource demands to violations in the deterministic demand assumption. He applied the LSQ and the Random Coefficients Method (RCM) for regression [23] to estimate per-function resource demands. Simulated values for per-function demands were drawn from distributions such as the Deterministic, Normal, and Exponential distributions. RCM aims to overcome issues of non-determinism in demands and did improve upon the results of LSQ. However both techniques failed when estimating the per-function demands for systems with more than 5 functions and as the resource usage distributions of the functions became less deterministic.

Zhang *et al.* apply a LSQ regression technique to estimate the per-URL demands and hence the resource utilizations of a TPC-W system [26]. The problem and approach was similar to that of Sun's work [23]. For the TPC-W system considered regression was found to achieve good accuracy overall in predicting resource utilizations. However, the study did not focus on multicollinearity since the dataset for which predictions were offered was the same as that used to obtain the regression coefficients. We consider the same TPC-W problem as Zhang *et al.* in the case study and apply LSQ, LAD, as well as our proposed DEC. We also consider in detail the impact of multicollinearity.

Recently several studies have investigated the use of queuing models to deduce workload parameters such as resource demands [13][14][25]. These techniques rely on measured response times from a system and a performance model for the system. Demand values are computed such that the model's mean response time prediction closely matches the mean of the measured response times. The problem posed in this paper differs in that we assume a customized workload mix is given as input but that both demands and response times for the customized mix are not known in advance. DEC is used to estimate the demands. Performance models that are not the focus of this paper can then use the demand estimates to predict response times.

Our work is related to Dujmovic's seminal work on benchmark design theory [6] and the work of Krishnaswamy and Scherson [11]. They model benchmarks as an algebraic space. However neither provides a method to express one benchmark as a linear combination, i.e., a ratio, of other benchmarks.

Krishnamurthy *et al.* introduce SWAT which views Web user sessions with computer systems as an algebraic space [9][10]. SWAT includes a method that automatically selects a subset of pre-existing user sessions from a session based system, each with a particular URL mix, and computes a ratio of sessions to achieve specific workload characteristics. For example, the technique can reuse the existing sessions to simultaneously match a new URL mix and a particular session length distribution and to prepare a corresponding synthetic workload to be submitted to the system. The work showed how such workload features impact the performance behaviour of session based systems. In the SWAT work, the concept of a session is identical to benchmarks as defined in this paper. DEC exploits this ratio computation technique to automatically compute a ratio of benchmarks that are then used to compute a resource demand estimate.

3. PROBLEM STATEMENT AND METHODS

Consider a service S with C capabilities that make use of M system functions that cause resource demands on R resources. Suppose S is a Customer Relationship Management (CRM) system. It has C business process variants for a customer to choose from. Each of the C variants c uses a subset of the M software functions. The functions cause resource demands on R resources such as application server CPUs and database server CPUs and disks. A customer's desired use of the system is specified using a throughput vector X of dimension C, such that c^{th} element of X is the required relative completions per unit time for a capability c, i.e., its required throughput X_c divided by the total required throughput T over all capabilities (equation (3)).

$$X = \begin{bmatrix} X_1/T \\ X_2/T \\ \vdots \\ X_C/T \end{bmatrix}, \quad where \quad T = \sum_{i=1}^{i=C} X_i \tag{3}$$

Given X we want to estimate the overall demands D^S of the service by the customer on the R resources as shown in equation (4).

$$D^S = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_R \end{bmatrix} \tag{4}$$

We now consider trivial, regression based, and DEC based solutions to this problem. A *trivial solution* to this problem is to create C benchmarks one for each capability c. Each benchmark

can be run in isolation to obtain per-capability resource measurements D^C on the R resources as shown in equation (5).

$$D^C = \begin{bmatrix} D_{11} & D_{12} & \cdots & D_{1C} \\ D_{21} & D_{22} & \cdots & D_{2C} \\ \vdots & \vdots & \cdots & \vdots \\ D_{R1} & D_{R2} & \cdots & D_{RC} \end{bmatrix} \quad (5)$$

$$D^S \approx D^C X \quad (6)$$

The product $D^C X$ shown in equation (6) estimates D^S the customer's overall demand on the R resources. However, this trivial approach is not feasible for systems with many capabilities. For example, the SAP ByDesign system [20] has thousands of business process variants. Developing, maintaining, and executing benchmarks for such a large number of process variant alternatives is cost prohibitive.

As described in the related work section, *statistical regression solutions* have also been used to assist with this kind of a problem. Suppose we have B benchmarks where $B << C$, each of the B benchmarks b uses a subset of the M functions. The B benchmarks are run to obtain function counts and resource demand measurements upon the R resources, as in equation (1). Regression, as described in Section 2, is used to *estimate* the per-function demands D^F on the R resources (equation (7)).

$$D^F = \begin{bmatrix} D_{11} & D_{21} & \cdots & D_{1M} \\ D_{21} & D_{22} & \cdots & D_{2M} \\ \vdots & \vdots & \cdots & \vdots \\ D_{R1} & D_{R2} & \cdots & D_{RM} \end{bmatrix} \quad (7)$$

As shown in equation (8), the C capabilities together have a known *feature visit matrix* F for the M functions such that F_{ij} represents the number of times function i is invoked in capability j.

$$F = \begin{bmatrix} F_{11} & F_{12} & \cdots & F_{1C} \\ F_{21} & F_{22} & \cdots & F_{2C} \\ \vdots & \vdots & \cdots & \vdots \\ F_{M1} & F_{M2} & \cdots & F_{MC} \end{bmatrix} \quad (8)$$

The product $F X$ gives a vector such that the i^{th} row of the vector contains the expected number of times function i is to be invoked due to the customer's throughput requirement X. As shown in equation (9), the product $D^F F X$ estimates the customer's overall demand D^S on the R resources.

$$D^S \approx D^F F X \quad (9)$$

We propose a *new Demand Estimation with Confidence (DEC) solution* to estimate resource demands. Suppose we have B benchmarks, $B << C$, such that each of the benchmarks b uses a subset of the M functions. As shown in equation (10), each benchmark b can be run in isolation to obtain resource measurements D^B upon the R resources for its aggregate use of functions.

$$D^B = \begin{bmatrix} D_{11} & D_{12} & \cdots & D_{1B} \\ D_{21} & D_{22} & \cdots & D_{2B} \\ \vdots & \vdots & \cdots & \vdots \\ D_{R1} & D_{R2} & \cdots & D_{RB} \end{bmatrix} \quad (10)$$

$$D^{B'} = \begin{bmatrix} D_{11} & D_{12} & \cdots & D_{1B'} \\ D_{21} & D_{22} & \cdots & D_{2B'} \\ \vdots & \vdots & \cdots & \vdots \\ D_{R1} & D_{R2} & \cdots & D_{RB'} \end{bmatrix} \quad (11)$$

where $B' \leq B$

Given a customer's requirement on system functions $F X$ we solve for a linear combination L of a subset of some size B' of the B benchmarks that yields the same use of functions $F X$ [9]. As shown in equation (11), $D^{B'}$ contains a subset of the columns of D^B.

$$L = \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_{B'} \end{bmatrix} \quad (12)$$

$$D^S \approx D^{B'} L \quad (13)$$

L, as depicted in equation (12), gives the weight of each benchmark in the subset needed to synthesize $F X$. A corresponding linear combination of the B' chosen benchmarks' demand measurements, denoted by the product $D^{B'} L$ in equation (13), estimates the customer's overall demand D^S on the R resources. This approach also permits a measurement based validation test where the linear combination L of the chosen B' benchmarks can be executed together to emulate the customer's use of system features $F X$. We note that it may not always be possible to obtain an exact match of $F X$ with the available B benchmarks. This is discussed in detail in Section 5. A description of the ratio finding technique is provided in Section 4.

DEC has several characteristics that differ from the regression based approach. First, DEC does not attempt to fit all measurement data with a single predictive regression model. Instead, for each customer desired function mix, an appropriate subset of benchmarks is found that mimics the desired function mix. Accuracy tolerances can be specified for each function. For example, one can specify that a resource intensive function be matched more accurately than other less resource intensive functions. Furthermore, DEC reports if there is insufficient information, i.e., benchmarks, to find an exact solution.

Second, the confidence interval calculation for a predicted demand is more robust for DEC than for regression based techniques. Recalling from Section 2, the confidence interval calculations for LSQ and LAD rely on distributional assumptions about the errors in measuring the dependent variable. Poor confidence interval estimates can result when these assumptions are violated, for example due to non-deterministic per-function demands. With DEC, confidence interval calculation is straightforward. Several independent experiment replications are conducted for each benchmark. Each benchmark replication yields a *benchmark replication demand*. The mean of the benchmark replication demands is computed as the overall *benchmark mean demand*. From the central limit theorem a

benchmark mean demand is normally distributed if the number of replications is large [8]. Furthermore, a linear combination of normally distributed independent random variables results in a random variable that is also normally distributed. Since DEC uses a linear combination of measured benchmark mean demands to predict the resource demand of a workload function mix, the predicted demand is also normally distributed. This allows confidence intervals to be computed with certainty for the predictions from DEC[1]. We provide experimental evidence in Section 5 that shows that DEC's confidence interval predictions are more robust than those of LSQ and LAD.

4. THE DEC TECHNIQUE

This section provides a brief summary of the method used to compute a linear combination vector L for a subset of benchmarks. As stated previously, we adapted for the demand prediction problem a ratio computation technique we devised previously [10] to support synthetic workload generation. Although the modifications required relative to the previous work were fairly straightforward, we describe the technique here for the sake of clarity and completeness.

Let the product FX, see equations (8) and (3), correspond to a customer's desired use of a system's functions. The problem of computing L is to determine a linear combination of a subset of the B benchmarks that results in $F\,X$. In other words, let A^* be a M x K matrix containing function execution counts vectors for K benchmarks, $K \leq B$, upon the M functions. The K rows of L represent benchmark execution counts for the benchmarks corresponding to the function counts vectors of A^*. If the chosen benchmarks were run with these benchmark execution counts the resulting function counts would be $F\,X$. The problem of computing the linear combination vector L is to determine an A^* and L such that the following conditions are satisfied.

$$A^*\,L = F\,X \tag{14}$$

$$L(l) \geq 0 \;\forall\, l \tag{15}$$

Equation (14) specifies that the function counts achieved by combining the benchmarks in A^* according to the benchmark counts in L should equal the desired function counts given by $F\,X$. Equation (15) restricts the computed benchmark counts to be non-negative values.

To solve the problem, we devised an algorithm which iteratively determines the A^* matrix and the L vector that satisfy the conditions given by equation (14) and equation (15). To begin, an initial A^* matrix is determined by identifying a small subset of the B benchmarks. This relies on the computation of an algebraic basis set for the B benchmarks. At each step of the iteration, linear programming is used to find a value of L for a given A^* such that the difference between the desired function counts $F\,X$ and the achieved function counts $A^*\,L$ is minimized. Equation (15) forms one of the constraints of the LP problem. The second constraint is obtained by relaxing the condition specified by equation (14) to that given by equation (16).

$$A^*\,L \leq F\,X \tag{16}$$

This change facilitates an iterative solution by which A^* is progressively modified by adding more benchmarks until an L that satisfies the stricter constraint given by (14), i.e., corresponding to a better match between $F\,X$ and $A^*\,L$ is found. Specifically, the difference between the desired function counts and the achieved function counts is calculated as a M x 1 *slack vector* $E = F\,X - A^*\,L$. The benchmark that offsets E the most is identified from the remaining benchmarks in B that are not part of A^*. The new benchmark to be selected is determined by computing the Euclidean distances between E and the function execution counts vectors of the remaining benchmarks**.** The vector that yields the minimum distance is selected and appended to A^* as an additional column. This is followed by another iteration of the algorithm.

The algorithm is guaranteed to terminate when the goal is to match a workload mix of functions that results from the B benchmarks. This is because in the worst case all B benchmarks will be included to achieve the match. The algorithm terminates if the mismatches in function counts, given by the elements of E, are less than or equal to user-specified tolerance thresholds for function counts or if a user-specified maximum number of iterations is reached. When an exact match of mix is not possible with a given set of benchmarks, the LP problem can be easily modified to match certain functions, e.g., resource intensive functions more closely than other functions. We exploit this capability in Section 5.

5. CASE STUDY

To verify the effectiveness of the approach we conducted a case study using the industry standard e-commerce benchmark system named TPC-W [24]. TPC-W has 14 system functions that correspond to various URL request types. Emulated customer browsers interact with the system to conduct ordering, shopping, and browsing sessions.

This section is organized as follows. Section 5.1 describes the experiment setup used for the study. Section 5.2 describes the experiment process. Section 5.3 presents results.

5.1. Experiment setup

Our testbed consists of a Web server node, a database server node, and a client node connected by a non-blocking Ethernet switch that provides a dedicated 1 Gbps connectivity between any two machines in the setup. The Web and database server nodes are used to execute the TPC-W bookstore application. We used the PHP-based TPC-W application developed at Rice University [2]. The client node is dedicated for running the *httperf* [15] Web request generator that was used to execute benchmarks on the TPC-W system. All nodes in the setup contain an Intel 2.66 GHz Core 2 CPU and 2 GB of RAM. We used the Windows *perfmon* utility to collect resource usage information from the Web and database server nodes using a sampling interval of 1 second. The CPU demands are much larger than disk and network demands for this system so we focus on demand estimation for these values. We note that the very low disk demands are likely due to caching mechanisms employed by the operating system and the database management system.

[1] The calculation of confidence intervals for linear combination of means is straightforward and can, for example, be found in [6].

5.2. Experiment process

For the case study, we created 100 benchmarks. These benchmarks were constructed from 40 random ordering sessions,

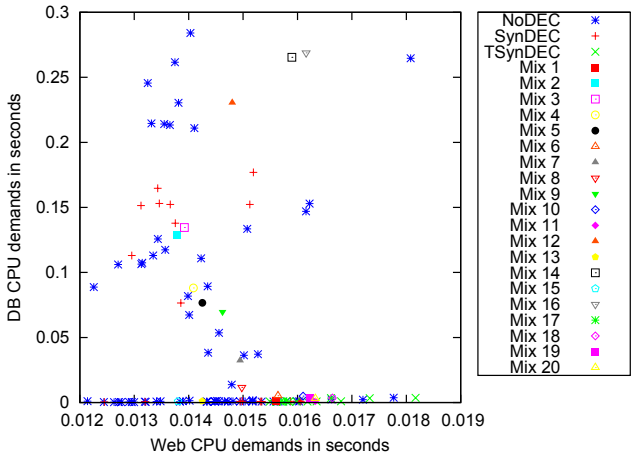

Figure 1. DB CPU per request demands versus Web CPU per request demands

40 random shopping sessions, and 20 random browsing sessions. These sessions were obtained by conducting random walks over the Markov chains specified by TPC-W. Each benchmark is based on exactly one of these 100 sessions. An execution of the benchmark involves sequentially submitting its corresponding session multiple times with stochastically generated arguments such as author names and book categories. For regression, the benchmarks were executed one after another sequentially using *httperf* and resource usage was tracked during this run. For DEC, each benchmark was submitted in isolation using *httperf* and resource usage was collected at the end of the run to characterize the benchmark replication demand. For each benchmark, 5 independent replications were performed to calculate the overall benchmark mean demand. Care was taken to ignore data from the initial part of each replication during which the system's caches are likely to be atypically "cold".

To validate DEC, our case study considers two scenarios. In the first scenario 20 workload mixes were considered with each representing a desired workload mix of a customer, *F X*. Care was taken to ensure that these mixes use functions in a different proportion than any of the 100 benchmarks alone. DEC was able to successfully synthesize workloads corresponding to each of these mixes as a linear combination of a subset of the 100 benchmarks. For each such synthesized workload 3 independent measurement runs were conducted to characterize that workload's overall mean demand so that it can be compared with the demand predicted by DEC and the other techniques. To provide more examples, in the second scenario, each of the benchmarks is in itself treated as a customer's desired workload mix with the remaining 99 serving as benchmarks from which this mix could be synthesized.

Benchmarking exercises were conducted to obtain reliable resource demand measurements for the 100 benchmarks and the 20 workloads. Confidence intervals for the mean measured demands were computed following the standard procedure that employs the Student's t distribution [8]. For DEC, confidence intervals were calculated by employing the confidence intervals for linear combinations of means approach [7]. Since the number of independent replications for each benchmark is less than 30, the Student's t distribution was used with this approach to calculate confidence intervals instead of the Normal distribution [7].

Figure 1 illustrates the Database server node (DB) CPU demand versus the Web server node (Web) CPU demand value for the 100 benchmarks and 20 workloads. The 20 workloads are denoted as *Mix 1* to *Mix 20* and were synthesized using DEC as described previously. *Mix 1* to *Mix 14* have mixes selected to give coverage of the Web CPU versus DB CPU space of the 100 benchmarks. *Mix 15* to *Mix 20* have mixes that were chosen to explore the impact of multicollinearity on demand prediction. The 100 benchmarks are presented in three groups. The 20 *SynDEC* benchmarks could be synthesized using a combination of two or more of the remaining benchmarks using DEC. The 13 *TSynDEC* benchmarks had the exact same use of system functions as some other of the 100 benchmarks, i.e., they could be synthesized in a trivial manner. The 67 *NoDEC* benchmarks represent cases where an exact match of mix could not be achieved.

From Figure 1, it can be observed that DEC is able to provide predictions throughout the Web and DB CPU space. The figure also shows that different kinds of sessions impose very different per request demands upon the system. The Web CPU demands differ by a factor of 1.5 over all cases. However, the DB CPU demands differ by a factor of 1000 over all cases, and by a factor of 265 if only cases where DB CPU demands greater than 1 ms are considered. For this system good demand estimates are needed for planning exercises, in particular for the DB CPU.

For LSQ and LAD, we considered measurement windows of 5, 10, 30, and 60 seconds for aggregating function counts and utilizations. The regression predictions were insensitive to the window size. As a result, we only present results from the 5-second case.

We used the following approach to measure the goodness of fit of a regression model to the input data. The error e_i for a measurement interval i is computed as the difference between the measured aggregate demand Y_i and regression's prediction of aggregate demand $\hat{Y}_i$. The mean absolute error is computed as follows and used as an indicator of model accuracy.

$$mean\ absolute\ error = \sum_i \frac{|e_i|}{Y_i} \qquad (17)$$

For both LSQ and LAD, the mean absolute error for the Web and DB CPU demands is approximately 10%. This indicates that both the LSQ and LAD regression models are able to overall accurately explain the behaviour over the 5-second measurement intervals considered. We note that the coefficient of multiple determination R^2 cannot be used as a goodness of fit metric for our models since they do not have a y intercept term [16].

5.3. Comparison of DEC, LSQ, and LAD

We now consider the ability of LSQ, LAD, and DEC to predict demands for the 100 benchmarks and 20 workloads. Figures 2 and 3 show the percentiles of relative error for the Web and DB CPU demand predictions, respectively. The figures have five sets of data. The DEC, LSQ, and LAD sets correspond to the 53 cases

where DEC was able to achieve an exact match of workload mix. The LSQ-NDEC and LAD-NDEC sets correspond to the 67 cases where DEC was not able to provide an exact solution.

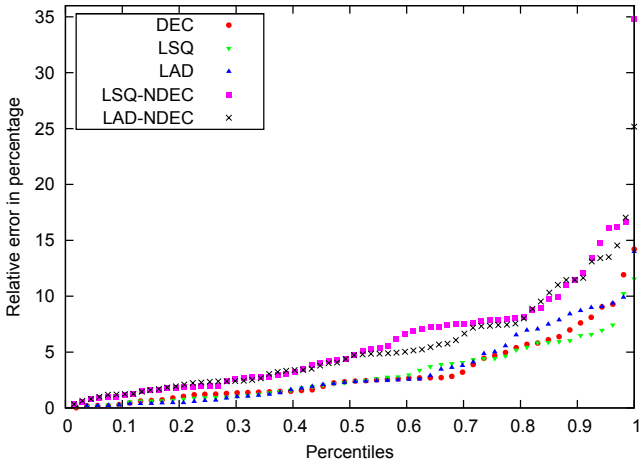

Figure 2. Percentiles of relative errors for Web CPU demand predictions

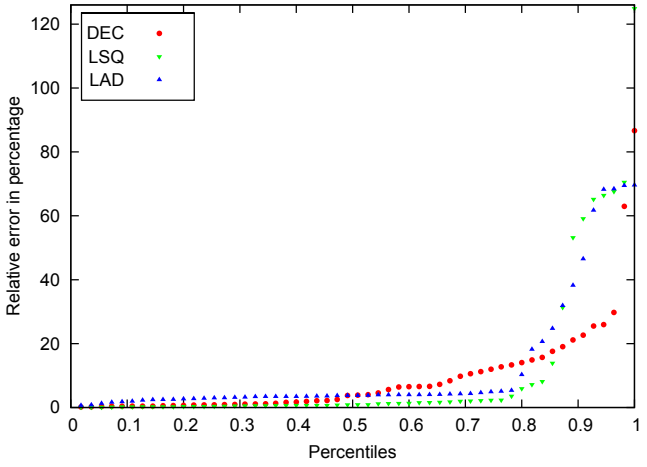

Figure 4. Percentiles of relative errors for DB CPU demands when cases with non-exact matches

For the cases where DEC provided an exact solution, Figure 2 shows that for the Web CPU demand estimates LSQ, DEC, and LAD perform similarly. In general the errors are low for all methods because the range of potential Web CPU demand values differed by a factor of only 1.5. Figure 3 shows DB CPU demand estimates for a subset of 30 of the 53 cases where CPU demands were greater than 1 ms and where DEC synthesized a non-trivial solution, i.e., using a mix of two or more benchmarks. The results show that DEC does much better than both LAD and LSQ for the DB CPU demand estimates. Recall that demand estimates differ by a factor of 265 for these 30 cases. We note that DEC had its largest errors when the demands were very small and when the DB CPU was not the bottleneck resource. LSQ and LAD had some of their greatest errors for the six cases that suffered due to the presence of multicollinearity.

Figures 2 and 3 also show the results for LSQ-NDEC and LAD-NDEC. These are regression results for the 67 cases where DEC did not have sufficient information to achieve an exact match of workload mix. Figure 2 shows that for these cases the regression based techniques have higher errors than for the cases where DEC was able to achieve an exact match. While the regression

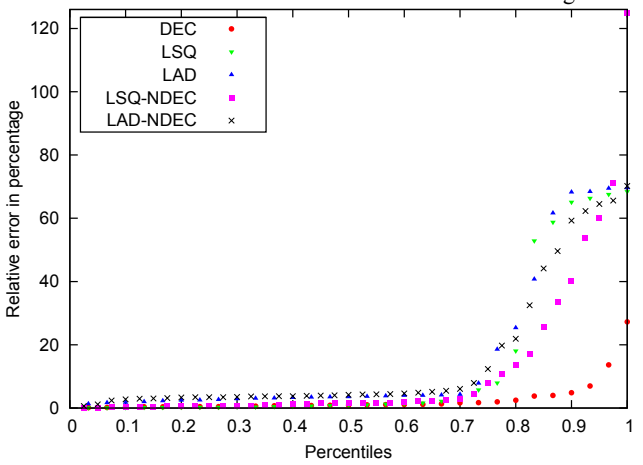

Figure 3. Percentiles of relative errors for DB CPU demand predictions

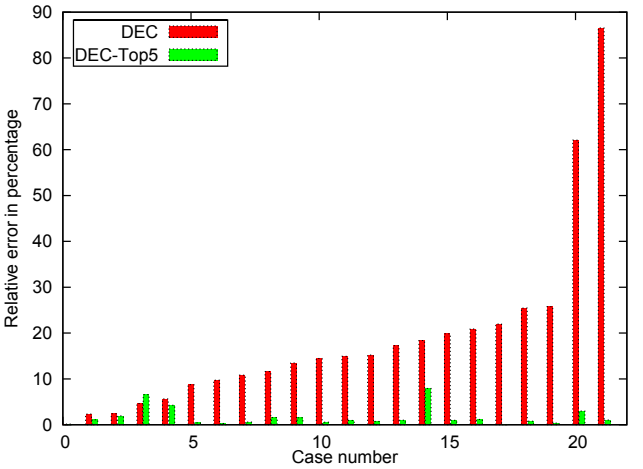

Figure 5. Comparison of DB CPU demand predictions by DEC and DEC-Top5 for cases with non-exact matches

techniques could be applied in these cases, they provide less accurate demand predictions. In contrast, if DEC cannot achieve an exact match it implies that more benchmarks are needed to provide appropriate measurement coverage of the workload mix space for functions.

Figure 4 compares the DB CPU demand predictions from DEC, LSQ, and LAD when the cases for which DEC did not have an exact match are included. As before, we only consider non-trivial cases whose demands were greater than 1 ms. The results show that although DEC's accuracies are comparable to those of LSQ and LAD, DEC predictions can become unreliable when mixes are not matched exactly.

We explore one possible solution for improving DEC for cases where an exact match is not possible. First we identified the top five resource intensive functions based on an analysis of response time data collected from executing the benchmarks. DEC's LP formulation was relaxed such that the technique only attempts to

Table 1: Predictions for per-function demands compared to no-load response time measurements

Request type	Mean R (ms)	CI_WIDTH (%)	Mean R LSQ (ms)	Mean R LAD (ms)	Error-LSQ (%)	Error-LAD (%)
Shopping Cart	55.73	206.25	22.34	20.19	59.91	63.78
Customer Registration	12.16	19.01	22.69	22.28	86.63	83.24
Buy Request	55.68	177.72	10.13	9.12	81.80	83.62
Buy Confirm	102.88	179.24	23.48	28.42	77.18	72.38

match the counts for these five functions exactly. A best effort solution is employed with respect to matching counts for the other functions. We denote this approach as DEC-Top5. Figure 5

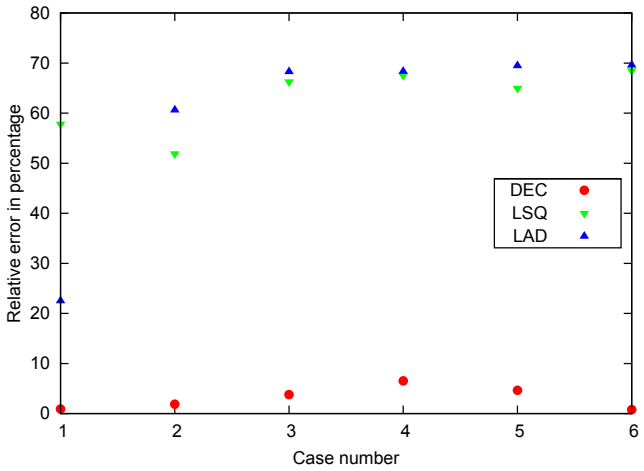

Figure 6. Relative errors for DB CPU demand predictions for cases with multicollinearity

considers the 22 cases from Figure 4 for which DEC did not have an exact match. The results show that DEC-Top5 is able to significantly reduce prediction errors for 21 of the 22 cases. For the one case where DEC-Top5 does worse than DEC (case 4 of Figure 5), the demand is very low and the difference in relative errors between the two techniques is only about 1.9%. The maximum error is about 8% for DEC-Top5 when compared to 86% for DEC and 20% for regression for this subset of 22 cases.

We conclude that DEC is flexible enough to permit systematic methods to improve predictions for cases where the available benchmarks do not permit an exact match of workload mix. Our future work will focus on automating for such cases the process of determining the LP formulation that achieves the best possible improvements in prediction accuracy. For the remainder of the paper we only consider cases for which DEC had an exact match.

Figure 6 shows the errors for DB CPU demand predictions for the three techniques for the six workloads chosen to study the impact of multicollinearity. A careful analysis of the function execution counts for the 100 benchmarks used for regression showed that the counts for the Buy Request and Buy Confirm functions were highly correlated. Specifically, the count for Buy Confirm function was equal to the count for the Buy Request function in 87% of the 100 benchmarks. The six workloads shown in Figure 6 were valid workloads for the TPC-W system and were constructed from a subset of the 100 benchmarks where the correlation for these functions is different. Only 40% of the 849 sessions making up these six workloads had equal counts for the Buy Request and Buy Confirm request types. The 100 benchmarks and these six workloads also exhibit a similar difference in the way the Shopping Cart and Customer Registration functions are used. As seen in Figure 6, due to the differences in correlation, the estimates of LSQ and LAD for the aggregate demand of the six workloads have large errors. The results show how sensitive regression can be to the multicollinearity phenomenon and that the accuracy of DEC is not sensitive to multicollinearity.

We now consider regression and its dependency on per-function demand estimates. Recall that regression estimates per-function demands and uses them to estimate a new service's aggregate demand value.

Table 1 shows the mean per-function response times *Mean R* measured for the Customer Registration, Shopping Cart, Buy Request, and Buy Confirm functions along with their 95% confidence intervals widths (*CI_WIDTH*). The *CI_WIDTH* values are expressed as percentage of their corresponding *Mean R* values. The per-function response time measurements were made such that there was only one active request at a time at the system. Consequently, the *Mean R* values computed from these response times can be compared with estimates for total resource demands over all resources as found using LSQ and LAD. We denote *Mean R LSQ* as the total resource demand over all resources computed as the summation of the Web CPU and DB CPU demands estimated by LSQ. Similarly, *Mean R LAD* is the total resource demand over all resources as estimated by LAD. *Error-LSQ* and *Error-LAD* represent the percentage absolute error between the total resource demands estimated by the regression techniques and the measured total demands represented by the *Mean R* values. The table shows that the per-function demand estimates of LSQ and LAD can have very large errors. As noted earlier, if the new service uses the functions in a different proportion to that given as input to regression, regression may yield a poor aggregate demand estimate. DEC does not rely on such per-function demand estimates when estimating the resource demands of services.

Table 1 also shows that the *CI_WIDTH* values for measured per-function mean response times can be quite large compared to the measured mean response time values. For the Shopping Cart function, the width of the two-sided confidence interval, based on measurements, is more than 200% of its measured mean response time. This suggests that the TPC-W system's per-function demands can be highly variable and are clearly not deterministic.

Figure 7 compares measured and predicted two-sided 95% confidence interval widths for the aggregate demand predictions for cases where DEC found an exact solution. From Figure 7 (a), for the Web server CPU the measured demands and predictions from DEC, LSQ, and LAD had confidence intervals that were within 6% of the predicted demand over 90% of the time. This is not surprising since the measured Web Server CPU demand only varied by a factor of 1.5 over all 120 cases. For the DB server CPU, we consider a subset of 30 cases where the mean DB CPU demand was greater than 1 ms and DEC reported a non-trivial solution. Recall that for these cases the demands varied by a factor of 265. Figure 7 (b) shows that the measured demands and DEC predicted demands for these 30 cases had confidence

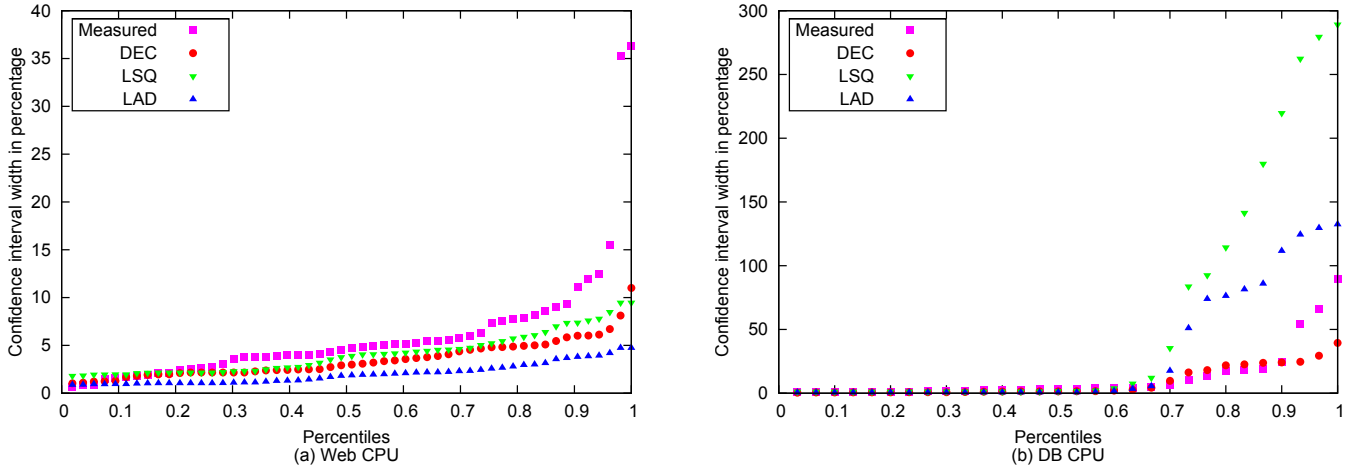

Figure 7. Two-sided confidence interval as percentage of mean demand where LSA had a solution

intervals well within 50% of their corresponding mean values for about 90% of the cases. In contrast, the confidence interval width for LSQ and LAD was within 50% only for about 70% of the cases. Their 90-percentile of confidence interval widths are 215% and 130% of their corresponding mean values, respectively. Figure 7 (b) shows that confidence interval estimates for DB CPU demands for LSQ and LAD diverge from our sample of measured confidence intervals after the 70th percentile, i.e., for about 9 of the 30 cases. DEC's confidence interval distribution is similar to the measured distribution until the 90-percentile of errors and does not diverge in such a large manner. Note that the confidence intervals for LAD are about half as wide as those of LSQ. This is due to the nature of the confidence interval calculation for LAD [4].

We now discuss possible reasons for the poor confidence interval estimates from LSQ and LAD. Closer inspection of the results revealed that 6 of the 9 cases for which LSQ and LAD have confidence intervals that are much wider than the corresponding measured confidence intervals correspond to the six workloads constructed to study the impact of multicollinearity. As mentioned previously, it is well known that the adverse impact of multicollinearity manifests itself as a combination of high prediction errors *and* very wide confidence intervals. However, 3 of the 9 cases had very wide confidence intervals *in spite* of having low prediction errors. Due to the low prediction errors, these represent workloads for which multicollinearity had very little impact. A similar behaviour was also observed while including those cases for which DEC did not provide an exact match. 8 of those 70 cases have low prediction errors but very wide confidence intervals. For such cases the wide confidence intervals are likely a result of violations of the distributional assumptions underlying the confidence interval calculations[2]. In particular, it is likely that these violations are due to the confounding of demand distribution information with measurement error that is caused by non-deterministic resource demands. As discussed previously, confidence intervals for DEC's predictions can be obtained with certainty due to the central limit theorem. Consequently, as evident from Figure 7, they are more robust than the confidence intervals obtained with LSQ and LAD.

[2] To verify this further we carried out the standard Quantile-Quantile plot visual test [7] for the normality of errors in LSQ. The test indicated that the normality assumption deviates significantly in several error regions.

6. SUMMARY AND CONCLUSIONS

This paper introduced our newly proposed Demand Estimation with Confidence (DEC) for estimating the resource demands of services that may be implemented by multi-tier systems. The technique differs from related work in that it predicts the aggregate resource demand of new workload mixes directly rather than by taking the product of the desired mix and per-function demand estimates. We evaluated the technique using a measurement based case study that employed an e-commerce benchmark system. For the cases considered in detail, the CPU demands being predicted varied by a factor of up to 265 depending on workload mix. This demonstrates the importance of accurate demand prediction for system sizing, capacity planning, costing and pricing, and change impact analysis exercises.

We found that DEC provides results as accurate and in many cases more accurate than the Least Squares (LSQ) and Least Absolute Deviation (LAD) regression techniques. It clearly outperforms these approaches when the measurement data suffers from multicollinearity and outperforms them for many other cases as well. Furthermore, DEC has a confidence interval calculation that is simple and not impacted by the distribution of per-function demands. Regression techniques assume such demands are deterministic, which is unlikely for computer system applications. Our results suggest that this results in very wide confidence intervals that provide poor guidance regarding the validity of the demand estimates. DEC's reported confidence intervals are close to observed confidence intervals obtained from measurements.

The work presented in this paper is novel in that it provides the first viable alternative to regression that we are aware of as a resource demand estimation technique for computer systems that provides both demand estimates and robust confidence interval estimates. Together the estimates can help service providers assess the risks of providing services to new customers based on customer-specific workloads.

DEC can be adapted in a straightforward manner to address the problem of estimating resource demands for a production system based on historical measurements data collected from the system.

Specifically, demands can be obtained for various mixes based on the measurement data. DEC can then use this demand-workload mix mapping to offer predictions for new mixes.

DEC has several drawbacks. A sufficient number of benchmarks must be created and evaluated to enable resource demand predictions for a wide variety of workload mixes. Preparing a set of benchmarks would benefit from the concept of benchmark design [8] to ensure predictive coverage of the workload mix space. Furthermore, there may be many different sets of benchmarks that can be used to mimic a new workload mix. Each may lead to different demand estimates.

Our future work includes further evaluation of the method for different systems. For cases where an exact match is not found, we will automate the process of determining an LP formulation that achieves the best possible improvements in prediction accuracy. We will also explore generalizations of the technique that use historical measurements from production systems as input. We also intend to compare the technique with appropriate machine learning algorithms such as support vector machines [21]. Furthermore, our work will focus on demonstrating how DEC can be deployed to handle systems whose demands for a given workload mix shift with time. Finally, challenges introduced by systems that have load dependency in demands will be addressed.

7. ACKNOWLEDGMENTS

This work was financially supported by the Natural Sciences and Engineering Research Canada (NSERC) and Hewlett Packard Labs.

8. REFERENCES

[1] Amazon Elastic Compute Cloud (Amazon EC2), http://aws.amazon.com/ec2/

[2] C. Amza, A. Chanda, A. L. Cox, S. Elnikety, R. Gil, K. Rajamani, W. Zwaenepoel, E. Cecchet, and J. Marguerite. "Specification and implementation of dynamic web site benchmarks", Proceedings of the *Fifth IEEE Workshop Workload Characterization (WWC-5)*, November 2002.

[3] Y. Bard and M Shatzoff, "Statistical Methods in Computer Performance Analysis", *Current Trends in Programming Methodology*, Vol. 3, Prentice-Hall, Englewood Cliffs, N.J.

[4] Y. Dodge, and J. Jureckova, *Adaptive Regression*, Springer, 2000.

[5] N. R. Draper and H. Smith. *Applied Regression Analysis*, John Wiley Sons, 1998.

[6] J. J. Dujmovic, "Universal Benchmark Suites", In proceedings of the *IEEE MASCOTS Conference*, pp. 197-205, 1999.

[7] Engineering Statistics Handbook, http://www.itl.nist.gov/div898/handbook/

[8] R. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Eesign, Measurement, Simulation, and Modeling*, John Wiley & Sons, 1991.

[9] D. Krishnamurthy, "Synthetic Workload Generation for Stress Testing Session-Based Systems", PhD Thesis, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, 2004.

[10] D. Krishnamurthy, J. A. Rolia, and S. Majumdar, "A Synthetic Workload Generation Technique for Stress Testing Session-Based Systems", *IEEE Transactions on Software Engineering*, Vol. 32, No. 11, pp. 868-882, November 2006.

[11] U. Krishnaswamy and D. Scherson, "A Framework for Computer Performance Evaluation using Benchmark Sets", *IEEE Transactions on Computers*, Vol. 49, No. 12, pp. 1325-1338, December 2000.

[12] T. Kubokawa and M. Srivastava, "Improved Empirical Bayes Ridge Regression Estimators under Multicollinearity", *Communications in Statistics – Theory and Methods*, Vol. 33, No. 8, pp. 1943-1973, December 2004.

[13] Y. Lu, T. Abdelzaher, C. Lu, L. Sha, and X. Liu," Feedback Control with Queuing-Theoretic Prediction for Relative Delay Guarantees in Web Servers," In proceedings of the *IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 208, 2003.

[14] D. Menasce, "Computing Missing Service Demand Parameters for Performance Models." In proceedings of *CMG 2008*, pp. 241-248, 2008.

[15] D. Mosberger and T. Jin, "httperf – A Tool for Measuring Web Server Performance", *ACM SIGMETRICS Performance Evaluation Review*, Vol. 26, No. 3, pp. 31-37, 1998.

[16] J. Neter, M. H. Kutner, C. J. Nachtsheim, and W. Wasserman, *Applied Linear Statistical Models,* Irwin, 1996.

[17] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi, "CPU Demand for Web Serving: Measurement Analysis and Dynamic Estimation", *Performance Evaluation*, Vol. 65, No. 6-7, pp. 531-553, June 2008.

[18] J. Rolia and V. Vetland, "Correlating Resource Demand Information with ARM Data for Application Services", Proceedings of the *International Workshop on Software and Performance (WOSP)*, pp. 219-230, 1998.

[19] J. Rolia and V. Vetland, "Parameter Estimation for Performance Models of Distributed Application Systems", Proceedings of the *CASCON Conference*, pp. 54-63, 1995.

[20] SAP Business by Design, http://www.sap.com/solutions/sme/businessbydesign/index.epx

[21] A. J. Smola and B, Scholkopf, "A Tutorial on Support Vector Regression", *Statistics and Computing*, Vol. 14, No. 3, pp. 199-222, August 2004.

[22] C. Stewart, T. Kelly, and A. Zhang, "Exploiting Nonstationarity for Performance Prediction", *ACM SIGOPS Operating Systems Review*, Vol. 41, No. 3, pp. 31-44, 2007.

[23] X. Sun, "Estimating Resource Demands for Application Services", M. Sc. Thesis, Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada, 1999.

[24] TPC-W benchmark, http://www.tpc.org/tpcw/default.asp

[25] M. Woodside, T. Zhen, and M. Litoiu, "Service system resource management based on a tracked layered performance model," In proceedings of the *International Conference on Autonomic Computing*, pp. 175-184, 2006.

[26] Q. Zhang L. Cherkasova, N. Mi, and E. Smirni, "A Regression-Based Analytic Model for Capacity Planning of Multi-Tier Applications", In proceedings of *IEEE International Conference on Autonomic Computing*, June 2007.