Estimating Service Resource Consumption From Response Time Measurements

Stephan Kraft*, Sergio Pacheco-Sanchez†, Giuliano Casale, Stephen Dawson
SAP Research
CEC Belfast, UK
{firstname.lastname}@sap.com

ABSTRACT

We propose a linear regression method and a maximum likelihood technique for estimating the service demands of requests based on measurement of their response times instead of their CPU utilization. Our approach does not require server instrumentation or sampling, thus simplifying the parameterization of performance models. The benefit of this approach is further highlighted when utilization measurement is difficult or unreliable, such as in virtualized systems or for services controlled by third parties. Both experimental results from an industrial ERP system and sensitivity analyses based on simulations indicate that the proposed methods are often much more effective for service demand estimation than popular utilization based linear regression methods. In particular, the maximum likelihood approach is found to be typically two to five times more accurate than utilization based regression, thus suggesting that estimating service demands from response times can help in improving performance model parameterization.

1. INTRODUCTION

Performance models are fundamental to predict the scalability of software and hardware systems, either by analytical methods or by simulation [9]. Although many modeling and evaluation techniques exist to obtain performance predictions [3], there exist few resource consumption estimation methods that can provide good estimates of the service demand placed by requests at system resources [12, 13, 17, 8, 16, 5, 11]. Service demands are the key parameters needed to specify performance models, thus their accurate estimation is of paramount importance for defining models that are both representative and robust. In this paper, we propose the first estimation methods that use only measured response times of requests, rather than CPU utilization samples, to estimate service demands for a multiclass workload. Response time is intended as the end-to-end time for completing a request, thus including delays due to resource contention. We assume that the workload is composed of several classes (i.e., transactions types) and focus on the problem of estimating mean service demands for each class from response time measurements.

We first study a real-world industrial ERP application showing that the proposed response time based approach can result in much improved performance predictions compared to parameterizations provided by standard methods based on linear regression of utilization measurements [12, 13, 16, 5, 11]. In addition to these observations, we note that our approach is more widely applicable than utilization based methods, since response times are often logged directly by applications and can also be obtained by an external observer. Conversely, utilization measurement requires access to the operating system and CPU sampling instrumentation that should not interfere with normal system activities or tamper the measurement while still returning reliable samples. For instance, in virtualized environments, accurate utilization sampling is more difficult, since it requires to filter hypervisor overheads [6]. Additionally, response times may be easily available also for systems that are owned by third parties, such as web service providers, which do not publicly expose utilization data for their servers and yet have performance levels that should be accounted for in capacity planning.

The proposed service demand estimation methods focus on systems with first-come first-served (FCFS) scheduling and are based on linear regression and on the classic maximum likelihood (ML) estimation approach. The response time based linear regression (RR) uses exact equations which relate the mean response time of requests to the queue-length seen on arrival to the system, which can be both easily obtained from application logs. Instead, the ML approach considers in the estimate the entire distribution of the measured response times, thus achieving increased estimation accuracy. Using phase-type distributions, we formulate a computational method to evaluate the likelihood values and estimate resource consumption by an optimization program. Numerical results illustrate the good accuracy of the two estimation techniques and their competitiveness with respect to utilization based approaches, which are found to produce significant errors in many cases where the response time based approach achieves very good accuracy. Furthermore, an intrinsic advantage of the proposed response time approach is also that response times at a server depend on

*Stephan Kraft is also affiliated with Queen's University, Belfast, UK.
†Sergio Pacheco-Sanchez is also affiliated with University of Ulster, Coleraine, UK.

VALUETOOLS 2009 October 20-22, 2009 - Pisa, Italy

all the latency degradations incurred by requests within the system, therefore they are inherently more comprehensive descriptors of performance as they also account for bandwidth, memory, and I/O contention delays. Very rarely, these components are all directly accounted in models, yet they can be critical performance drivers. An advantage of the proposed response time based approach is that the service demands estimated from response times readily account for these delays, which are otherwise ignored by the utilization based estimation approach.

The remainder of this paper is organized as follows. In Section 2 we describe previous work in the area of service demand estimation. Section 3 presents a motivating example on the effectiveness of response time based estimation using data from an industrial ERP application. The RR and ML methods are presented in Section 4 and validated in Section 5 using simulations and sensitivity analyses. Finally, Section 6 draws conclusions and discusses future work.

2. PREVIOUS WORK

Existing techniques for characterization of resource consumption are based on direct data measurement and statistical inference. Direct data measurement techniques propose to install performance probes in the system possibly at different levels, depending on the layers to be analyzed (e.g. application, middleware, kernel). For example, Magpie [1] automatically extracts system workload characteristics using low-overhead instrumentation. The monitor first records fine-grained events generated by kernel, middleware, and applications. Then, it correlates these events to the control flow and to the resource consumption of the requests. Although effective, it is not always possible to adopt instrumentation based approaches of this type in production systems, thus statistical inference is often preferred. The two methods presented in this paper fall into the area of statistical inference techniques, which calibrate performance model parameters from aggregate statistics such as CPU utilization. In [12], linear regression is used for demand estimation and is found accurate with less than 10% error with respect to simulation data. However, regression fails when there is not enough variability in the observed data. Later on, [13] studies the precision of linear regression using simulation of different service demand distributions and it is shown to decrease as the service demand variance grows. In [17], it is investigated the use of an extended Kalman filter to adjust the estimated parameters based on utilization and response time measurements. The work in [8] proposes instead service demand estimation from utilization and end-to-end response times simultaneously: the problem is formulated as quadratic optimization programs based on $M/GI/1/PS$ formulas; results are in good agreement with experimental data. The study in [16] presents a regression based approximation of the CPU demand of customer transactions, which is later used to parameterize a queueing network model where each queue represents a tier of the web application. It is shown that such an approximation is effective for modeling different types of workloads whose transaction mix changes over time. Moreover, [5] presents an optimization based inference technique that is formulated as a robust linear regression problem that can be used with both closed and open queueing network performance models. It uses aggregated measurements (i.e. system throughput and utilization of the servers), commonly retrieved from log files, in order to estimate service demands. The work in [11] considers the problem of dynamically estimating CPU demands of diverse types of requests using CPU utilization and throughput measurements. The problem is formulated as a multivariate linear regression problem and accounts for multiple effects such as data aging. Recently, [15] explores the problem of inferring workload classes automatically from high-level measurement of resources (e.g. request rate, total CPU and network usage) using a machine learning technique known as Independent Component Analysis (ICA). Compared to the work above, the present paper elaborates for the first time the estimation of service demands without knowledge of server utilization. In addition, since previous work has pointed out limits of the linear regression approach [13, 11], we propose for the first time to adopt the maximum likelihood estimation approach to improve accuracy in the service resource consumption estimation.

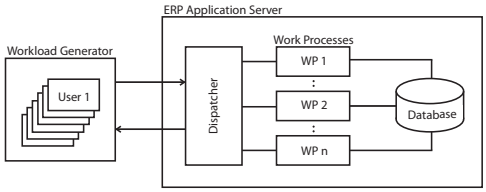


Figure 1: SAP application server architecture.

3. MOTIVATING EXAMPLE

We show an example of service demand estimation in a real world Enterprise Resource Planning (ERP) application that motivates the response time based estimation approach.

3.1 ERP System Description

The system we consider is the ERP application of SAP Business Suite [14] in a two-tier configuration composed of an application server and a database server installed on the same virtual machine. No other virtual machines run on the physical machine. The system is stress-tested using a workload of sales and distribution operations, which include standard business transactions such as creation and listing of sales orders and invoices. The workload generator is of closed-type: requests are issued by a fixed group of N users; after the completion of a request submitted by a user, it is waited an exponentially-distributed think time with mean $Z = 10s$ before submitting a new request to the system. All users cyclically submit the same sequence of requests to the ERP application. After submission, the SAP application server has to process requests from these multiple concurrent users using also information extracted from a database, see Figure 1. To support this, the application server uses a dispatcher, which buffers the requests in a queue and transfers them to the work processes, which are independent operating system threads serving the ERP requests in parallel. The work processes are executed as either dialog processes which execute interactive programs or update processes which perform database changes. We consider the problem of estimating the mean service demands of the requests at the work process based either on utilization or response time measurements.

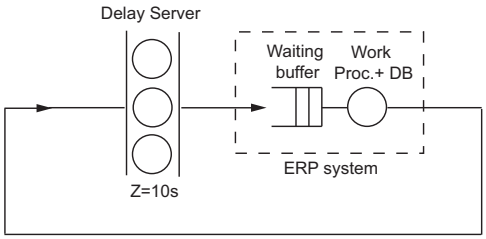


Figure 2: Simple queueing model of the SAP ERP application for the configuration with one work process considered in the motivating example.

Users	U_{meas}	D_{est}^{cpu}	D_{est}^{twp}	D_{est}^{rsp}
25	0.42	0.154	0.161	0.200
50	0.73	0.139	0.163	0.181
75	0.92	0.133	0.220	0.170
100	0.96	0.128	0.330	0.156

Table 1: Estimates of the mean service demand $E[D]$ using linear regression of the cumulative times for CPU and DB processing (D_{est}^{cpu}), of the total time spent in the work process after admission in the system (D_{est}^{twp}), and of the measured response times (D_{est}^{rsp}). All times are expressed in milliseconds.

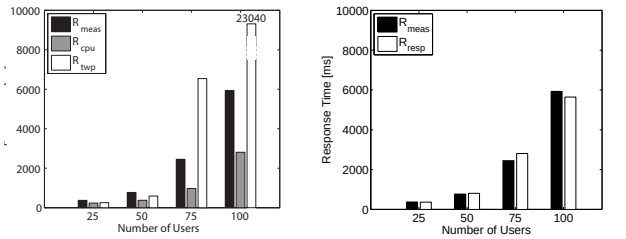
3.2 Service Demand Estimation

We use the ERP application described above to illustrate common difficulties in estimating service demands using CPU utilization measurements. We have run a set of experiments on the ERP application using a configuration with a single work process, think times with mean $Z = 10s$, and an increasing number of users in the range $N \in [25, 100]$ corresponding to an interval of system CPU utilizations $U_{meas} \in [0.44, 0.99]$. For each experiment, we have used the internal monitor of the ERP application to collect several performance measures:

- the total CPU consumption T_{meas}^{cpu} of each request
- the total time T_{meas}^{db} to provision data from the DB
- the total time T_{meas}^{twp} spent in the work process
- the response time R_{meas} of each request

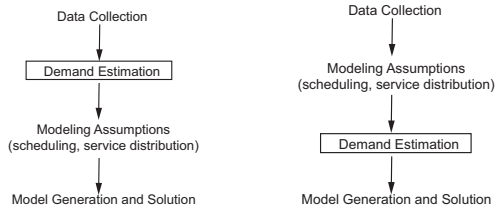
All measurements are obtained with resolution of one millisecond. The aim of this section is to show that the parameterization of a basic queueing model that describes the performance of the ERP application can be significantly improved if the response times R_{meas}, instead of utilization measurements, are used to estimate resource consumption of requests. The performance model considered for this task is a basic $M/M/1//N$ model, which is an $M/M/1$ queue with requests generated by a finite population of N users. This can also be seen as a closed queueing network composed of a delay station, which models user think times, followed by a queueing station on a closed-loop: the waiting buffer represents the admission control in the ERP application; the server models the resource consumption of the requests when executed in the work process. Figure 2 illustrates the structure of the $M/M/1//N$ queueing model. In all considered experiments, we have found the utilization of the database server to be always less than 5%, thus queueing effects at the database tier are negligible and T_{meas}^{db} can be used as a component of the service demand in the work process. Both think times and service demands in the queueing model are assumed exponentially distributed: we have verified that this assumption is a good approximation of the actual service demand distribution in the ERP system. In fact, the coefficient of variation of the measured service demands (i.e., the ratio between standard deviation and mean) ranges over different experiments in $CV = [1.02, 1.35]$, whereas for the theoretical exponential is $CV = 1$; consistently with the model assumptions, all think times for the validation experiments have been generated using an exponential distribution as well. The challenge of the $M/M/1//N$ model parameterization is to determine the mean service demand $E[D]$ of the requests at the server such that the response times predicted by the model (R) match accurately those measured of the real system (R_{meas}) for all possible numbers of users N. In particular, due to the large role of caching which affects the behavior of the ERP system very different at light and heavy loads, we specify the service demands as a function of the number of users N in the model, i.e., $E[D] \equiv E[D](N)$. This estimation approach is routinely used in modeling complex software systems.

Utilization Approaches. To determine $E[D]$, we consider three approaches. The first estimate D_{est}^{cpu} of $E[D]$ is obtained by assuming that the sum $T_{meas}^{cpu} + T_{meas}^{db}$ can fully explain the resource consumption of a request and thus the D_{est}^{cpu} estimate is obtained by linear regression of $T_{meas}^{cpu} + T_{meas}^{db}$ against the number of completed requests. This estimation technique is essentially the utilization based regression approach studied in previous work, e.g., [16]. In fact, we have verified that the sum of the $T_{meas}^{cpu} + T_{meas}^{db}$ values captures more than 95% of the busy time of the CPU, thus it is essentially a utilization measurement. A second estimate D_{est}^{twp} is obtained by regression of the time T_{meas}^{twp} spent in the work process against the number of completed requests. This corresponds to assuming that the residence time in a work process is equivalent to its service demand. Theoretically, this assumption is satisfied if the ERP application does not perform other service activities while processing the current request. The results presented below indicate that this is not the case although there is a single work process but, as typical of complex software systems, detailed information on the internal behavior of the application is not available to explain the nature of these concurrent service activities. Note that D_{est}^{twp} may be seen as an upper bound on the actual service demand of the requests, while D_{est}^{cpu} may be interpreted as a lower bound, since it ignores other components of the service demands such as I/O delays. Estimates and model prediction accuracy results for the $M/M/1//N$ queueing models parameterized respectively with $E[D] = D_{est}^{cpu}$ and $E[D] = D_{est}^{twp}$ are reported in Table 1 (third and fourth columns) and in Figure 3(a). The queueing model has been solved using the JMT open-source suite [2]. The results in the figure indicate that the model parameterized with D_{est}^{cpu} (resp. D_{est}^{twp}) grossly underestimates (resp. over-estimates) the measured response time of the ERP system. In particular, the estimate R^{cpu} obtained from the D_{est}^{cpu} parameterization suggests that the CPU processing time and the DB data provisioning times are alone insufficient to fully explain the service demand of the requests. For instance, overheads due to memory, I/O access, and contention at the processor from other OS pro-



(a) Utilization approach (b) Response time approach

Figure 3: Comparison of response time predictions.



(a) Utilization approach (b) Response time approach

Figure 4: Comparison of modeling methodologies.

cesses may contribute to this under-estimation. Conversely, the very large response times predicted by the D_{est}^{twp} parameterization suggest that other queueing effects not captured explicitly by the model in Figure 2 are responsible for additional degradation during the service period in the work process. In this case, one would generally use a more detailed model of the system's internal behavior, adding waiting queues to model the I/O subsystem and possibly request classes to better capture system performance. However, in presence of limited information on the system it may not be possible to increase model accuracy and one then aims to obtain a simple queueing model that best captures the observed behavior regardless of the unknown internal activities that have not been accounted for. As proved by the above results, this may not be done effectively using utilization based service estimation approaches that focus only on CPU consumption.

Response Time Approach. To address the limitation of the current service demand estimation approach based on utilization measurements, we propose to directly parameterize queueing models based on the response time measurements R_{meas}. The fundamental idea is to estimate service demands in order to best match the distribution or moments of the measured response times. This requires an important paradigm shift in the modeling methodology as illustrated in Figure 4. Since the response times depend on several system properties related to scheduling or service demand distribution, in the proposed methodology one should take assumptions on scheduling or general form of the distribution prior to starting the service demand estimation activity, thus the returned service demands depend on the characteristics of the model in which they will be used, see Figure 4(b). Conversely, Figure 4(a) illustrates the traditional utilization based estimation methodology, where resource consumption is estimated from the utilization, which is independent of model characteristics such as scheduling or service demand distribution. Therefore, in the classic approach, the impact of the modeling assumption is deferred to the model solution step and the service demand estimates are *absolute*, i.e., they represent intrinsic properties of the system that do not depend on the particular type of model to be used. Conversely, our proposed methodology requires preliminary assumptions on the characteristics of the final model to determine the best possible service demand estimates *relatively* to the target model. Since the final goal of the modeling activity is to obtain good agreement between experimental observations and model predictions, we believe that the response time approach has a stronger focus on achieving this goal by returning the best parameters under model assumptions.

Following the above observations, the third service demand estimation technique we consider is based on the proposed response time approach, where we assume that the ERP system can be modeled as a FCFS queue with exponentially-distributed service demands. Under this assumption, a job that finds the system with n waiting requests and one job in service receives a response time distribution given by the distribution of the sum of random variables

$$R = T + D^1 + D^2 + \ldots + D^n + D^{n+1},$$

where T is the residual time before completion of the request in service, D^i for $1 \leq i \leq n$ is the service demand of the ith enqueued job, and $D^{n+1} = D$ is the service demand of the newly arrived request. Note that by definition of exponential distribution $T = D$ and thus the distribution of the random variable R is the convolution of $n + 1$ exponential random variables. By simple derivations it is easy to show that the mean response time is found to satisfy the following relation

$$E[R] = E[D](1 + E[A]), \tag{1}$$

where $E[D]$ is the mean service demand and $E[A]$ stands for the mean queue length observed by a request upon time of arrival. This is also a well-known formula appearing in Mean Value Analysis of single class product-form queueing networks [3]. Equation 1 can be used to estimate the mean service demand of the requests given the knowledge of the mean number of jobs in the ERP system at the time of arrival of a new job. The A value can be easily obtained from the log files of the experiments we have performed which report the time of arrival and departure of requests. The estimate D_{est}^{rsp} of $E[D]$ is then obtained by linear regression of the above formula using a sequence of samples of $E[R]$ and $E[A]$ obtained by averaging R_{meas} and the measured A values for groups of 20 consecutive requests. A graphical illustration of the measured linear relation between $E[R]$ and $E[A]$ in the ERP system is given in Figure 5.

From (1), we evaluate $E[D]$ by linear regression as

$$E[R^i_{meas}] = E[D](1 + E[A^i_{meas}]), \tag{2}$$

where the index i stands for the ith value collected in the measurement activity and $E[R^i_{meas}]$ and $E[A^i_{meas}]$ are the response time and arrival queue-length averaged on 20 consecutive i samples.

Numerical results of the service demand estimation performed following this approach are reported in Table 1 (fifth column). Figure 3(b) shows the accuracy of this approach which is clearly much more effective in producing high-quality predictions compared to the utilization based service demand estimation considered in Figure 3(a). The results of

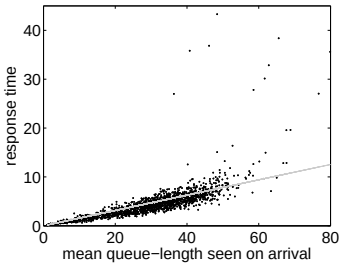


Figure 5: Linear relation between measured response times and measured queue-length seen by a job arriving to the ERP system for $N = 100$ users and aggregation level equal to 20 consecutive requests. Response times are in seconds.

this modeling study clearly motivate the investigation in the remainder of this paper on techniques to estimate service demands starting from response time measurements.

4. ESTIMATION FROM RESPONSE TIMES

Throughout this section, we describe two service demand estimation techniques based only on response time measurements. We assume that the system under consideration can be modeled as a $FCFS$ queue and derive in Section 4.1 an inexpensive linear regression based estimation approach. In Section 4.2, we give a maximum likelihood method, which provides higher accuracy than linear regression by describing the probability of observing exactly the measured data. Throughout the rest of this paper, we assume that the user has provided an input trace with a total of I samples to the algorithms. This consists of a sequence of measured response times R_c for requests of class c and on the corresponding queue-length A_k^c of class-k jobs seen upon arrival by each class-c request. The number of workload classes is henceforth denoted by K.

4.1 Linear Regression

We now show that the linear relation given in (2) does not change if the assumptions on exponential service are dropped or we use multiple classes. The formula obtained below is essentially a variant of the expression developed for approximating FCFS queues in non-product-form queueing networks with multiple classes [3].

PROPOSITION 1. *In a $GI/GI/1/FCFS$ queue, the mean response time of a request of class c is*

$$E[R_c] = E[T_c] + \sum_{k=1}^{K} E[D_k](1_{k,c} + E[A_k^c]), \qquad (3)$$

where

$$1_{k,c} = \begin{cases} 1 & k = c \\ 0 & k \neq c, \end{cases}$$

T_c is the residual time before completion of the job in execution at the instant of arrival of the class-c request, K is the number of request classes, $E[D_k]$ is the mean service demand of class k, and $E[A_k^c]$ is the mean number of jobs of class k queueing in the system excluding both the newly arrived request and the job currently in service.

PROOF. In a $GI/GI/1$ queue with FCFS scheduling, the random variable describing the response time of a request of class c arriving when the system has $n+1$ enqueued jobs (n waiting plus the job in service) is $R_c = T_c + D^1 + D^2 + \ldots + D^n + D_c$, where T_c is the residual time before completion of the current job in service, D^i for $1 \leq i \leq n$ is the service demand of the ith enqueued job, and D_c is the service demand of the newly arrived request. Taking expectations conditioned on the arrival queue-length being $A^c = n$ gives

$$E[R_c|A^c = n] = E[T_c|A^c = n] + \sum_{i=1}^{n} E[D^i|A^c = n] + E[D_c].$$

Now, expressing the service demands as a function of the job class we get

$$E[D^i|A^c = n] = \sum_{k=1}^{K} E[D_k]\mathbb{P}[C(i) = k|A^c = n],$$

where $C(i)$ is the class of the job in position i, $1 \leq i \leq n$. Substituting yields

$$E[R_c|A^c = n] = E[T_c|A^c = n] + \sum_{i=1}^{n}\sum_{k=1}^{K} \mathbb{P}[C(i) = k|A^c = n]E[D_k], \qquad (4)$$

and we can now compute the mean response time as

$$\begin{aligned} E[R_c] &= \sum_{n=1}^{+\infty} E[R_c|A^c = n]\mathbb{P}[A^c = n] \\ &= E[T_c] + \sum_{k=1}^{K} E[D_k] \sum_{n=1}^{+\infty} \Big(1_{k,c} + \sum_{i=1}^{n} \mathbb{P}[C(i) = k|A^c = n]\mathbb{P}[A^c = n]\Big) \\ &= E[T_c] + \sum_{k=1}^{K} E[D_k]\Big(1_{k,c} + \sum_{n=1}^{+\infty} E[A_k^c|A^c = n]\mathbb{P}[A^c = n]\Big) \\ &= E[T_c] + \sum_{k=1}^{K} E[D_k](1_{c,k} + E[A_k^c]), \qquad (5) \end{aligned}$$

where we noted that $\sum_{i=1}^{n} \mathbb{P}[C(i) = k|A^c = n]$ is the mean number $E[A_k^c|A^c = n]$ of class k jobs waiting in the queue upon arrival by a request of class c. □

Estimates of the class demands $E[D_k]$ can be obtained from (3) using regression methods such as the non-negative least-squares algorithm[1] and confidence intervals on the estimates can be generated using standard formulas [3]. Equation (3) can be used in linear regression for estimating the mean service demands $E[D_k]$ if the per-class response times R_c and arrival queue-lengths A_c are known. In general, $E[T_c]$ can be difficult to measure directly, therefore approximations are needed to estimate this quantity. We focus on the rest of this paper on the standard Poisson arrival approximation $T_c = D_k$ and we only outline an alternative approximation scheme for general service workloads. Due to the lack of exact results for the residual time expression in $GI/GI/1/FCFS$ queues, we propose to use the renewal-theory expression for the residual time

$$E[T_c] = \frac{E[D_c]}{2}(1 + CV_c^2), \qquad (6)$$

[1]An implementation of non-negative least-squares is provided by the MATLAB function `lsqnonneg`.

where CV_c is the coefficient of variation of the service demand distribution of class c. This is clearly an approximation in $GI/GI/1/FCFS$ queues which becomes an exact expression when the arrival stream is a Poisson process. We also remark that, if the CV_c value is not known a priori, it is still possible to evaluate (3) for different CV_c values that are assumed feasible for the system under study. Then it is selected as best estimate of the CV_c values the set that produces the minimum mean residual error in the least-squares solution of (3), which is thus the group of coefficient of variations that best fits the observations. This outlines a general schema for possible application of (3) to non-exponential service distributions.

4.2 Maximum Likelihood Estimation

Maximum likelihood estimation is a classic approach for inferring statistics of random variables based on analytical expressions of the probability of observing a certain sample path [3]. Within the scope of mean service demand estimation, maximum likelihood can be formulated as follows. Let R^i denote the ith observed response time, for all $1 \leq i \leq I$, where I is the total number of measured response times. We seek for the mean service demand $E[D_k]$ for each class $1 \leq k \leq K$ such that the probability of observing the sequence of measured response times $R^1, \ldots, R^i, \ldots, R^I$ is maximal. Formally, this can be expressed as finding the set of mean demands $E[D_1], \ldots, E[D_K]$ which solves the maximization problem

$$\max_{E[D_1],\ldots,E[D_K]} \mathbb{P}[R^1, \ldots, R^i, \ldots, R^I | E[D_1], \ldots, E[D_K]], \quad (7)$$

subject to $E[D_k] \geq 0$, for all classes k. Assuming the response time as independent random variables the joint probability expression in (7) is simplified into the product

$$\max_{E[D_1],\ldots,E[D_K]} \prod_{i=1}^{I} \mathbb{P}[R^i | E[D_1], \ldots, E[D_K]],$$

and taking the logarithm to equivalently express the products as a summation we get

$$\max_{E[D_1],\ldots,E[D_K]} \mathbb{L}(E[D_1], \ldots, E[D_K]) \quad (8)$$

where the argument is now the *likelihood function*

$$\mathbb{L}(E[D_1], \ldots, E[D_K]) = \sum_{i=1}^{I} \log \mathbb{P}[R^i | E[D_1], \ldots, E[D_K]].$$

The challenge is to obtain an expression for the likelihood function which is representative of the problem under study and for which the maximum can be computed efficiently. It is also important that this expression is analytically tractable, since optimization can be otherwise too expensive. Note in particular that we focus here on the estimation of the *mean* service demand, since this is the fundamental parameter required for the specification of standard capacity planning models based on product-form queueing models [3].

4.2.1 Likelihood Function

The method we have considered to derive the likelihood function characterizes both the service and response times of the system under study by phase-type distributions [10], which enjoy efficient analytical expressions for their evaluation [7]. To avoid unnecessary complexity, we first describe the approach for the case of exponentially-distributed service demands and then discuss how this extends to the case of general service demands. Consider a request that arrives to the system when $n_1, \ldots, n_K$ jobs are enqueued for each class, including the job currently in service. Assume exponential service demands for all classes and denote by $\mu_k = 1/E[D_k]$ the mean service rate of class k. Then, the distribution of response times for the kth request is given by the sum of n_1 exponential service demands with rate μ_1, n_2 exponential service demands with rate μ_2, and so forth for all K classes. Therefore, this can be modeled as the time to absorption in a Markov chain with $n = n_1 + \ldots + n_K$ states, each one representing the waiting time due to a job service. For instance, if there are $K = 2$ classes and the queue seen on arrival is $n_1 = 3$ and $n_2 = 2$ including both the arrived job and the one in service, the time to absorption can be described by the phase-type distribution with the following $(\mathbf{D}_0, \mathbf{D}_1)$ representation [7]

$$\mathbf{D}_0 = \begin{bmatrix} -\mu_1 & \mu_1 & 0 & 0 & 0 \\ 0 & -\mu_1 & \mu_1 & 0 & 0 \\ 0 & 0 & -\mu_1 & \mu_1 & 0 \\ 0 & 0 & 0 & -\mu_2 & \mu_2 \\ 0 & 0 & 0 & 0 & -\mu_2 \end{bmatrix} \quad (9)$$

$$\mathbf{D}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \mu_2 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (10)$$

The $\mathbf{D}_0$ matrix has off-diagonal elements in position (i, j) representing a transition from state i to state j that does not lead to absorption, while $\mathbf{D}_1$ elements are transitions associated to absorption. The diagonal of $\mathbf{D}_0$ is such that $\mathbf{D}_0 + \mathbf{D}_1$ is an infinitesimal generator of a Markov chain describing the evolution of the active state over time.

The $(\mathbf{D}_0, \mathbf{D}_1)$ representation allows to compute efficiently the probability that a request receives a response time R^i. In fact, it is easily seen that the time to absorption is described by the probability density of the phase-type distribution $(\mathbf{D}_0, \mathbf{D}_1)$. From basic theory of absorbing Markov chains this is readily given by by [7]

$$\mathbb{P}[R^i | E[D_1], \ldots, E[D_K]] = \vec{\pi}_e e^{\mathbf{D}_0 R^i} (\mathbf{D}_1) \vec{e}, \quad (11)$$

where $\vec{e} = (1, 1, \ldots, 1)^T$, $\vec{\pi}_e = \vec{\pi}_e(-\mathbf{D}_0)^{-1}\mathbf{D}_1$ is a row vector with elements $(\vec{\pi}_e)_j$ representing the initial state of a request immediately after absorption, and we have noted that knowledge of $E[D_1], \ldots, E[D_k]$ is equivalent to knowing the rates $\mu_1, \ldots, \mu_K$. Since there is a single transition in $\mathbf{D}_1$ leading to absorption, it is immediate to conclude that in (11) it is always $\vec{\pi}_e = (1, 0, \ldots, 0)$ and the main cost of evaluating (11) is the computation of the matrix exponential function, which can be approximated in an efficient way by the uniformization technique [4] or by Padé expansion (e.g., in MATLAB). The approach discussed above generalizes immediately to non-exponential service demands that can be approximated as phase-type. Below we sketch the concept. Let $(\mathbf{S}_0^k, \mathbf{S}_1^k)$ denote the phase-type representation of the service demands of class k. As a matter of example, for the Erlang-2 distribution this is

$$\mathbf{S}_0^k = \begin{bmatrix} -\mu_k & \mu_k \\ 0 & -\mu_k \end{bmatrix} \quad \mathbf{S}_1^k = \begin{bmatrix} 0 & 0 \\ \mu_k & 0 \end{bmatrix}$$

which implies a probability vector $\vec{\pi}_e^k = (1, 0)$.

For the same example of the exponential case, we have that if the service is Erlang-2 distributed we can express the response time spent in the queue as

$$\mathbf{D}_0 = \begin{bmatrix} -\mathbf{S}_0^1 & \mathbf{S}_1^1 & 0 & 0 & 0 \\ 0 & -\mathbf{S}_0^1 & \mathbf{S}_1^1 & 0 & 0 \\ 0 & 0 & -\mathbf{S}_0^1 & \mathbf{S}_1^1 \vec{e}\vec{\pi}_e^2 & 0 \\ 0 & 0 & 0 & -\mathbf{S}_0^2 & \mathbf{S}_1^2 \\ 0 & 0 & 0 & 0 & -\mathbf{S}_0^2 \end{bmatrix} \quad (12)$$

$$\mathbf{D}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \mathbf{S}_1^2 \vec{e}\vec{\pi}_e^1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (13)$$

where $\vec{\pi}_e^k = \vec{\pi}_e^k(-\mathbf{S}_0^k)^{-1}\mathbf{S}_1^k$. The $\vec{e}\vec{\pi}_e^k$ terms ensure proper initialization of the phase-type distribution, since in general the state space sizes of $(\mathbf{S}_0^1, \mathbf{S}_1^1)$ and $(\mathbf{S}_0^2, \mathbf{S}_1^2)$ are different. The above expression can be immediately evaluated for computing the likelihood function by (11) and the only difference is that this involves larger matrices. Finally, we remark that for service demand distributions specified by several parameters, such as the hyper-exponential distribution, the above approach requires either to fix a priori all moments except the mean, e.g., by guessing the variability of the service process, or to integrate these additional parameters as unknown variables in the ML program (8). This changes the structure of the ML estimation problem to evaluating conditional probabilities in the form

$$\log \mathbb{P}[R^i | E[D_k], E[D_k^2], \ldots, E[D_k^M], 1 \leq k \leq K]$$

if the phase-type distributions are uniquely specified by their first M moments $E[D_k^m]$, $1 \leq m \leq M$. For example, the hyper-exponential distribution is fully determined by three parameters, thus the first three moments $E[D_k^m]$ are sufficient to completely specify the distribution. It is also fundamental to remark that, in presence of high-variability distributions which are the most challenging to address [13], it seems more practical to decompose the workload in an increased number of classes, such that the service distribution of each class can be approximated by an exponential. For example, if a workload class has hyper-exponential service demands with density $f(t) = p\mu_1 e^{-\mu_1 t} + (1-p)\mu_2 e^{-\mu_2 t}$ and arrival rate λ, then one may approximate this workload by two classes with exponential service demands $f_1(t) = \mu_1 e^{-\mu_1 t}$ and $f_2(t) = \mu_2 e^{-\mu_2 t}$ and arrival rates $\lambda_1 = p\lambda$ and $\lambda_2 = (1-p)\lambda$, respectively, which allows to focus only on mean service demand estimation under simpler exponential assumptions. For this reason, we believe that the focal point of the service demand analysis is to be able to estimate reliably exponentially-distributed service demands in a multiclass setting, a goal that remains very challenging for all existing demand estimation methods as we show in the next section.

5. VALIDATION EXPERIMENTS

In this subsection we study the accuracy of the proposed service demand estimation methods based on linear regression and maximum likelihood. We consider utilization and response time traces generated by a queueing system simulator for simulation lengths of 60, 600, and 3600 seconds. The simulator has been written in MATLAB and can log, in addition to standard performance measures, the queue-length seen upon arrival by requests. Inter-arrival times and mean service demands are generated from exponential distributions and their means are assigned to match the utilization levels fixed for the experiments. Surprisingly, we have observed that with multiple service classes even state-of-the-art methods based on utilization sampling have considerable difficulties in estimating correctly the mean service demands even if distributions are exponential. This may be probably attributed to the fact that, assuming a 1 second utilization sample granularity as in most computer systems, the number of samples made available in less than 3600 seconds is not very large. However, since modern systems are moving toward much more dynamic environments where system characteristics change rapidly over time, we believe that reliable service demand estimates should be able to work also with few hundreds or thousands samples. Thus, throughout section 5 we focus on this basic exponential case which remains an open challenge for current estimation techniques. Additionally, we investigate the sensitivity of estimation accuracy to changes in the number of samples.

5.1 Main Validation Scenario

Our main experimental run consists of system utilization and response time measurements over a period of 600 seconds[2]. We have run the simulation experiments with low, medium and high server utilization levels $\rho \in \{0.1, 0.5, 0.9\}$, as well as with request classes $K \in \{1, 2, 5\}$. Response time measurements are collected on a per-request basis, while utilization is sampled every second: this is the typical situation in modern systems which easily log response times of individual requests, but cannot probe utilization with fine-grain or even per-request resolution. Demand estimation accuracy is compared for the following methods:

Utilization based regression (UR): this is the state-of-the art service demand estimation approach studied both theoretically [12, 13, 5, 11] and in applications, e.g., [16]. We consider three variants: UR1 performs regression directly on all available samples; UR50 first averages measurements over groups of 50 consecutive samples, then it performs linear regression on these aggregate values; finally, UR30r uses similar averages but on partitions formed by consecutive samples such that 30 requests fall into each partition.

Response-time based regression (RR): this is the estimation technique based on the linear regression formulas in Section 4.1. RR1r uses all available samples, while RR30r uses averages over partitions of 30 requests of the same class. No equivalent of UR50 is defined, since averages over consecutive requests are not easy to interpret with the results in Section 4.1 unless all requests belong to the same class.

Maximum Likelihood (ML). This is an implementation of the algorithm described in Section 4.2 based on the MATLAB optimization toolbox. All available samples are used for ML estimation, thus the information employed is the same as UR1 and RR1r. We return the service demand estimates that correspond to the best likelihood values among two runs: the first run has random initialization points, the second run is initialized with the demand values estimated

[2]The length of 10 minutes is here considered representative of a typical time interval before an online resource management system takes a new decision for tuning system performance.

by the RR1r method. Computational times are generally good, always below very few minutes in the worst case; however, these can be reduced to the desired value by imposing a limit on the number of iterations or a timeout condition for the solver. In this case, ML returns the best found service demand estimates. For each experiment, corresponding to a different choice of the number of classes K and of the utilization ρ, we evaluate estimation accuracy by the error function

$$\Delta = \sum_{k=1}^{K} \frac{1}{K} \left| \frac{E[D_{estimate,k}] - E[D_{exact,k}]}{E[D_{exact,k}]} \right|, \qquad (14)$$

which is the Mean relative error over all classes of the estimated service demands $E[D_{estimate,k}]$ with respect to the exact value $E[D_{exact,k}]$ used in the simulations. For each simulation, the $E[D_{exact,k}]$ value of each class is randomly drawn with uniform distribution ranging in $[0, 1]$. Figure 6 illustrates the mean value $E[\Delta]$ obtained by averaging Δ over 100 simulations with random service demands and having all the same number of classes K and utilization ρ. Similarly, Figure 7 shows the cumulative distribution function (CDF) of Δ over 100 random models for $\rho = 0.5$ and $K = 1, 2, 5$. Detailed comments of the results are given below.

Single class. The single-class scenario is the simplest one, since only a single mean service demand $E[D]$ needs to be estimated. Most methods perform extremely well in this case showing an error around 5% for the different utilization levels, see Figure 6(a). Only UR1 delivers a low quality result featuring error values of more than 30% even at medium load. We explain this effect in terms of insufficient number of events in the sample period which invalidates the linear regression formula, since the sample averages are computed on a subset of values that has too small cardinality (even 1,2 or 3 requests). Consequently, the aggregations used in UR30r and UR50 have a remarkably positive effect on removing the UR1 errors, lowering $E[\Delta]$ to below 5% for UR30r. Also the RR methods deliver high quality results, although the aggregation used in RR30r does not produce appreciable changes in the estimation accuracy. ML displays similar high-quality results for all utilization levels. Finally, we remark that the above observation are also confirmed by the error distribution. Figure 7(a) shows low probabilities of high errors for most of the methods, approximately with a 95th-percentile of 12% for all techniques[3]. Summarizing, the single class case indicates that, except UR1, all methods perform accurately.

Two classes. When moving from single to multi-class workloads the errors grow as shown by Figure 6(b). ML overall has the smallest errors when considering two request classes and delivers errors of less than 8% in medium load ($\rho = 0.5$). The quality of the ML performance becomes obvious when comparing with UR1, where we observed error values of almost 35% for the same utilization level. Quite surprisingly, aggregation of utilization samples does not prove to be as effective as in the single-class scenario and UR30r, as well as UR50 are not very different from UR1. RR1r performs better than the UR methods, supporting our claim that this is a competitive alternative to UR. The RR30r aggregation technique does not seem to work better than RR1r again. We conjecture that the reduced effectiveness of aggregation in the two class case could be due to the shrinking of the number of samples made available to the regression program, that becomes problematic to handle in the multiclass case. Finally, the distribution analysis presents similar results: for $\rho = 0.5$ in Figure 7(b) ML has a 95th-percentile of 17% whereas RR30r and UR30 have 95th percentiles of 54% and 61% respectively.

Five classes. Figure 6(c) shows the fact that the quality of the results significantly changes again with respect to the case of two classes. First we note that the general accuracy decreases (note that the vertical scale is different for all figures) and especially in high load, but this is expected due to the larger number of service demands to be estimated. UR1, UR30r, and UR50 have always rather large errors, with the best result among UR methods being achieved by UR1 in the $\rho = 0.5$ case. Aggregation for UR again does not lead to accuracy increases. RR1r delivers similar bad results as UR1 in medium utilization levels, while RR30r deteriorates the error value. The only method that delivers high-quality results, at least at low and medium utilization levels, is ML (less than 12% error). The heavy load case instead seems very difficult for all methods, with UR1 being slightly better than ML. Finally, the CDF of the $K = 5$ scenario in Figure 7 (c) emphasizes the quality of ML, which achieves producing 95% of errors below 21%. Neither of the remaining methods can perform similarly well: RR30r has a 95th-percentile of 111%, while for UR30 it is 162%. For UR1 (not shown in the figure) it is 56%, thus ML is overall more robust.

Summary. The above results indicate that multiclass estimation within medium sized data sets, corresponding to sets of events in 600 seconds for different utilization values, is a challenging problem. ML is the only method that can perform such estimations robustly in almost all cases, with the exceptions of $\rho = 0.9$ and $K = 5$. Noticeably, the ML results are extremely good with limited information (low utilization). RR1r appears more effective than RR30r on all experiments and generally competitive with the UR methods at low and medium load. Instead, the UR methods prove to be potentially effective, but have a very unpredictable behavior. This makes it difficult to choose which method among UR1, UR30r, and UR50 should be used for estimating service demands in a given system.

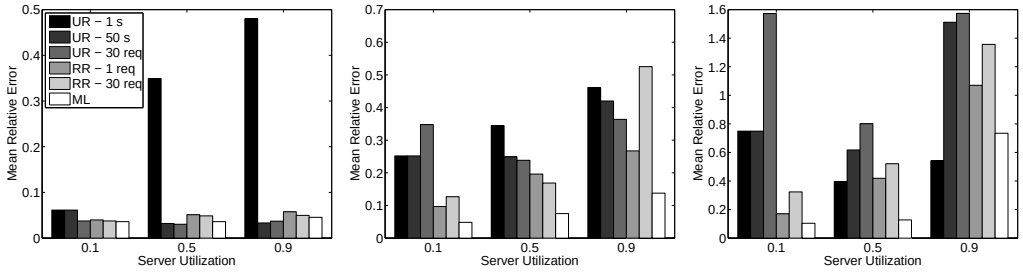
5.2 Sensitivity Analysis with Less Samples

We now explore the sensitivity of the results presented in the previous section to a reduced number of samples corresponding to simulations of 60 seconds. This represents an extreme case which further exacerbates the estimation difficulties already illustrated in the previous section.

Single class. From Figure 8(a), we observe that the Mean relative error remains quite stable for all methods with respect to the results in Figure 6(a). UR1 has again the highest errors of 35% and 49% for the medium and high utilization cases, respectively. Conversely, for the low utilization case all methods perform equally well, with errors ranging between 10% and 12%, which are twice as large compared to the 600 samples case. We note again that all methods with the exception of UR1 perform well with a single class.

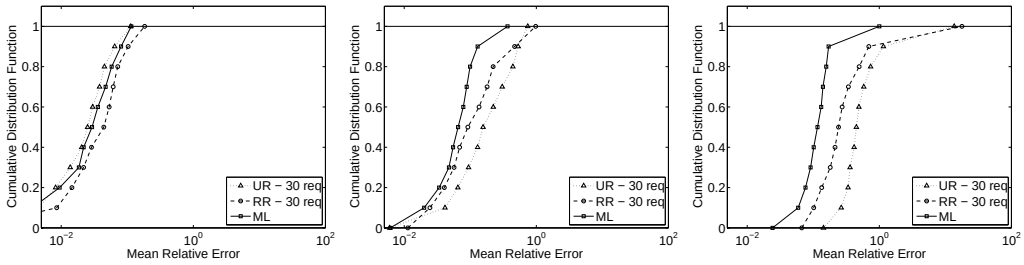
Two classes. In contrast to the single class scenario, the two-class case depicted in Figure 8(b) shows significantly different results. First, there is a substantial increase in the Mean relative error with respect to Figure 6(a), with the

[3]We here only show UR30r, RR30r, and ML for ease of plotting and also because the curves not shown lead to qualitative conclusions in line with the ranking in Figure 6.



(a) Single class (b) Two classes (c) Five classes

Figure 6: Mean relative error $E[\Delta]$ at different server utilization levels for the 600-sample case. The legend shown in figure (a) is identical for figures (b) and (c).



(a) Single class ($\rho = 0.5$) (b) Two classes ($\rho = 0.5$) (c) Five classes ($\rho = 0.5$)

Figure 7: CDF of Mean relative error $E[\Delta]$ in the main validation experiment for the 600-sample case.

exception of UR1, RR1r and ML that continue to be reasonably accurate. ML is the method that best performs across all server utilizations with an error ranging between 20% and 25%. It is instead evident that the aggregation based methods fail dramatically in this case with errors more than 200%, confirming our supposition that the tradeoff between aggregation (i.e., quality of each sample) and total number of available samples can be critical for linear regression.

Five classes. In the five-class case depicted in Figure 8(c), we observe an even higher increase of the Mean relative error. This is justified by the fact that we are considering the most difficult case with several service classes and very limited number of samples. The Mean relative error of the UR and RR methods obtained for all server utilizations is always above 50% and it is frequently above 100%. We conclude from these results that the ML method is the most accurate with many classes and limited information, with errors less than 40% for $\rho = 0.1$ and $\rho = 0.5$.

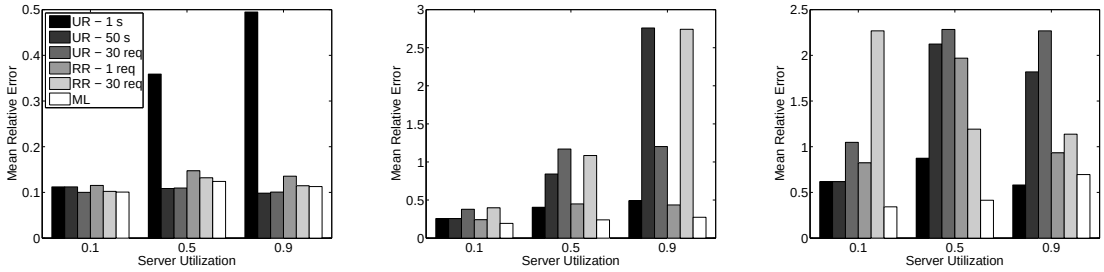
Summary. This analysis has shown that in presence of limited data the choice of estimating many workload classes can have bad effects on the accuracy of the methods. Yet it is remarkable that, even in these adverse conditions, the ML method achieves accuracy errors of about 40%, which are in line with the accuracy of the UR methods on models with two classes instead of five classes. Thus, it seems possible to conclude that the ML method allows to significantly increase, compared to established UR methods, the number of cases where service demand errors are in an acceptable range.

5.3 Sensitivity Analysis with More Samples

Finally, we explore the effects of increasing the simulation length to 3600 seconds (1 hour). It is interesting to assess the quality of the estimates in cases where traces can be studied without time constraints on the estimation. The results shown in Figure 9 indicate that the outcome of the analysis is similar to the 600 case, thus we give a brief summary. Overall, we observe a noticeably lower incidence of error in the 3600-sample case compared to its 60-sample counterpart. For instance, in the two class case in Figure 9(b), RR1r and RR30 achieve an acceptable average error of 5% approximately for the low utilization and 18.5% approximately error average for the highest load. Moreover, the RR1r method performs overall better than any of the UR methods, further confirming that it is a useful addition to linear regression based estimation approaches. On the other hand, UR1 performs worse overall, whereas ML is again the best method achieving overall only an average error of 5% approximately. It is striking to see that the different methods gain accuracy at different rates compared to the results for 600 and 60 samples. Specifically, it seems possible to conclude that UR1 is the slowest method in improving its quality by an increase of the number of available samples. Conversely, the aggregation based techniques, and specifically RR30r, show a dramatic accuracy improvement of over 50% compared to the estimates for 600 samples.

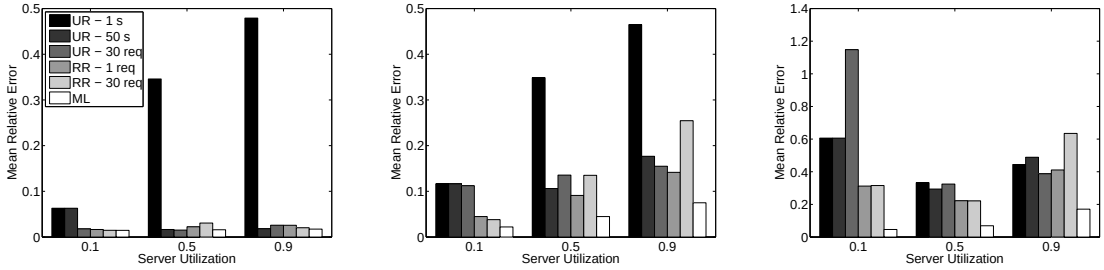
6. CONCLUSIONS

We have proposed linear regression and maximum likelihood methods for estimating request service demands needed for performance model parameterization. The proposed methods differ from established approaches in that they estimate demands from response time measurements only, which are often easier to collect than utilization samples. Extensive simulations and experiments on a real-world ERP application demonstrate the very good accuracy of our approach for different levels of utilization, number of available samples, and number of workload classes in the target model. In particular, the maximum likelihood approach is the most accurate, showing robust results in all experiments. Future work will investigate the accuracy impact of non-exponential features in arrivals and service demands such as high-variability and burstiness. We are also interested in assessing if the proposed approach generalizes to queueing network models



(a) Single class (b) Two classes (c) Five classes

Figure 8: Mean relative error at different server utilization levels for the 60-sample case. The legend shown in figure (a) is identical for figures (b) and (c).



(a) Single class (b) Two classes (c) Five classes

Figure 9: Mean relative error at different server utilization levels for the 3600-sample scenario. The legend shown in figure (a) is identical for figures (b) and (c).

where one needs to jointly estimate service demands at multiple stations from measurement data. In this case, response time information should be probably collected at each individual server in order to be able to compete with the estimation accuracy of utilization based methods.

7. ACKNOWLEDGEMENT

This research has been partially funded by the InvestNI/SAP MORE project.

8. REFERENCES

[1] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using magpie for request extraction and workload modelling. *Proc. of OSDI*, pp. 259–272, 2004.

[2] M. Bertoli, G. Casale, and G. Serazzi. Java modelling tools: an open source suite for queueing network modelling and workload analysis. *Proc. of QEST*, pp. 119–120. IEEE, 2006.

[3] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. 2nd ed., John Wiley and Sons, 2006.

[4] P. Buchholz and A. Panchenko. A two-step em algorithm for MAP fitting. *Proc. of ISCIS*, pp. 217–227. Springer, 2004.

[5] G. Casale, P. Cremonesi, and R. Turrin. Robust workload estimation in queueing network performance models. *Proc. of Euromicro PDP*, pp. 183–187, 2008.

[6] L. Cherkasova and R. Gardner. Measuring CPU overhead for I/O processing in the xen virtual machine monitor. pp. 387–390. Proc. of USENIX Annual Technical Conf., 2005.

[7] A. Heindl. *Traffic-Based Decomposition of General Queueing Networks with Correlated Input Processes*. Ph.D. Thesis, Shaker Verlag, Aachen, 2001.

[8] Z. Liu, L. Wynter, C. H. Xia, F. Zhang. Parameter inference of queueing models for it systems using end-to-end measurements. *PEVA*, 63(1):36–60, 2006.

[9] D. Menasce and V. A. F. Almeida. *Capacity Planning for Web Performance*. Prentice Hall, 1998.

[10] M. F. Neuts. *Structured Stochastic Matrices of M/G/1 Type and Their Applications*. Marcel Dekker, 1989.

[11] G. Pacifici, W. Segmuller, M. Spreitzer, and A. Tantawi. CPU demand for web serving: Measurement analysis and dynamic estimation. *PEVA*, 65(6-7):531–553, 2008.

[12] J. Rolia and V. Vetland. Parameter estimation for performance models of distributed application systems. *Proc. of CASCON*, p. 54. IBM Press, 1995.

[13] J. Rolia and V. Vetland. Correlating resource demand information with arm data for application services. *Proc. of WOSP*, pp. 219–230. ACM, 1998.

[14] Enterprise resource planning - software from SAP, Apr 2009. http://www.sap.com/solutions/business-suite/erp/index.epx

[15] A. B. Sharma, R. Bhagwan, M. Choudhury, L. Golubchik, R. Govindan, and G. M. Voelker. Automatic request categorization in internet services. *SIGMETRICS PER*, 36(2):16–25, 2008.

[16] Q. Zhang, L. Cherkasova, and E. Smirni. A regression-based analytic model for dynamic resource provisioning of multi-tier applications. *Proc. of ICAC*, pp. 27–27. IEEE, 2007.

[17] T. Zheng, J. Yang, M. Woodside, M. Litoiu, and G. Iszlai. Tracking time-varying parameters in software systems with extended kalman filters. *Proc. of CASCON*, pp. 334–345. IBM Press, 2005.