Evaluating Approaches to Resource Demand Estimation

Simon Spinner[a,*], Giuliano Casale[b], Fabian Brosig[a], Samuel Kounev[a]

[a] *University of Würzburg, Am Hubland, Würzburg, Germany*
[b] *Imperial College London, Department of Computing, SW7 2AZ, UK*

Abstract

Resource demands are a key parameter of stochastic performance models that needs to be determined when performing a quantitative performance analysis of a system. However, the direct measurement of resource demands is not feasible in most realistic systems. Therefore, statistical approaches that estimate resource demands based on coarse-grained monitoring data (e.g., CPU utilization, and response times) have been proposed in the literature. These approaches have different assumptions and characteristics that need to be considered when estimating resource demands. This paper surveys the state-of-the-art in resource demand estimation and proposes a classification scheme for estimation approaches. Furthermore, it contains an experimental evaluation comparing the impact of different factors (monitoring window size, number of workload classes, load level, collinearity, and model mismatch) on the estimation accuracy of seven different approaches. The classification scheme and the experimental comparison helps performance engineers to select an approach to resource demand estimation that fulfills the requirements of a given analysis scenario.

Keywords: Resource demand estimation, workload characterization, quantitative performance analysis, performance modeling

1. Introduction

Performance models can be used to answer performance-related questions for a software system during system design, capacity planning and sizing, or system operation. There are different performance modeling formalisms, e.g. stochastic performance models (Queueing Networks (QN) [1], Queueing Petri Nets (QPN) [2]), or architecture-level performance models (e.g., Palladio Component Model (PCM) [3]. The performance models can be analyzed using ana-

*Corresponding author
Email addresses: simon.spinner@uni-wuerzburg.de (Simon Spinner), g.casale@imperial.ac.uk (Giuliano Casale), fabian.brosig@uni-wuerzburg.de (Fabian Brosig), samuel.kounev@uni-wuerzburg.de (Samuel Kounev)

lytic methods or simulation to predict the performance of a system. However, the creation of model instances for a given system can be a complex and time-consuming task. During model creation, various model parameters need to be quantified. This usually requires experimentation with the system under study to obtain the measurement data required for model parameterization. It is of paramount importance to find representative parameter values in order to ensure accurate and reliable performance predictions.

A key parameter of stochastic performance models are *resource demands* (a.k.a. service demands). A resource demand is the average time a unit of work (e.g., request or transaction) spends obtaining service from a resource (e.g., CPU or hard disk) in a system over all visits excluding any waiting times [4, 5]. The resource demand for processing a request is influenced by different factors, for example, the application logic specifies the sequence of instructions to process a request, and the hardware platform determines how fast individual instructions are executed. The definition of a resource demand implies that the value of a resource demand is platform-specific (i.e., only valid for a specific combination of application, operating system, hardware platform, etc.).

In order to quantify resource demands, a dynamic analysis of the system of interest is required. Resource demands are difficult to measure directly with state-of-the-art monitoring tools. Modern operating systems can only provide resource usage statistics on a per-process level. However, the mapping between operating system processes and application requests is non-trivial. Many applications serve different requests with one or more operating system processes (e.g., HTTP web servers). Standard profiling tools for performance debugging [6, 7] can be used to obtain execution times of individual application functions when processing an individual request. However, the resulting execution times are not broken down to the processing times at individual resources and profiling tools typically introduce high overheads significantly influencing the performance of a system. Furthermore, advanced instrumentation techniques have been proposed in the literature to measure resource demands on the operating system layer [8], or the application layer [9, 10, 11]. These techniques build upon specific capabilities of the underlying platform and are not generally applicable.

This survey focuses on statistical approaches to resource demand estimation. The advantage of resource demand estimation compared to direct measurement techniques is their general applicability and low overheads. These estimation approaches rely on coarse-grained measurements from the system (e.g., CPU utilization, and end-to-end response times), which can be easily and cheaply monitored with state-of-the-art tools without the need for fine-grained code instrumentation. These measurements are routinely collected for many applications (e.g., in data centers). Therefore, approaches to resource demand estimation are also applicable on systems serving production workloads. Over the years, a number of approaches to resource demand estimation have been proposed using different statistical estimation techniques (e.g., linear regression, Kalman filter, etc.) and based on different laws from queueing theory. When selecting an appropriate approach to resource demand estimation, one has to consider different

characteristics of the estimation approach, such as the expected input parameters, its accuracy and its robustness to measurement anomalies. Depending on the constraints of the application context, only a subset of the estimation approaches may be applicable.

The target audience of this paper are performance engineers who want to apply resource demand estimation techniques to build a performance model of a system as well as researchers working on improved estimation approaches. This paper makes the following contributions: i) a survey of the state-of-the-art in resource demand estimation, ii) a classification scheme for approaches to resource demand estimation, and iii) an experimental comparison of a subset of the estimation approaches.

The remainder of the paper is organized as follows. Section 2 summarizes the state-of-the-art and introduces the different approaches to resource demand estimation. Section 3 describes the classification scheme including a categorization of existing estimation approaches. Section 4 presents the experimental comparison of the estimation approaches and discusses the results. Section 5 concludes the paper.

2. Approaches to Resource Demand Estimation

In this section, we survey the state-of-the-art in resource demand estimation and introduce the different approaches that have been proposed in the literature.

2.1. Methodology

In order to obtain the estimation approaches listed in Table 2, we started the literature search by reading the titles and abstract of articles in the proceedings of 12 established conferences and workshops in the performance engineering community in the last 10 years. Relevant articles were analyzed further regarding references to other articles on resource demand estimation. Based on the found articles found we compiled a list of keywords to use for a broader search in common scientific search engines (scholar.google.com, portal.acm.org and citeseerx.ist.psu.edu) The keywords used for search were *resource demand (estimation)*, including synonyms *service demand*, *service time*, *service requirement*. Furthermore, we also considered the more general terms *workload characterization*, *parameter estimation* and *model calibration*. The list of articles resulting from this search was then filtered based on the titles and abstracts. After filtering, we got the list of 37 papers on resource demand estimation shown in Table 2.

2.2. Notation and Assumptions

In the following, we use a consistent notation for the description of the different approaches to resource demand estimation. We denote resources with the index $i = 1 \dots I$ and workload classes with the index $c = 1 \dots C$. The variables used in the description are listed in Table 1. We assume the *Flow Equilibrium Assumption* [12] to hold, i.e., that over a sufficiently long period of

time the number of completions is approximately equal to the number of arrivals. As a result, the arrival rate λ_c is assumed to be equal to the throughput X_c. Furthermore, we use the term resource demand as a synonym for service demand and for simplicity of exposition we assume $V_{i,c} = 1$, i.e., no distinction is made between service demand and service time.

$D_{i,c}$	average resource demand of requests of workload class c at resource i
$U_{i,c}$	average utilization of resource i due to requests of workload class c
U_i	average total utilization of resource i
$\lambda_{i,c}$	average arrival rate of workload class c at resource i
$X_{i,c}$	average throughput of workload class c at resource i
$R_{i,c}$	average residence time of workload class c at resource i
R_c	average end-to-end response time of workload class c
$A_{i,c}$	average queue length of requests of workload class c seen on arrival at resource i, excluding the arriving job
$V_{i,c}$	average number of visits of a request of workload class c at resource i
I	total number of resources
C	total number of workload classes

Table 1: Explanation of variables.

2.3. Description of Approaches

In this section, we describe the different approaches to resource demand estimation that exist in the literature. Table 2 gives an overview of all approaches.

2.3.1. Approximation with Response Times

The response time of a request at a queue is the sum of the queueing delay and the resource demands. If we assume that there are no queueing delays and the response times do not include time spent at other resources, the response time is equal to the resource demand. Thus, if queueing delays are significantly smaller than the resource demand, the resource demands can be approximated with measured response times [14, 13, 15].

2.3.2. Service Demand Law

The utilization at resource i due to requests of workload class c can be derived using the Utilization Law [5]. Solving for the resource demand leads to the Service Demand Law [5]:

$$D_{i,c} = \frac{U_{i,c}}{X_{i,c}}. \tag{1}$$

We can use this relationship to determine resource demands based on measured utilization and throughput data. In cases where a mix of requests of different workload classes arrive at the system of interest, the measured total utilization U_i needs to be apportioned appropriately among the different workload classes. This can be done by ratios obtained from additional per-class metrics provided by the operating system [4, 5] or from workload class response times [15].

Table 2: Overview of estimation approaches categorized according to statistical techniques.

Technique	**Variant**	**References**
Approximation with response times		Urgaonkar et al. [13] Nou et al. [14] Brosig et al. [15]
Service Demand Law		Lazowksa [4] Brosig et al. [15]
Linear regression	Least squares	Bard and Shatzoff [16] Rolia et al. [17, 18] Pacifici et al. [19] Kraft et al. [20, 21]
	Least absolute differences	Zhang et al. [22, 23, 24]
	Least trimmed squares	Casale et al. [25, 26]
Kalman filter		Zheng et al. [27, 28] Kumar et al. [29] Wang et al. [30, 31]
Optimization	Non-linear constrained optimization	Zhang et al. [32] Menascé [33]
	Quadratic programming	Liu et al. [34, 35, 36] Kumar et al. [37]
Machine learning	Clusterwise linear regression	Cremonesi et al. [38]
	Independent component analysis	Sharma et al. [39]
	Support vector machine	Kalbasi et al. [40]
	Pattern matching	Cremonesi et al. [41, 42]
Maximum likelihood estimation		Kraft et al. [20] Perez et al. [21]
Gibbs sampling		Sutton and Jordan [43] Wang et al. [44]
Demand Estimation with Confidence (DEC)		Kalbasi et al. [45, 46]

2.3.3. Linear Regression

A common way to infer resource demands is based on linear regression [16, 17, 20, 19, 25, 23, 22, 24]. Given a workload consisting of multiple workload classes, the linear model is usually defined based on the Utilization Law:

$$U_i^{(j)} = \sum_{c=1}^{C} \lambda_{i,c}^{(j)} D_{i,c} + U_{i,0}^{(j)}, \tag{2}$$

where index (j) denotes measurement samples obtained in time window j. For the regression to be meaningful, we need to obtain at least M simultaneous measurement samples, where M is the number of resource demands to estimate.

Commonly, non-negative Least Squares (LSQ) regression is used to solve the model [16, 17, 18, 19, 20]. However, the following issues can arise: i) resource demands are random variables with a certain distribution, thus focusing only on

the *mean* resource demands $D_{i,c}$ may lead to significant estimation errors [17], and ii) close correlations between the control variables (*multicollinearity*) may cause non-unique and unstable solutions [19]. Ad-hoc techniques to reduce the influence of multicollinearity are presented in [19]. Further techniques increasing the robustness of the regression to cope with multicollinearity, outliers and discontinuities due to software or hardware upgrades include Least Absolute Differences (LAD) regression [22, 23, 24] or Least Trimmed Squares (LTS) regression [25, 26].

An approach based on measurements of response times and queue length on arrival is proposed in [20]. The authors assume a closed Queueing Network (QN), where the system is represented by a queue with exponential service times and FCFS scheduling. For a single workload class, the mean response time of requests can then be described by $R_i = D_i(1 + A_i)$. A_i is the queue length seen by a newly arriving job, not including the job currently in service. Generalized to multiple workload classes, they define a linear model expecting response times and the average queue length on arrival as input. The model is solved with LSQ regression. In [21], the authors extend this approach to PS scheduling.

2.3.4. Kalman Filter

A Kalman filter estimates the hidden state of a dynamic system [47]. The authors in [28, 29, 31] apply it to resource demand estimation. The following filter description is based on [29]. The system state vector is defined as:

$$\mathbf{x} = \begin{pmatrix} D_{i,1} & \cdots & D_{i,C} \end{pmatrix}^T. \tag{3}$$

Without any a-priori knowledge about the system state dynamics, the system state model that describes how the system state evolves over time is reduced to

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{w}_k, \tag{4}$$

where index k denotes discrete time steps. A process noise term $\mathbf{w}_k$ is assumed to be normally distributed with zero mean.

The vector $\mathbf{z}_k$ contains the measurements obtained from the system at time step k. The relationship between system state $\mathbf{x}_k$ and measurements $\mathbf{z}_k$ is denoted as measurement model. For a M/M/1 queue, the measurement equation can be described by:

$$\mathbf{z} = h(\mathbf{x}) = \begin{pmatrix} R_{i,1} \\ \cdots \\ R_{i,C} \\ U_i \end{pmatrix} = \begin{pmatrix} \frac{D_{i,1}}{1-U_i} \\ \cdots \\ \frac{D_{i,C}}{1-U_i} \\ \Sigma_{c=1}^{C} \lambda_{i,c} D_{i,c} \end{pmatrix}. \tag{5}$$

The measurement equation is of non-linear nature. To derive a linear measurement model for the measurements $\mathbf{z}_k$, the extended Kalman filter design [47, 29] can be used:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \text{where } \mathbf{H} = \frac{\partial h}{\partial \mathbf{x}}, \tag{6}$$

where $\mathbf{v}_k$ is the observation noise, which is assumed to be white Gaussian noise with zero mean. In [27, 28], the authors give recommendations on how to choose filter configurations such as initial state vectors or covariance matrices of process and observation noise. In [30, 31], Wang et al. propose an alternative Kalman filter based on the Utilization Law, which they use to estimate resource demands for multi-tenant applications.

2.3.5. Optimization

In this section, we describe estimation approaches that are defined as optimization problems and solved with mathematical programming methods. In contrast to the linear regression approaches in Section 2.3.3, the estimation approaches described here are based on more general objective functions.

In [34, 35, 36, 37], the objective function aims at reducing the prediction error of response times and utilizations:

$$\min \quad \sum_{c=1}^{C} p_c (R_c - \tilde{R}_c)^2 + \sum_{i=1}^{I} (U_i - \tilde{U}_i)^2, \tag{7}$$

where $\tilde{R}_c$ denotes the measured response time of workload class c and $\tilde{U}_i$ the measured utilization of resource i. Expressions of R_c respectively U_i are derived from standard queueing formulas. The factor p_c weights the response time errors with the proportion of the number of requests of workload class c, $p_c = \frac{\lambda_c}{\sum_{d=1}^{C} \lambda_d}$. The resulting optimization problems can be solved using quadratic programming techniques. The authors in [37] extends this optimization approach to estimate load-dependent resource demands. Their approach requires a-priori knowledge of the type of function, e.g., polynomial, exponential or logarithmic, that best describes the relation between workloads and resource demands.

The work in [33] formulates an alternative optimization problem that depends only on response time and arrival rate measurements:

$$\begin{aligned} &\min \sum_{c=1}^{C} (R_c - \tilde{R}_c)^2 \text{ with } R_c = \sum_{i=1}^{I} \frac{D_{i,c}}{1 - \sum_{d=1}^{C} \lambda_{i,d} D_{i,d}} \\ &\text{subject to } D_{i,c} \geq 0 \quad \forall i, c \text{ and } \sum_{c=1}^{C} \lambda_{i,c} D_{i,c} < 1 \quad \forall i. \end{aligned} \tag{8}$$

The resulting optimization requires a non-linear constrained optimization solver.

2.3.6. Machine Learning

In [38], the authors use cluster-wise regression techniques to improve the robustness to discontinuities in the resource demands due to system configuration changes. The observations are clustered into groups where the resource demands can be assumed constant, and the demands are then estimated for each cluster separately. In [41, 42], the authors propose a novel algorithm based on a combination of change-point regression methods and pattern matching to address the same challenge.

Independent Component Analysis (ICA) is a method to solve the *blind source separation* problem, i.e., to estimate the individual signals from a number of aggregate measurements. [39] describes a way to use ICA for resource demand estimation, using a linear model based on the Utilization Law. ICA can provide estimates solely based on utilization measurements, when the following constraints hold [39]: i) the number of workload classes is limited by the number of observed resources; ii) the arrival rate measurements are statistically independent; iii) the inter-arrival times have a non-Gaussian distribution while the measurement noise is assumed zero-mean Gaussian. ICA not only provides estimates of resource demands, but also automatically categorizes requests into workload classes.

In [40], Kalbasi et al. consider the use of Support Vector Machines (SVM) [48] for estimating resource demands. They compare it with results from LSQ and LAD regression and show that it can provide better resource demand estimates depending on the characteristics of the workload.

2.3.7. Maximum Likelihood Estimation (MLE)

MLE allows the inference of the statistics of a random variable by determining the probability of observing a certain sample path. For resource demand estimation, the authors of [20, 21] use MLE with measured response times and queue lengths seen upon arrival of requests. The authors obtain N response time measurements $R_i^1, \ldots, R_i^N$ of *individual* requests and then search for the resource demands $D_{i,1}, \ldots, D_{i,C}$ so that the probability of observing the measured response times is maximized. The maximization problem is defined as:

$$\max \mathbb{L}(D_{i,1}, \ldots, D_{i,C}) = \sum_{k=1}^{N} \log \mathbb{P}[R_i^k \mid D_{i,1}, \ldots, D_{i,C}]. \tag{9}$$

The actual representation of the likelihood function is obtained using phase-type distributions. With the likelihood function we can determine the global maximum of the likelihood function and thus get values for the resource demands $D_{i,1}, \ldots, D_{i,C}$ that explain the measured response times best.

2.3.8. Gibbs sampling

Bayesian inference methods based on Markov-Chain Monte Carlo techniques are used in [43, 44] to estimate the resource demands of a queueing network. Both authors propose to use Gibbs sampling techniques [49] to construct a Markov Chain that simulates the density $f(D_{i,c})$. Through Gibbs sampling techniques one can infer the $D_{i,c}$ based on the observed posterior distribution of $f(D_{i,c})$. Sutton and Jordan [43] provide an estimator that is applicable to single-class, open queueing networks. Wang et al. [44] extend the estimator to multi-class, closed queueing networks.

2.3.9. Other Approaches

In [45, 46], Rolia et al. propose a technique for estimating the aggregate resource demand of a given workload mix, called Demand Estimation with Confidence (DEC). This technique assumes that a set of benchmarks is available for

a system under study. Each benchmark utilizes a subset of the different functions of an application. DEC expects the measured demands of the individual benchmarks as input and then derives the aggregate resource demand of a given workload mix as a linear combination of the demands of the individual benchmarks. DEC is able to provide confidence intervals of the aggregate resource demand [45, 46].

3. Classification Scheme

In this section, we describe our classification scheme for categorizing the approaches described in Section 2. The goal of the classification scheme is to help performance engineers to select an estimation approach that best fits their specific requirements. We distinguish between three dimensions: input parameters, output metrics and robustness to anomalies in the input data. For each dimension, we first describe its features and then categorize the estimation approaches accordingly.

3.1. Input Parameters

Approaches to resource demand estimation often differ in terms of the set of input data they require. We do no consider parameters of the underlying statistical techniques (e.g., parameters controlling the optimization algorithm) because these are specific to the concrete implementation of an estimation approach.

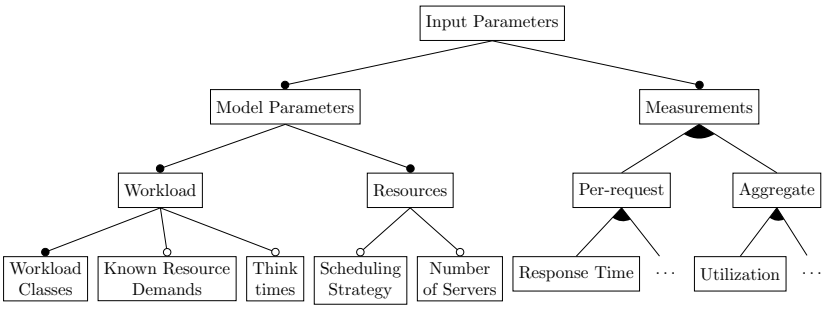

Figure 1: Types of input parameters.

Figure 1 depicts the main types of input parameters for demand estimation algorithms. The parameters are categorized into *model parameters* and *measurements*. In general, parameters of both types are required. Model parameters capture information about the performance model for which we estimate resource demands. Measurements consist of samples of relevant performance metrics obtained from a running system, either a live production system or a test system.

Before estimating resource demands, it is necessary to decide on certain modeling assumptions. As a first step, resources and workload classes need to

be identified. This is typically done as part of the workload characterization activity when modeling a system. It is important to note, that the observability of performance metrics may influence the selection of resources and workload classes for a system under study. In order to be able to distinguish between individual resources or workload classes, observations of certain per-resource or per-class performance metrics are necessary. At a minimum, information about the number of workload classes and the resources for which the demands should be determined is required as input to the estimation. Depending on the estimation approach, more detailed information on resources and workload classes may be expected as an input (e.g., *scheduling strategies*, *number of servers*, or *think times*).

Measurements can be further grouped into *per-request* or *aggregate*. Common per-request measurements used in the literature include response times, arrival rates, visit counts, and queue length seen upon arrival. Aggregate measurements can be further distinguished in *class-aggregate* and *time-aggregate* measurements. Class-aggregate measurements are collected as totals over all workload classes processed at a resource. For instance, utilization is usually reported as an aggregate value because the operating system is agnostic of the application internal logic and is not aware of different request types in the application. Time-aggregate measurements, e.g., average response times or average throughput, are aggregated over a sampling period. The sampling period can be evenly or unevenly spaced.

Categorization of Existing Approaches

We considered the approaches to resource demand estimation listed in Table 2 and examined their input parameters. Table 3 contains an overview of the input parameters of each estimation approach. Parameters common to all estimation approaches, such as the number of workload classes and the number of resources, are not included in this table. The required input parameters vary widely between different estimation approaches. Depending on the system under study and the available performance metrics, one can choose a suitable estimation approach from Table 3. Furthermore, approaches based on optimization can be adapted by incorporating additional constraints into the mathematical model capturing the knowledge about the system under study. For example, the optimization approach by Menascé [33] allows one to specify additional known resource demand values as input parameters. These a-priori resource demands may be obtained from the results of other estimation approaches or from direct measurements.

Another approach that requires resource demand data is described by Lazowska [4, Chapter 12]. Lazowska assumes that the resource demands are approximated based on measurements provided by an accounting monitor. Such an accounting monitor, however, does not include the system overhead caused by each workload class. The system overhead is defined as the work done by the operating system for processing a request. Lazowska [4, Chapter 12] describes a way to distribute unattributed computing time among the different workload classes providing more realistic estimates of the actual resource demands.

Table 3: Input parameters of estimation approaches (utilization U_i, response time R_c, throughput X_c, arrival rate λ_c, queue length $A_{i,c}$, visit counts $V_{i,c}$, demands $D_{i,c}$, think time Z, scheduling policy P).

Estimation approach	**Measurements**					**Parameters**		
	U_i	R_c	X_c/λ_c	$A_{i,c}$	$V_{i,c}$	$D_{i,c}$	Z	P
Approximation with response times								
Urgaonkar et al. [13]		✗[1]			✗			
Nou et al. [14]	✗	✗						
Brosig et al. [15]		✗						
Service Demand Law								
Lazowska [4]			✗			✗[2]		
Brosig et al. [15]	✗	✗			✗			
Linear regression								
Bard and Shatzoff [16], Rolia et al. [17, 18], Pacifici et al. [19]	✗		✗					
Zhang et al. [22, 23, 24]	✗		✗					
Kraft et al. [20, 21]		✗		✗				✗
Casale et al. [25, 26]	✗		✗					
Kalman filter								
Zheng et al. [27, 28]	✗	✗	✗					
Kumar et al. [29]	✗	✗	✗					
Wang et al. [30, 31]	✗		✗					
Optimization								
Zhang et al. [32]	✗	✗	✗			(✗)[5]		✗
Liu et al. [34, 35, 36]	✗	✗	✗		✗			✗
Menascé [33]		✗	✗			✗[3]		
Kumar et al. [37]	✗	✗	✗					✗
Machine learning								
Cremonesi et al. [38]	✗		✗					
Sharma et al. [39]	✗							
Kalbasi et al. [40]	✗		✗					
Cremonesi et al. [41, 42]	✗		✗					
Maximum likelihood estimation								
Kraft et al. [20]		✗[4]		✗[4]			✗	✗
Perez et al. [21]		✗[4]		✗[4]			✗	✗
Gibbs sampling								
Sutton and Jordan [43]		✗[4]	✗[4]					✗
Wang et al. [44]				✗[4]			✗	
Kalbasi et al. [45, 46] (DEC)			✗		✗			

[1] Response time per resource.
[2] Measured with accounting monitor. System overhead is not included.
[3] A selected set of resource demands is known a priori.
[4] Non-aggregated measurements of individual requests.
[5] Requires coefficient of variation of resource demands in case of FCFS scheduling.

Approaches based on response time measurements, such as those proposed by Zhang et al. [32], Liu et al. [34, 35, 36] and Kumar et al. [37], require information about the scheduling strategies of the involved resources abstracted as queueing stations. This information is used to construct the correct problem definition

for the optimization technique. The estimation approaches proposed by Kraft et al. [20], Perez et al. [21], and Wang et al. [44] assume a closed queueing network. Therefore, they also require the average think time and the number of users as input.

In addition to requiring a set of specific input parameters, some approaches also provide a rule of thumb regarding the number of required measurement samples. Approaches based on linear regression [17, 20, 19] need at least $K+1$ linear independent equations to estimate K resource demands. When using robust regression methods, significantly more measurements might be necessary [25]. In [37], Kumar et al. provide a formula to calculate the number of measurements required by their optimization-based approach. The formula only provides a minimum bound on the number of measurements and more measurements are normally required to obtain good estimates [24].

3.2. Output Metrics

Approaches to resource demand estimation are typically used to determine the mean resource demand of requests of a given workload class at a given resource. However, in many situations the estimated mean value may not be sufficient. Often, more information about the confidence of estimates and the distribution of the resource demands is required. The set of output metrics an estimation approach provides can influence the decision to adopt a specific method.

Generally, resource demands cannot be assumed to be deterministic [45]; for example, they might depend on the data processed by an application or on the current state of the system [17]. Therefore, resource demands are described as random variables. Estimates of the mean resource demand should be provided by every estimation approach. If the distribution of the resource demands is not known beforehand, estimates of higher moments of the resource demands may be useful to determine the shape of their distribution.

We distinguish between point and interval estimators of the real resource demands. Confidence intervals would be generally preferable, however, it is often a challenge to ensure that the statistical assumptions underlying a confidence interval calculation hold for a system under study (e.g., distribution of the regression errors).

In certain scenarios, e.g., if Dynamic Voltage and Frequency Scaling (DVFS) or hyperthreading techniques are used [37], the resource demands are load-dependent. In such cases, the resource demands are not constant, but a function that may depend, e.g., on the arrival rates of the workload classes [37].

Categorization of Existing Approaches

Table 4 provides an overview of the output metrics of the considered estimation approaches. Point estimates of the mean resource demand are provided by all approaches. Confidence intervals can be determined for linear regression using standard statistical techniques, as mentioned by the authors in [17, 20]. These techniques are based on the central limit theorem assuming an error

Table 4: Output metrics of estimation approaches.

Estimation approach	**Resource demands**			
	Point estimates	Confidence interval	Higher moments	Load-dependent
Response time approximation				
Urgaonkar et al. [13]	✗			
Nou et al. [14]	✗			
Brosig et al. [15]	✗			
Service Demand Law				
Lazowska [4]	✗			
Brosig et al. [15]	✗			
Linear regression				
Rolia et al. [17, 18], Pacifici et al. [19]	✗	✗[2]		
Zhang et al. [23]	✗	✗[2]		
Kraft et al. [20, 21]	✗	✗[2]		
Casale et al. [25, 26]	✗	✗[2]		
Kalman filter				
Zheng et al. [27, 28]	✗			
Kumar et al. [29]	✗			
Wang et al. [30, 31]	✗			
Optimization				
Zhang et al. [32]	✗		✗[1]	
Liu et al. [34, 35, 36]	✗			
Menascé [33]	✗			
Kumar et al. [37]	✗			✗
Machine learning				
Cremonesi et al. [38]	✗			
Sharma et al. [39]	✗			
Kalbasi et al. [40]	✗			
Cremonesi et al. [41, 42]	✗			
Maximum likelihood estimation				
Kraft et al. [20]	✗		✗	
Perez et al. [21]	✗		✗	
Gibbs sampling				
Sutton and Jordan [43]	✗			
Wang et al. [44]	✗			
Kalbasi et al. [45, 46] (DEC)	✗	✗		

[1] Only feasible if a-priori knowledge of the resource demand variance is available.
[2] The accuracy of the confidence intervals is not evaluated.

term with a normal distribution. Resource demands are typically not deterministic violating the assumptions underlying linear regression. The influence of the distribution of the resource demands on the accuracy of the confidence intervals is not evaluated for any of the approaches based based on linear regression. DEC [45, 46] is the only approach for which the confidence intervals have been evaluated in the literature [45, 46]. The Maximum Likelihood Estimation (MLE) [20] approach and the optimization approach described by Zhang

et al. [32] are capable of providing estimates of higher moments. This additional information comes at the cost of a higher amount of required measurements.

All of the estimation approaches in Table 2 can estimate load-independent mean resource demands. Additionally, the Enhanced Inferencing approach [37] also supports the estimation of load-dependent resource demands, assuming a given type of function.

3.3. Robustness

It is usually not possible to control every aspect of a system while collecting measurements. This can lead to anomalous behavior in the measurements [25]. The authors in [25, 26] and [19] identified the following issues with real measurement data:

- presence of outliers,
- background noise,
- non-stationary resource demands,
- collinear workload,
- and insignificant flows.

Background activities can have two effects on measurements: the presence of outliers and background noise [25]. Background noise is created by secondary activities that utilize a resource only lightly over a long period of time. Outliers result from secondary activities that stress a resource at high utilization levels for a short period of time. Outliers can have a significant impact on the parameter estimation resulting in biased estimates [25]. Different strategies are possible to cope with outliers. It is possible to use special filtering techniques in an upstream processing step or to use parameter estimation techniques that are inherently robust to outliers. However, tails in measurement data from real systems might belong to bursts, e.g., resulting from rare, but computationally complex requests. The trade-off decision as to when an observation is to be considered as an outlier has to made on a case-by-case basis taking into account the characteristics of the specific scenario and application.

The resource demands of a system may be non-stationary over time (i.e., not only the arrival process changes over time, but also the resource demands, which for example can be described by a $M_t/M_t/1$ queue). Different types of changes are observed in production systems. Discontinuous changes in the resource demands can be caused by software and hardware reconfigurations, e.g., the installation of an operating system update [25]. Continuous changes in the resource demands may happen over different time scales. Short-term variations can often be observed in cloud computing environments where different workloads experience mutual influences due to the underlying shared infrastructure. Changes in the application state (e.g., database size) or the user behavior (e.g., increased number of items in a shopping cart in an online shop during Christmas season) may result in long-term (over days, weeks, and months) trends

and seasonal patterns. When using the estimated resource demands to forecast the required resources of an application over a longer time period, these non-stationary effects need to be considered in order to obtain accurate predictions. In order to detect such trends and seasonal patterns, it is possible to apply forecasting techniques on a time series resulting from the repeated execution of one the considered estimation approach over a certain time period. An overview of such forecasting approaches based on time series analysis can be found in [50].

Another challenge for estimation approaches is the existence of collinearities in the arrival rates of different workload classes. There are two possible reasons for collinearities in the workload: low variation in the throughput of a workload class or dependencies between workload classes [19]. For example, if we model *login* and *logout* requests each with a separate workload class, the resulting classes would normally be correlated [19]. The number of logins usually approximately matches the number of logouts [19]. Collinearities in the workload may have negative effects on resource demand estimates. A way to avoid these problems is to detect and combine workload classes that are correlated [19].

Insignificant flows are caused by workload classes with very small arrival rates in relation to the arrival rates of the other classes. Pacifici et al. [19] experience numerical stability problems with their linear regression approach when insignificant flows exist. However, it is noteworthy, that there might be a dependency between insignificant flows and the length of the sampling time intervals. If the sampling time interval is too short, the variance in arrival rates might be high.

Categorization of Existing Approaches

Ordinary Least Squares (LSQ) regression are often sensitive to outliers. Stewart et al. [24] come to the conclusion that Least Absolute Differences (LAD) regression is more robust to outliers than LSQ regression. Robust regression techniques as described by Casale et al. [25, 26] try to detect outliers and ignore measurement samples that cannot be explained by the regression model. Liu et al. [36] also include an outlier detection mechanism in their estimation approach based on optimization.

In general, sliding window or data aging techniques can be applied to the input data to improve the robustness to non-stationary resource demands [19]. In order to detect software and hardware configuration discontinuities, robust and cluster-wise regression approaches are proposed in [25, 26, 38]. If such discontinuities are detected, the resource demands are estimated separately before and after the configuration change. Approaches based on Kalman filters [27, 28, 29] are designed to estimate time-varying parameters. Therefore, they automatically adapt to changes in the resource demands after a software or hardware discontinuity. None of the considered estimation approaches are able to learn long-term trends or seasonal patterns (over days, weeks, or months).

Collinearities are one of the major issues when using linear regression [51]. A common method to cope with this issue is to check the workload classes for collinear dependencies before applying linear regression. If collinearities are detected, the involved workload classes are merged into one class. This is

proposed in [19, 25]. The DEC approach in [45] mitigates collinear dependencies, since it only estimates the resource demands for mixes of workload classes.

Pacifici et al. also consider insignificant flows in [19]. They call a workload class insignificant if the ratio between the throughput of the workload class and the throughput of all workload classes is below a given threshold. They completely exclude insignificant workload classes from the regression in order to avoid numerical instabilities [19].

4. Experimental Evaluation

The goal of the experiments presented in this section is to compare the accuracy of different estimation approaches. A set of experiments was conducted to evaluate the impact of the following factors on the estimation accuracy of the considered estimation approaches: *(RQ1)* length of sampling interval, *(RQ2)* number of samples, *(RQ3)* number of workload classes, *(RQ4)* load level, *(RQ5)* collinear workload classes, *(RQ6)* missing jobs in workload model, and *(RQ7)* delays during processing. (RQ8) analyses the execution time of the considered estimation approaches. We describe the conducted experiments in detail and discuss the results. Section 4.1 describes the experiment setup used to obtain the measurement traces. Section 4.2 explains the selection and comparison of the estimation approaches. Finally, Section 4.3 discusses the experiment results.

4.1. Experiment Setup

In the experimental evaluation, we used two different sources to obtain the measurement traces for the comparison: a queueing simulator and a set of micro-benchmarks executed on a real system. The simulator and the micro-benchmarks each produce traces of observations of the performance metrics required for resource demand estimation. These traces are provided as input to the estimation approaches and the resulting resource demands are used to evaluate the estimation accuracy.

4.1.1. Dataset D1: Queueing Simulator

Dataset D1 consists of traces of arrival times and response times of individual requests from experiments with different number of workload classes $C = \{1, 2, 5\}$ and different load levels $U = \{10\%, 50\%, 90\%\}$. Each experiment was repeated 100 times resulting in a total of 900 different traces. We used a queueing simulator based on a M/M/1 queue with FCFS scheduling and an open workload that logs detailed statistics of each simulated request. Each experiment run simulated 3600 requests with exponential inter-arrival times. This corresponds to one hour of simulated time. Inter-arrival times and resource demands are both generated from exponential distributions. For each experiment run, the mean resource demand of each workload class is randomly drawn from a uniform distribution between 0 and 1 seconds, and scaled to yield the expected load level.

4.1.2. Dataset D2: Micro-Benchmarks

In order to obtain dataset D2, we performed a series of experiments running micro-benchmarks with a known CPU resource demand on a real system. The micro-benchmarks generate a closed workload with exponentially distributed think times and resource demands. As mean values for the resource demands, we selected 14 different subsets of the base set $[0.02s, 0.25s, 0.5s, 0.125s, 0.13s]$ with number of workload classes $C = \{1, 2, 3\}$. The subsets were arbitrarily chosen from the base set so that the resource demands are not linearly growing across workload classes. The subsets intentionally also contained cases where two or three workload classes had the same mean value as resource demand. The mean think times were determined according to the desired load level of an experiment. We again varied the number of workload classes $C = \{1, 2, 3\}$ the load level $U = \{20\%, 50\%, 80\%\}$ between experiments.

Each experiment run has a length of approximately one hour. Dataset D2 contains measurement traces from a total of 210 experiment runs. The mean think time was calculated according to the required load level. We also used the micro-benchmarks to generate specialized traces for the scenarios evaluating a high number of workload classes (up to 20 classes) in Section 4.3.3, collinear workload classes in Section 4.3.5, background jobs in Section 4.3.6, and delayed processing in Section 4.3.7.

The micro-benchmarks were implemented with the Ginpex experiment framework [52]. The CPU load of the micro-benchmarks consists of the calculation of Fibonacci numbers, the number of iterations is calibrated by Ginpex before an experiment run to match the desired resource demand. We used a pool of machines with similar hardware configurations for the experiments. Each machine had an Intel Core 2 Quad Q6600 4 x 2.4 GHz CPU, 8 GB RAM, and 2 x 500 GB SATA2 disks, running a Ubuntu 10.04 64-bit operating system. We deactivated CPU cores in the operating system to prevent the parallel execution of the resource demands and to simulate a single-core machine.

During each experiment run we collected observations of the arrival times and execution times of individual requests, and the average CPU utilization. The execution times were measured by Ginpex (using the `System.nanoTime()` method provided by Java). The utilization was measured with the `sar` tool from the sysstat package [53], which is part of most Linux distributions. Average statistics for the throughput and response times were derived from the measurements afterwards.

4.2. Comparison of Estimation Approaches

Table 5 lists the approaches considered in the experimental evaluation. For reasons of conciseness, we use the abbreviations listed in the table to refer to estimation approaches in the following description. All estimation approaches were considered in the experimental evaluation with exception of response time approximation, Independent Component Analysis (ICA) [39] and Maximum Likelihood Estimation (MLE) [20, 44]. Response time approximation is a rather trivial approach where the assumptions are well-known, i.e., the observed response time must be close to the considered resource demand. In most practical

scenarios this assumption does not hold, resulting in high estimation errors. ICA automatically groups the requests into workload classes besides estimating resource demands. However, the interpretation of the resulting classes is difficult and the resulting resource demands cannot be directly compared to other approaches. MLE has high computational requirements (both with respect to CPU and memory) and can take a long time to provide estimates compared to the other approaches (factor 10 to 100). The computational overhead made an application of MLE to our extensive datasets infeasible.

Given that there are no publicly available implementations of the considered estimation approaches, we developed our own implementations. Most of the approaches were implemented in MATLAB using its functions for non-negative least-squares regression (`lsqnonneg`) and constrained non-linear optimization (`fmincon`). The optimization approaches MO and LO were checked to be convex, so that a single run of fmincon is sufficient. KF is implemented in C++ using the `Covariance_scheme` class provided by the bayes++ library [54].

We used the following configuration for the experimental comparison: SDL uses the average utilization and throughput of the complete experiment length as input and apportions the aggregate utilization between workload classes using the observed average response time as described in [15]. UR uses a standard non-negative least-squares regression algorithm (see `lsqnoneg`). The parameterization of KF follows the guidelines suggested by Zheng et al. [28] (D1: state covariance Q=0.0025, observation covariance R=0.1; D2: Q=0.0001 R=0.0001). We also applied a moving average filter to the resulting demands with a window size of 10 minutes. MO uses the recursive optimization algorithm proposed by Menascé [33]. In contrast, LO executes the optimization algorithm once with the complete observation traces as input. RR comes in two different versions: one for FCFS [20] and one for PS scheduling [21]. We used the FCFS variant for dataset D1 and the PS variant for dataset D2.

Abbreviation	**Estimation Approach**
SDL	Service Demand Law (Brosig et al. [15])
UR	Utilization regression (Rolia et al. [17])
KF	Kalman filter (Kumar et al. [37])
MO	Menascé optimization [33]
LO	Liu optimization (Liu et al. [36])
RR	Response time regression (Kraft et al. [20])
GS	Gibbs Sampling (Wang et al. [31])

Table 5: Estimation approaches considered in the experimental evaluation.

To assess the accuracy of the estimation approaches, we rely on the mean relative demand error E_d as error metric. Equation 10 shows the definition of E_d. C is the number workload classes, D_c^{est} the estimated resource demand of class c and D_c^{act} the actual resource demand of class c.

$$E_d = \frac{1}{C} \sum_{c=1}^{C} \left| \frac{D_c^{est} - D_c^{act}}{D_c^{act}} \right| \tag{10}$$

In some of the experiments we also use the relative utilization error E_u and relative response time error E_r to show the effect of incorrect demand estimates on the predicted utilization and response time. Equation 11 shows the definition of the utilization error

$$E_u = \left| \frac{\sum_{c=1}^{C} \lambda_c * D_c^{est} - U}{U} \right|. \tag{11}$$

C is the number workload classes, λ_c the observed throughput of class c, D_c^{est} the estimated resource demand of class c and U is the observed utilization. Equation 12 shows the definition of the response time error

$$E_r = \frac{1}{C} \sum_{c=1}^{C} \left| \frac{R_c^{cal} - R_c^{act}}{R_c^{act}} \right|. \tag{12}$$

C is the number of workload classes, R_c^{act} is the average observed response time of class c, and R_c^{cal} is the predicted average response time of class c obtained with Mean Value Analysis (MVA) [55].

4.3. Results

4.3.1. RQ1: Length of Sampling Interval

The sampling interval defines the time period for which average statistics, e.g., of utilization or response times, are calculated. The total experiment length is divided into fixed-length sampling intervals. In this experiment, we used observation traces from datasets D1 and D2 with medium load ($U = 50\%$) and one workload class. The average statistics are calculated for different sampling intervals, varying between one second and and two minutes. A sampling interval of one second is usually the lowest resolution for operating system monitoring tools (e.g., the `sar` utility for obtaining resource usage statistics). The maximum sampling interval of two minutes is chosen so that there are at least 30 samples per experiment run.

From the considered estimation approaches, only UR, KF, MO, and LO rely on average statistics. To be concise, we leave out the results for RR and GS, which are based non-aggregated measurements of individual requests, and SDL, which always takes the average over the complete observation period. As expected, the latter estimation approaches are not influenced by the length of the sampling interval.

Figure 2 shows the relative demand errors E_d for dataset D1 under medium load ($U = 50\%$) and one workload class. All four estimation approaches are negatively influenced by small sampling intervals. Under small sampling intervals with one second, estimation accuracy of LO suffers the most and the error decreases only slowly with longer sampling intervals. However, the relative error is comparable to the other approaches in case of 60 and 120 seconds sampling intervals (below 5%).

In addition to dataset D1, Table 6 shows the results for dataset D2. This table includes an additional column containing the mean number of requests

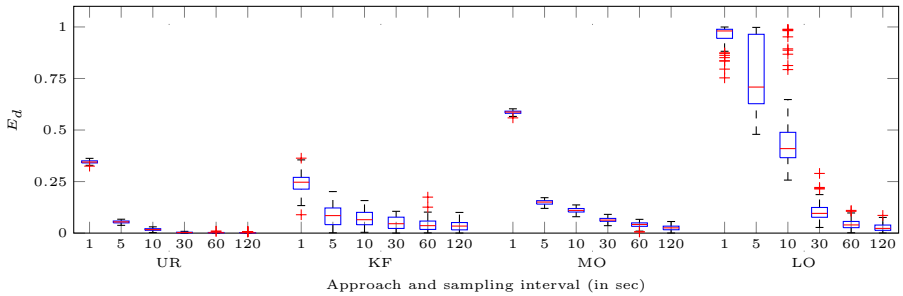

Figure 2: Boxplot of demand estimation error E_d for different sampling intervals (dataset D1, load level $U = 50\%$, number of workload classes $C = 1$).

			$mean[E_d]$ ($std[E_d]$)			
		N	UR	KF	MO	LO
D1	1s	1.00	34.54 (0.74)	24.35 (4.89)	58.67 (0.79)	95.82 (4.80)
	5s	4.99	5.43 (0.66)	8.44 (5.03)	14.91 (1.16)	77.20 (17.35)
	10s	9.99	1.74 (0.55)	7.00 (4.01)	11.03 (1.18)	46.55 (17.11)
	30s	29.97	0.31 (0.20)	4.80 (3.17)	6.37 (1.06)	10.42 (4.38)
	60s	59.95	0.23 (0.17)	4.20 (2.91)	4.04 (1.26)	4.31 (2.31)
	120s	119.90	0.19 (0.17)	3.61 (2.38)	2.57 (1.23)	2.68 (1.82)
D2	1s	11.58	8.60 (6.97)	13.41 (15.89)	15.04 (3.36)	27.31 (26.32)
	5s	57.89	0.59 (0.32)	5.66 (4.05)	9.79 (1.19)	3.42 (3.31)
	10s	115.79	0.60 (0.59)	3.51 (2.10)	8.78 (0.82)	2.01 (1.72)
	30s	347.36	0.77 (0.66)	1.41 (0.74)	8.03 (0.79)	1.41 (1.13)
	60s	694.40	0.80 (0.56)	1.73 (1.24)	7.82 (0.83)	1.38 (1.09)
	120s	1387.79	0.91 (0.81)	1.38 (1.50)	7.87 (0.79)	1.30 (1.04)

Table 6: Mean and standard deviation of demand estimation error E_d for different sampling intervals (dataset D1 and D2, load level $U = 50\%$, number of workload classes $C = 1$). N denotes the average number of requests observed in one sampling interval.

N observed during each sampling interval. The average resource demands in dataset D2 were by a magnitude smaller than in dataset D1. Therefore, more requests are observed during each sampling interval and the peaks at the one second sampling interval are smaller in D2. However, we can again observe that LO shows the highest relative error for the one second sampling interval.

The influence of the length of the sampling interval can be explained by *end-effects* due to requests which are fully attributed to one sampling period, although they start and end in different intervals. For linear regression this has been identified before by Rolia and Lin [17, 56] as one source of inaccuracy. In [23], Zhang et al. come to the conclusion that longer sampling intervals improve the accuracy of regression-based approaches. However, in practice, the maximum length of the sampling interval is usually limited because it increases the required experiment length and may hinder the ability of the estimator to adapt to changes in the resource demands. Given that a good choice for the sampling interval always depends on the length of the resource demands, one should ensure that sufficient requests are observed in each sampling interval. The results in Table 6 suggest that a sampling interval length where on average

$N > 60$ requests are observed yields acceptable estimates for all approaches.

4.3.2. RQ2: Number of Samples

In this experiment, we employed dataset D1 and reduced the number of samples used for resource demand estimation from 3600 to 600. This corresponds to an experiment length of ten minutes. Dynamic, self-adaptive systems require an estimator to keep up with frequent changes. Therefore, we argue that an estimator should also be able to converge to a stable value in shorter time frames.

	N	SDL	UR	KF	MO	LO	RR	GS
$mean[E_d]$	600	0.13	0.79	7.3	4.1	6.6	2.5	4.9
	3600	0.023	0.23	4.2	4	4.3	1.4	4.8
stat. sig. (95%)		✓	✓	✓		✓	✓	
p-value		1.2e-24	4.2e-15	4.8e-129	0.81	1.3e-05	5.3e-05	0.75

Table 7: Mean demand estimation error E_d for different number of samples N (dataset D1, load level $U = 50\%$, number of workload classes $C = 1$).

Table 7 shows the results for dataset D1. Differences in the mean relative resource demand errors from the experiment runs are tested for statistical significance using a non-paired T-test with a 95% confidence level. The estimation approaches SDL, UR, KF and LO exhibit a significant dependency on the number of available samples. With $N = 600$ they show a decreased accuracy compared to $N = 3600$. However, all approaches still yield results with acceptable accuracy (below 10%).

4.3.3. RQ3: Number of Workload Classes

A higher number of workload classes makes the estimation problem more complex since more variables need to be estimated. In RQ3, we analyze the sensitivity of the considered estimation approaches to the number of workload classes. The analysis is structured into three subquestions: RQ3.1 compares the relative demand errors of experiments with different number of workload classes, RQ3.2 explores properties of the dataset that influence the estimation accuracy in case of several classes, and RQ3.3 tests the behavior of the estimation approaches if the number of classes is scaled out.

RQ3.1: Comparison of relative demand errors. We now compare the relative demand errors from runs with three different number of workload classes. We used a subset of dataset D1 containing samples for 1, 2 and 5 classes at a load level of 50% (in total 300 repetitions) and D2 for 1, 2, and 3 also at 50% (in total 70 repetitions).

Table 8 shows the results for datasets D1 and D2. We used a single factor Analysis of Variance (ANOVA) with a confidence level of 95% to test for significant differences in E_d with different number of workload classes. The hypothesis that E_d is influenced by the number of workload classes cannot be rejected for any of the considered approaches. However, there are clear differences in the

	C	SDL	UR	KF	MO	LO	RR	GS
$mean[E_d]$	1	0.023	0.231	4.2	4.04	4.31	1.44	4.76
	2	127	27.7	88.3	83.4	98.8	8.56	93.4
	5	153	59.8	110	97.2	120	18.2	111
stat. sig. (95%)		✓	✓	✓	✓	✓	✓	✓
p-value		2.52e-04	6.53e-17	7.59e-04	1.12e-03	8.62e-04	1.34e-04	1.61e-03

(a) Dataset D1

	C	SDL	UR	KF	MO	LO	RR	GS
$mean[E_d]$	1	0.833	0.8	1.73	7.82	1.38	1.11	2.87
	2	1.02	12.8	3.84	5.23	4.33	1.85	3.56
	3	2.07	24.1	4.01	5.56	4.94	3.44	4.9
stat. sig. (95%)		✓	✓	✓	✓	✓	✓	✓
p-value		5.96e-06	1.02e-05	0.0368	5.05e-05	0.033	5.56e-05	0.0081

(b) Dataset D2

Table 8: Mean relative demand error E_d for number of workload classes $C = \{1, 2, 5\}$ (load level $U = 50\%$).

quantitative effect on E_d between both datasets. While most estimation approaches yield relatively accurate results for dataset D2 (E_d mostly below 10% except for UR), we consider the results for dataset D1 insufficient for most use cases. With 2 or 5 workload classes, the estimated resource demand largely deviates from the actual one by more than 50% in most cases on dataset D1. We analyze the reasons for these high deviations in RQ3.2.

UR shows a degraded accuracy for multiple workload classes on both datasets. A deeper analysis of the resulting estimates show that the estimates converge very slowly compared to the other estimation approaches. The linear regression is done based on measurements from approximately 60 measurement intervals, which is assumed to be sufficient for the considered number of workload classes. However, the performance of UR heavily depends on the workload [24]. We explain the poor accuracy of UR in our experiments with too few variations in the workload. Given that the utilization is kept at a fixed level during the experiments, UR can only explore a limited region of the complete space.

RQ3.2: Correlation Analysis. The comparison in RQ3.1 shows a largely degraded accuracy of most estimation approaches on D1 with multiple workload classes. Given that high variances in E_d were observed between experiment runs, we performed a correlation analysis testing the influence of different properties of a sample set on E_d.

The property $mean[Q]$ stands for the mean queue length Q observed during an experiment run. $min[X * D]$ takes the minimum of the mean throughput X and the average resource demand D over all workload classes. A low value of $min[X * D]$ is an indicator that the workload includes classes with a small contribution in relation to the other classes. These are also called *insignificant flows*. $std[D]$ is the standard variance of the service demands. If this value is high, the mean service demands of workload classes are very diverse.

Table 9 shows the results of the correlation analysis. We used the Spearman's rank correlation coefficient (denoted with ρ) in order to be able to identify non-

C			SDL	UR	KF	MO	LO	RR	GS
$mean[Q]$	1	ρ	-0.042	-0.14	0.065	-0.42	-0.5	0.27	0.65
		p-value	0.68	0.16	0.52	1.3e-05	2.1e-07	0.0062	0
	2	ρ	**0.71**	0.27	0.67	0.68	**0.73**	0.46	**0.75**
		p-value	0	0.0073	0	0	0	1.8e-06	0
	5	ρ	0.52	0.25	0.65	0.63	0.66	0.39	0.63
		p-value	5.6e-08	0.013	0	0	0	5.8e-05	0
$min[X * D]$	1	ρ	-	-	-	-	-	-	-
		p-value	-	-	-	-	-	-	-
	2	ρ	-0.46	-0.54	-0.55	-0.56	-0.44	-0.52	-0.45
		p-value	2.2e-06	1.2e-08	5.2e-09	2.6e-09	7.7e-06	6.9e-08	4.5e-06
	5	ρ	-0.45	-0.44	-0.48	-0.47	-0.49	-0.61	-0.5
		p-value	4e-06	6.9e-06	5.7e-07	1.4e-06	3.6e-07	0	1.9e-07
$std[D]$	1	ρ	-	-	-	-	-	-	-
		p-value	-	-	-	-	-	-	-
	2	ρ	**0.91**	0.35	**0.88**	**0.89**	**0.88**	0.52	**0.9**
		p-value	0	0.0004	0	0	0	5.1e-08	0
	5	ρ	**0.72**	0.37	**0.78**	**0.8**	**0.8**	0.44	**0.79**
		p-value	0	0.0002	0	0	0	4.8e-06	0

Table 9: Correlation analysis results (dataset D1, load level $U = 50\%$). Entries with $\rho > 0.7$ are in bold letters.

linear correlations. Table 9 summarizes the correlations of three properties of the sample set which were identified to influence E_d.

We identified the highest correlations ($\rho > 0.7$) for SDL, KF, MO, LO, RR, GS with $std[D]$, i.e., if the differences between the resource demand of workload classes is higher, the relative demand error E_d is also higher. In these cases, the underlying model is based on the response time equation $R = D/(1 - U)$. Assuming an open workload, this equation is only applicable to multi-class queues with FCFS scheduling if the service time of each workload class is equal [57]. This requirement does not hold for dataset D1. The higher the variation of the resource demands between workload classes is the more it lessens the estimation accuracy of the estimation approaches. The impact of this violated assumption increases if the mean queue length $mean[Q]$ in an experiment run is higher. The high correlations show that when using response times for resource demand estimation, it is important to ensure that the estimator is based on the correct scheduling strategy assumptions.

Furthermore, we could observe moderate negative correlations for $min[X*D]$ for all estimation approaches. That mean if in an experiment run, there exists a workload class with a low the total resource demand $X * D$ compared to the other workload classes, the relative demand error increases. We conclude that all considered estimation approaches are sensitive to workload classes with a low total resource demand (sometimes also called insignificant flows [19].

RQ3.3: High number of workload classes. In Section 4.3.3, the results indicate an influence of the number of workload classes on the accuracy of certain estimation approaches. In the following experiment we consider scenarios with a higher number of workload classes than before. We employed the micro-benchmarks used to obtain dataset D2 and varied the number of workload classes between 5 and 20. In total, we performed 40 experiment runs.

Table 10 shows the results from this experiment. We used a single factor

	C	SDL	UR	KF	MO	LO	RR	GS
$mean[E_d]$	5	1.24	20.5	2.89	3.51	2.44	1.78	5.17
	10	2.53	36.2	3.99	3.36	2.39	3	8.55
	15	2.86	56.9	4.32	3.44	3.11	3.52	12.5
	20	2.99	57.8	5.33	4.04	3.28	3.58	13.6
stat. sig. (95%)		✓	✓	✓		✓	✓	✓
p-value		6.59e-09	3.58e-09	5.02e-05	0.303	0.00151	6.61e-06	9.76e-10

Table 10: Mean relative demand error E_d for high numbers of workload classes $C = \{5, 10, 15, 20\}$ (dataset D2, load level $U = 50\%$).

ANOVA with a confidence level of 95% to test for significant differences in E_d with a different number of workload classes. Several estimation approaches (SDL, KF, MO, LO, RRPS) do not show a clear dependence on the number of workload classes in the considered range. In these cases, we could not observe a statistically significant difference in the estimation errors regarding the utilization and response times. The results for UR support the findings with multiple workload classes in Section 4.3.3.

4.3.4. RQ4: Load Level

We now explore the sensitivity of the estimation approaches under different system load levels using measurement traces with low, middle and high load. Dataset D1 contains data of runs with an average utilization of 10%, 50% and 90% (in total 300 repetitions), dataset D2 has runs with an average utilization of 20%, 50% and 80% (in total 30 repetitions). Only the workload intensity changed between the experiments runs, other factors were kept fixed.

	U	SDL	UR	KF	MO	LO	RR	GS
$mean[E_d]$	10%	0.0232	0.219	2.36	0.81	0.427	0.434	3.39
	50%	0.023	0.231	4.2	4.04	4.31	1.44	4.76
	90%	0.0279	0.843	33.1	5.33	90.5	1.75	24.2
stat. sig. (95%)			✓	✓	✓	✓	✓	✓
p-value		0.167	6.86e-67	1.41e-22	3.53e-95	5e-273	3.13e-16	1.93e-13

(a) Dataset D1

	U	SDL	UR	KF	MO	LO	RR	GS
$mean[E_d]$	20%	2.85	2.71	2.37	2.98	1.8	1.05	3.17
	50%	0.833	0.8	1.73	7.82	1.38	1.11	2.87
	80%	0.461	0.515	4.55	12.4	5.39	0.825	7.12
stat. sig. (95%)		✓	✓		✓	✓		✓
p-value		9.38e-08	2.55e-07	0.0606	5.49e-19	0.0146	0.505	0.000554

(b) Dataset D2

Table 11: Mean demand estimation error E_d for different load levels U and number of workload classes $C = 1$).

Table 11 shows the mean relative demand error E_d for dataset D1 and D2 with sample sets from low, middle and high load. We used a single factor Analysis of Variance (ANOVA) with a confidence level of 95% to test for statistically significant differences in E_d between the load levels.

The results for dataset D1 suggest an influence of the load level on all estimation approaches except SDL. Apart from SDL, all approaches have a higher

mean E_d at 90% utilization compared to 50% and 10%. Most conspicuous are the high relative errors (above 20%) for KF, LO and GS at high load. We explain these inaccuracies with underlying model assumptions of these estimation approaches, which are violated at high load levels. GS is based on a closed queueing model while the queueing simulator used to obtain dataset D1 executed an open workload. KF and LO use the response time equation $R = D/(1-U)$ which is highly non-linear above 90% CPU utilization. We explain the observed inaccuracies of KF and LO with deficiencies of the underlying estimation algorithms which results in a reduced estimation accuracy in highly non-linear regions. While MO is similar to LO regarding the underlying model, MO uses an iterative optimization algorithm which seems to be more stable in high load scenarios.

On dataset D2 the differences between the estimation approaches at high loads are smaller in comparison to D1. KF, MO, LO and GS are again negatively influenced by the high utilization. However, with 80% the utilization is further away from the critical region close to 100% utilization. In summary, we conclude that it may be beneficial to avoid high-load situations (above 80%) during resource demand estimation, or best use one of the SDL, UR or RR approaches.

4.3.5. RQ5: Collinear Workload Classes

In the following experiments, the influence of collinear workload classes is evaluated. For determining the level of collinearity, we use the Variance Inflation Factor (VIF) which is defined as $VIF_i = \frac{1}{1-R_i^2}$. R_i^2 is the coefficient of determination if we calculate the regression of $X_i = \sum_{j=1}^{j \leq N, j \neq i} \beta X_j$. Based on the rule of thumb proposed by Kutner et al. [58], we assume a strong collinearity between workload classes if $VIF_i > 10$ for the observed throughput.

The traces in datasets D1 and D2 both do not contain clearly collinear workload classes. The maximum VIF observed are 1.1772 and 3.1602. Therefore, we adapted the workload used for generating D2 so that one job of one workload class is followed by a job from another workload class with a certain probability p_c (including a certain think time between the two workload classes). The experiment is executed with $p_c = 0.33$) and $p_c = 1.0$. The observed VIF is on average 1.1624 and 26.2972, respectively. So for the case of $p_c = 1.0$ we can safely assume a strong collinearity between workload classes.

Collinearity		SDL	UR	KF	MO	LO	RR	GS
$mean[E_d]$	Low	2.68	39.7	3.86	5.63	5.39	3.26	4.81
	High	2.75	111	3.64	6.83	5.21	3.43	5.47
stat. sig. (95%)			✓		✓			
p-value		0.854	0.00234	0.675	0.00447	0.787	0.627	0.54
$mean[E_u]$	Low	0.0045	0.0457	1.94	4.72	0.735	0.704	2.12
	High	0.00123	0.0534	1.76	5.15	0.792	1.02	2.3
$mean[E_{rt}]$	Low	4.63	43.3	7.42	4	8.95	2.7	4.62
	High	5.47	120	7.33	3.99	9.9	3.22	4.66

Table 12: Sensitivity to collinearity in throughput observations.

Table 12 shows the results from experiments with low and high levels of

multicollinearity. We used a non-paired T-test with a confidence level of 95% to test for statistically significant differences in E_d. The only estimation approach that is clearly influenced by high levels of multicollinearity in the workload is the UR approach. This issue has also been discussed in [19] proposing different approaches to improve the robustness of UR in case of collinear workload classes.

4.3.6. RQ6: Missing Jobs in Workload Model

On real systems, it can be difficult to capture all tasks executed by the application, middleware system, or operating system in a workload model. Performance engineers are often not aware of background processes that cannot be directly attributed to the processing of user requests and that may happen at points in time difficult to foresee. In order to evaluate the sensitivity of estimation approaches to missing workload classes, we adapted the micro-benchmarks used to obtain dataset D2. We implemented a workload consisting of 3 workload classes representing the user requests and one class representing the background process. The user requests incurred an average CPU utilization of $U = 50\%$. The intensity of the background job was varied between $U = 5\%, 10\%, 20\%, 30\%$. We executed a total of 40 experiment runs. The estimation approaches were only provided observations from the three workload classes processing user requests as input.

	U_b	SDL	UR	KF	MO	LO	RR	GS
$mean[E_d]$	5%	9.32	34.5	9.33	2.28	16.7	4.61	4.82
	10%	18.2	40	15.8	3.03	29.3	6.34	6.57
	20%	34.4	64.5	27.6	9.15	49.4	13.5	12
	30%	49.6	88.3	35.9	15.3	61.3	20.1	17.7
stat. sig. (95%)		✓	✓	✓	✓	✓	✓	✓
p-value		8.16e-50	6.17e-08	2.23e-21	7.6e-32	1.57e-48	6.15e-33	1.1e-15
$mean[E_u]$	5%	0.00104	0.0517	5.25	9.23	1.83	5.28	8.11
	10%	0.00369	0.0685	7.75	13.1	3.24	9.1	10.2
	20%	0.00404	0.0898	12.9	18.8	7.12	14.7	16.7
	30%	0.00413	0.123	17.1	22.9	13.5	18.8	21.1
$mean[E_{rt}]$	5%	15.1	38	11	4.14	21.6	2.77	5.46
	10%	26.6	50.1	15.4	4.08	35.3	3.34	3.9
	20%	48.7	87	21.7	4.82	54	3.79	4.13
	30%	72.5	124	22.8	5.3	56.3	4.33	4.64

Table 13: Demand error E_d, utilization error E_u and response time error E_r when system executes background job with intensity U_b.

Table 13 contains the results for this experiment. We used a single factor ANOVA with a confidence level of 95% to test for significant differences in E_d when the intensity of the background job is varied. All estimation approaches are significantly influenced by the hidden workload class. However, the influence seems to be stronger on approaches based on the CPU utilization (SDL, UR, KF, LO) compared to the other methods using response times. The direct influence on the utilization measurements seem to have a stronger influence on the estimation accuracy than the indirect effects of the background job on the observed response times. Table 13 also contains the relative errors E_u and E_{rt} to show the influence on predictions when using the estimated resource demands.

4.3.7. RQ7: Delays during Processing

Experiment RQ7 simulates the situation when the processing of one request may consist of several visits to the CPU resource with a certain delay between the visits. The delay may be caused, e.g., by waiting for software resources (e.g., thread or connection pool), or for data from other hardware resources (e.g., hard disk or network). We adapted the micro-benchmarks used to generate dataset D2, by splitting up the Fibonacci calculation into two parts with equal length and inserting a delay period. We varied the delay period between 25ms, 75ms, and 125ms. In total we have 30 experiment runs.

	Delay	SDL	UR	KF	MO	LO	RR	GS
	25ms	5.82	19.5	5.27	6.19	3.56	6.52	6.28
$mean[E_d]$	75ms	14.8	19.8	18.2	14.2	21.2	14.8	15.4
	125ms	22.3	12	29.6	21.3	38.9	22.4	21.9
stat. sig. (95%)		✓		✓	✓	✓	✓	✓
p-value		8.7e-30	0.0771	1.31e-23	9.35e-29	1.06e-31	2.95e-27	5.19e-13
	25ms	0.00374	0.0283	1.35	1.55	0.252	2.24	1.44
$mean[E_u]$	75ms	0.00156	0.0227	2.07	4.12	0.669	8.14	6.12
	125ms	0.00214	0.0254	4.88	9.17	1.59	13.7	12
	25ms	1.33	21.7	3.93	3.9	3.65	2.07	4.04
$mean[E_{rt}]$	75ms	11.1	26.1	10.6	4.3	15.1	1.81	6.02
	125ms	19.1	25.5	18.2	4.66	30.3	1.99	4.28

Table 14: Demand error E_d, utilization error E_u, and response time error E_r when the jobs are interrupted by wait periods.

Table 14 shows the result for the experiment. We used a single factor ANOVA with a confidence level of 95% to test for significant differences in E_d when varying the length of the delays. The relative error E_d of all estimation approaches except UR is negatively influenced by the additional delay. UR is the only considered approach that is not relying on response time measurements. While in the case of SDL, KF, and LO E_{rt} are mainly impacted, it is E_u for MO, RR, and GS.

4.3.8. RQ8: Execution Time

We measured the execution times of the estimation approaches on dataset D1 to compare the computational effort associated with each approach. Dataset D1 consists of 900 measurement traces each containing observations of 3600 individual requests observed over a simulation time of one hour.

	C	U	SDL	UR	KF	MO	LO	RR	GS
		10%	1.1	1.0	0.3	671.6	20.9	77.1	19413.7
	1	50%	0.5	0.4	0.2	873.1	22.9	75.9	19619.7
		90%	0.5	0.4	0.2	2288.0	21.5	78.8	20266.9
		10%	0.6	0.6	0.4	1028.8	23.1	80.0	42910.0
$mean[T]$	2	50%	0.6	0.5	0.2	1221.5	30.0	80.5	42685.1
		90%	0.6	0.5	0.2	3418.2	38.8	83.7	45921.4
		10%	0.8	0.7	0.6	2073.5	41.9	89.4	251675.4
	5	50%	0.8	0.7	0.5	2213.8	42.3	92.5	138163.4
		90%	0.8	0.7	0.5	6389.0	88.0	96.9	138735.7

Table 15: Mean execution time T (in milliseconds) partitioned by number of workload classes C and load level U.

Table 15 contains the average execution times T for each estimation approach. SDL, UR, and KF have a low computational effort, the execution times for a single measurement trace is on average below 1 millisecond. LO and RR have a moderate computational effort, on average between 20 and 100 milliseconds. The higher effort of RR compared to UR can be explained with the lack of measurement traces for the queue length seen on arrival in dataset D1. RR first needs to calculate this metric based on response times and arrival times. MO and GS show a significantly higher computational effort, on average between 0.5 seconds and 4 minutes. Although based on the same optimization algorithm, MO is slower compared to LO because it executes the optimization recursively for each new sample, while LO runs the optimization once for the complete measurement trace. GS has a high execution time compared to the other approaches because it needs to approximate the normalising constant of state probabilities, which is very costly operation [44].

4.4. Results Summary

In this section, we summarize the results of our experiments. We identified the following sensitivities:

RQ1. When using estimation approaches based on time-aggregated observations (e.g., UR, KF, MO, LO), the length of the sampling interval is an important parameter that needs to be adjusted to the system under study. A good sampling interval length depends on the response times of requests and the number of requests observed in one interval. The sampling interval should be significantly larger than the response times of requests to avoid end-effects and it should be long enough to be able to calculate the aggregate value based on the observations of a significant number of requests (more than 60 requests per sampling interval provided good results in our experiments).

RQ2. Most estimation approaches (except MO and LO) were negatively influenced when reducing the experiment length to 10 minutes (i.e., 10 samples). However, they still yielded results with acceptable accuracy (relative demand error below 8%).

RQ3. All estimation approaches are sensitive to the number of workload classes. The linear regression method UR that only uses utilization and throughput observations generally yielded a degraded accuracy in our experiments with several workload classes. Observations of the response times of requests can help to improve the estimation accuracy significantly (RQ3.1) even in situations with a very high number of workload classes (RQ3.3). However, it is crucial to ensure that the modeling assumptions of the estimation approaches using response times are fulfilled as they are highly sensitive to violated assumptions, e.g. wrong scheduling strategies (RQ3.2). Furthermore, insignificant flows can impair resource demand estimation (RQ3.2). Workload classes with a small contribution to the total resource

demand of a system should therefore be excluded from resource demand estimation.

RQ4. When a system operates at a high utilization level (80% or higher), the estimation approaches KF, MO, LO and GS may yield inaccurate results.

RQ5. Collinearities in throughput observations of different workload classes impairs the estimation accuracy of UR. While it correctly estimates the total resource demand, the apportioning between workload classes is wrong. The other evaluated estimation approaches did not show a sensitivity to collinearities in throughput observations.

RQ6. Estimation approaches relying on response time observations (e.g., MO, RR and GS) are more robust to missing workload classes than approaches using utilization observations.

RQ7. Delays due to non-captured software or hardware resources has a strong influence on the estimation accuracy of estimation approaches based on observed response times. While some estimation approaches (e.g., [32, 36, 33]) consider the scenarios where multiple resources contribute to the observed end-to-end response time, only the authors of [21] have considered contention due to software resources in their estimation approach.

RQ8. There are significant differences in the computational complexity of the different estimation approaches. On our datasets, the estimation took between under 1 millisecond and up to 20 seconds depending on the estimation approach. When using resource demand estimation techniques on a production system (e.g., for online performance and resource management), the computational effort needs to be taken into account (especially in data centers with a large number of systems).

In summary, our evaluation shows that using response times can improve the accuracy of the estimated resource demands significantly compared to the traditional approach based on the Utilization Law using linear regression, especially in cases with multiple workload classes (see Section 4.3.3). However, approaches employing response time measurements are very sensitive if assumptions of the underlying mathematical model are violated (e.g., wrong scheduling strategy in Section 4.3.3, or delayed processing in Section 4.3.7).

5. Conclusion

We have surveyed the state-of-the-art in research of resource demand estimation. The estimation approaches are categorized according to their required input parameters, their provided output metrics, and their measures to improve their robustness to anomalies in the measurement data. Furthermore, we have evaluated the influence of different factors (sampling interval, number of samples, number of workload classes, load level, collinear workload classes,

background jobs, and delayed processing) on the estimation accuracy of different estimation approaches.

The results show, that using response times can improve the accuracy of the estimated resource demands significantly compared to a linear regression based on the Utilization Law, especially in cases with multiple workload classes. However, approaches employing response time measurements are very sensitive if assumptions of the underlying mathematical model are violated (e.g., wrong scheduling strategy, or delays due to other resources). In order to fully leverage the benefits of using response time measurements in resource demand estimation, it is therefore necessary to find appropriate abstractions that sufficiently represent the relationship between observed response times and the hidden resource demands of a system of interest. When using resource demand estimation techniques at system run-time, the computational overhead of solving such models need to be taken into account as well. We see the following future research directions to better reflect system properties during resource demand estimation:

Multiple resources. The observed end-to-end response times includes significant processing time at different resources (i.e., software as well as hardware resources). While in a distributed system it is often possible to obtain response time statistics for each tier, the residence times at different resources on the same physical machine are usually infeasible to monitor. Approaches based on response times, such as [29, 20], estimate demands only for a single bottleneck resource. The approaches in [36, 32, 44] are applicable to cases with several processing resources, however, software resources cannot be represented. Initial work considering thread pool sizes in the estimation can be found in [21].

Layered architectures. Today's systems typically consists of different layers. Each layer has its own resources contributing the overall end-to-end response time. For example in virtualized environments, the scheduling of physical resources at the hypervisor is more complex, especially in over-committed scenarios. The observed response times include additional scheduling delays if several VMs contend for the same physical resources. None of the considered approaches using response times can cope with such additional delays (see Section 4.3.7). One possible way is to adapt existing estimation approaches to explicitly exploit knowledge of the layered architecture (e.g., using layered queueing models as a basis for resource demand estimation). Another way is to develop methods to filter out noise from underlying layers prior to resource demand estimation.

Parallel processing. Given that modern CPUs typically have multiple cores, an individual request may be processed in parallel by different threads. While the queueing models underlying most estimation approaches can be usually extended to multi-server queues, it still assumes that a request is processed by only one thread at a time. The parallel processing of individual requests is still an open research question.

Load-dependent resource demands. Load-dependent resources demands are only considered in [37]. Given that modern CPUs typically come with a dynamic frequency scaling scheme to reduce power consumption, the load-dependent performance behavior of these CPUs need to be reflected in the resource demand estimation techniques.

Furthermore, we see the need for a systematic methodology to resource demand estimation. Especially, if using observed response times for resource demand estimation, a performance engineer needs first decide on certain modeling assumptions (e.g., service time distributions, scheduling strategies). That means he must already define certain parts of his performance model before resource demands and the resource demands are no longer only input parameters to the performance model, that can be estimated independently. Thus the unparameterized performance model constitutes a input to the resource demand estimation and depending on it and also the available measurements, a mathematical model for the estimation needs to be derived. A methodology would help performance engineers by providing guidelines.

Acknowledgment

The work of Giuliano Casale has been supported by the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement no. 318484 (MODAClouds). The work of Fabian Brosig and Samuel Kounev has been partly funded by the German Research Foundation (DFG) under grant no. KO 34456-1.

References

[1] G. Bolch, S. Greiner, H. de Meer, K. S. Trivedi, Queueing networks and Markov chains: modeling and performance evaluation with computer science applications, John Wiley & Sons, 2006.

[2] F. Bause, Queueing petri nets-a formalism for the combined qualitative and quantitative analysis of systems, in: Petri Nets and Performance Models, 1993. Proceedings., 5th International Workshop on, 1993, pp. 14–23.

[3] S. Becker, H. Koziolek, R. Reussner, The palladio component model for model-driven performance prediction, Journal of Systems and Software 82 (1) (2009) 3 – 22, special Issue: Software Performance - Modeling and Analysis.

[4] E. D. Lazowska, J. Zahorjan, G. S. Graham, K. C. Sevcik, Quantitative system performance: computer system analysis using queueing network models, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1984.

[5] D. A. Menascé, L. W. Dowdy, V. A. F. Almeida, Performance by Design: Computer Capacity Planning By Example, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[6] S. L. Graham, P. B. Kessler, M. K. Mckusick, Gprof: A call graph execution profiler, SIGPLAN Not. 17 (6) (1982) 120–126.

[7] R. J. Hall, Call path profiling, in: Proceedings of the 14th International Conference on Software Engineering, ICSE '92, ACM, New York, NY, USA, 1992, pp. 296–306.

[8] P. Barham, A. Donnelly, R. Isaacs, R. Mortier, Using magpie for request extraction and workload modelling, in: Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6, OSDI'04, USENIX Association, Berkeley, CA, USA, 2004, pp. 18–18.

[9] M. Kuperberg, M. Krogmann, R. Reussner, ByCounter: Portable Runtime Counting of Bytecode Instructions and Method Invocations, in: Proceedings of the 3rd International Workshop on Bytecode Semantics, Verification, Analysis and Transformation , Budapest, Hungary, 5th April 2008 (ETAPS 2008, 11th European Joint Conferences on Theory and Practice of Software), 2008.

[10] M. Kuperberg, M. Krogmann, R. Reussner, TimerMeter: Quantifying Accuracy of Software Times for System Analysis, in: Proceedings of the 6th International Conference on Quantitative Evaluation of SysTems (QEST) 2009, 2009.

[11] A. Brunnert, C. Vögele, H. Krcmar, Automatic performance model generation for java enterprise edition (ee) applications, in: EPEW, 2013, pp. 74–88.

[12] D. A. Menascé, H. Gomaa, A Method for Design and Performance Modeling of Client/Server Systems, IEEE Trans. Softw. Eng. 26 (11) (2000) 1066–1085.

[13] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, A. Tantawi, Analytic modeling of multitier Internet applications, ACM Trans. Web 1.

[14] R. Nou, S. Kounev, F. Julií, J. Torres, Autonomic QoS control in enterprise Grid environments using online simulation, J. Syst. Softw. 82 (2009) 486–502.

[15] F. Brosig, S. Kounev, K. Krogmann, Automated Extraction of Palladio Component Models from Running Enterprise Java Applications, in: VALUETOOLS '09: Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools, 2009, pp. 1–10.

[16] Y. Bard, M. Shatzoff, Statistical Methods in Computer Performance Analysis, Current Trends in Programming Methodology III.

[17] J. Rolia, V. Vetland, Parameter estimation for performance models of distributed application systems, in: CASCON '95: Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research, IBM Press, 1995, p. 54.

[18] J. Rolia, V. Vetland, Correlating resource demand information with ARM data for application services, in: Proceedings of the 1st international workshop on Software and performance, ACM, 1998, pp. 219–230.

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

[20] S. Kraft, S. Pacheco-Sanchez, G. Casale, S. Dawson, Estimating service resource consumption from response time measurements, in: VALUETOOLS '09: Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools, 2009, pp. 1–10.

[21] J. F. Perez, S. Pacheco-Sanchez, G. Casale, An offline demand estimation method for multi-threaded applications, in: Proceedings of the 2012 IEEE 20th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013.

[22] T. Kelly, A. Zhang, Predicting performance in distributed enterprise applications, Tech. rep., HP Labs Tech Report (2006).

[23] Q. Zhang, L. Cherkasova, E. Smirni, A Regression-Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications, in: Proceedings of the Fourth International Conference on Autonomic Computing, 2007, p. 27ff.

[24] C. Stewart, T. Kelly, A. Zhang, Exploiting nonstationarity for performance prediction, SIGOPS Oper. Syst. Rev. 41 (2007) 31–44.

[25] G. Casale, P. Cremonesi, R. Turrin, How to Select Significant Workloads in Performance Models, in: CMG Conference Proceedings, 2007.

[26] G. Casale, P. Cremonesi, R. Turrin, Robust Workload Estimation in Queueing Network Performance Models, in: 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP), 2008, pp. 183–187.

[27] T. Zheng, J. Yang, M. Woodside, M. Litoiu, G. Iszlai, Tracking time-varying parameters in software systems with extended Kalman filters, in: CASCON '05: Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative research, IBM Press, 2005, pp. 334–345.

[28] T. Zheng, C. Woodside, M. Litoiu, Performance Model Estimation and Tracking Using Optimal Filters, Software Engineering, IEEE Transactions on 34 (3) (2008) 391–406.

[29] D. Kumar, A. Tantawi, L. Zhang, Real-time performance modeling for adaptive software systems, in: VALUETOOLS '09: Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools, 2009, pp. 1–10.

[30] W. Wang, X. Huang, Y. Song, W. Zhang, J. Wei, H. Zhong, T. Huang, A statistical approach for estimating cpu consumption in shared java middleware server, in: Computer Software and Applications Conference (COMPSAC), 2011 IEEE 35th Annual, IEEE, 2011, pp. 541–546.

[31] W. Wang, X. Huang, X. Qin, W. Zhang, J. Wei, H. Zhong, Application-Level CPU Consumption Estimation: Towards Performance Isolation of Multi-tenancy Web Applications, in: Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, 2012, pp. 439 –446.

[32] L. Zhang, C. H. Xia, M. S. Squillante, W. N. M. Iii, Workload Service Requirements Analysis: A Queueing Network Optimization Approach, in: Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2002, p. 23ff.

[33] D. Menascé, Computing missing service demand parameters for performance models, in: CMG Conference Proceedings, 2008, pp. 241–248.

[34] Z. Liu, C. H. Xia, P. Momcilovic, L. Zhang, AMBIENCE: Automatic Model Building using IferENCE, Tech. rep., IBM Research (2003).

[35] L. Wynter, C. H. Xia, F. Zhang, Parameter inference of queueing models for IT systems using end-to-end measurements, in: Proceedings of the joint international conference on Measurement and modeling of computer systems, 2004, pp. 408–409.

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

[37] D. Kumar, L. Zhang, A. Tantawi, Enhanced inferencing: estimation of a workload dependent performance model, in: VALUETOOLS '09: Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools, 2009, pp. 1–10.

[38] P. Cremonesi, K. Dhyani, A. Sansottera, Service Time Estimation with a Refinement Enhanced Hybrid Clustering Algorithm, in: Analytical and Stochastic Modeling Techniques and Applications, Vol. 6148 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2010, pp. 291–305.

[39] A. B. Sharma, R. Bhagwan, M. Choudhury, L. Golubchik, R. Govindan, G. M. Voelker, Automatic request categorization in internet services, SIGMETRICS Perform. Eval. Rev. 36 (2008) 16–25.

[40] A. Kalbasi, D. Krishnamurthy, J. Rolia, M. Richter, MODE: Mix Driven On-line Resource Demand Estimation, in: Proceedings of the 7th International Conference on Network and Services Management, 2011, pp. 1–9.

[41] P. Cremonesi, A. Sansottera, Indirect estimation of service demands in the presence of structural changes, in: Quantitative Evaluation of Systems (QEST), 2012 Ninth International Conference on, 2012, pp. 249–259.

[42] P. Cremonesi, A. Sansottera, Indirect estimation of service demands in the presence of structural changes, Performance Evaluation 73 (0) (2014) 18 – 40, special Issue on the 9th International Conference on Quantitative Evaluation of Systems.

[43] C. Sutton, M. I. Jordan, Bayesian inference for queueing networks and modeling of internet services, The Annals of Applied Statistics 5 (1) (2011) 254–282.

[44] W. Wang, G. Casale, Bayesian service demand estimation using gibbs sampling, in: Proceedings of the 2012 IEEE 20th International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013.

[45] J. Rolia, A. Kalbasi, D. Krishnamurthy, S. Dawson, Resource demand modeling for multi-tier services, in: WOSP/SIPEW '10: Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering, ACM, 2010, pp. 207–216.

[46] A. Kalbasi, D. Krishnamurthy, J. Rolia, S. Dawson, DEC: Service demand estimation with confidence, IEEE Transactions on Software Engineering 38 (3) (2012) 561–578.

[47] D. Simon, Optimal state estimation : Kalman, H. [infinity] and nonlinear approaches, Wiley-Interscience, Hoboken, NJ, 2006.

[48] A. J. Smola, B. Schölkopf, A tutorial on support vector regression, Statistics and Computing 14 (3) (2004) 199–222.

[49] S. Geman, D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, Pattern Analysis and Machine Intelligence, IEEE Transactions on PAMI-6 (6) (1984) 721–741.

[50] G. Box, G. Jenkins, G. Reinsel, Time Series Analysis : Forecasting and Control, 4th Edition, Wiley, 2008.

[51] S. Chatterjee, B. Price, Praxis der Regressionsanalyse, Oldenbourg, 1995.

[52] M. Hauck, M. Kuperberg, N. Huber, R. Reussner, Deriving performance-relevant infrastructure properties through model-based experiments with ginpex, Software & Systems Modeling (2013) 1–21.

[53] Sysstat utilities, last accessed: 07-07-2014 11:11.
URL `http://sebastien.godard.pagesperso-orange.fr/`

[54] Bayes++, last accessed: 07-07-2014 13:08.
URL `http://bayesclasses.sourceforge.net/Bayes++.html`

[55] G. Bolch, S. Greiner, H. de Meer, K. S. Trivedi, Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications, Wiley-Interscience, New York, 1998.

[56] J. Rolia, B. Lin, Consistency issues in distributed application performance metrics, in: Proceedings of the 1994 conference of the Centre for Advanced Studies on Collaborative research, CASCON '94, IBM Press, 1994, pp. 62–.

[57] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action, Cambridge University Press, 2013.

[58] M. Kutner, C. Nachtsheim, J. Neter, Applied Linear Regression Models, The McGraw-Hill/Irwin Series Operations and Decision Sciences, McGraw-Hill Higher Education, 2003.