Service Distribution Estimation for Microservices Using Markovian Arrival Processes

Runan Wang[0000−0001−9245−6096], Giuliano Casale[0000−0003−4548−7951], and Antonio Filieri[0000−0001−9646−646X]

Department of Computing, Imperial College London, UK
{runan.wang19, g.casale, a.filieri}@imperial.ac.uk

Abstract. Building performance models for microservices applications in DevOps is costly and error-prone. Accurate service demand distribution estimation is critical to performance model parameterization. However, traditional service demand estimation methods focus on capturing the mean service demand, disregarding higher-order moments of the distribution. To address this limitation, we propose to estimate higher moments of the service demand distribution for a microservice from monitoring traces. We first generate a closed queueing model to abstract a microservice and model the departure process at the queue node as a Markovian arrival process. This allows formulating the estimation of service demand as an optimization problem, which aims to find the optimal parameters of the first multiple moments of the service demand distribution based on the inter-departure times. We then estimate the service demand distribution with a novel maximum likelihood algorithm, and heuristics to mitigate the computational cost of the optimization process for scalability. We apply our method to real traces from a microservice-based application and demonstrate that its estimations lead to greater prediction accuracy than exponential distributions assumed in traditional service demand estimation approaches.

Keywords: Service demand distribution · Markovian arrival process · Maximum likelihood estimation · Queueing models · Performance

1 Introduction

DevOps has been widely adopted in industry, becoming an important part of today's software development methodologies [2]. Compared with traditional software development, DevOps exploits a high degree of automation throughout the whole pipeline to shorten the development life cycle and deliver high-quality applications.

While DevOps provides software engineers with advantages like frequent releases of new features and fast resolution of technical issues, how to keep a speedy pace of delivery to production and ensure the quality of the software at the same time remains an open challenge [3]. Performance models can help to describe the system with a simplified abstraction, further enabling simulation and forecasting for use by both developers and operators. Stochastic models such as queueing networks [14], layered queueing networks [13], Petri nets [20] are widely used to represent web applications. Instead, software architecture models are appropriate to describe changes in software structures and resources [21].

To build performance models in the context of DevOps, it is important to consider both architectural and analytical models. Existing methods for generating architecture-level models like UML [28] and Palladio Component Model [7] often rely on manual analysis and domain knowledge, which cannot satisfy the requirement of high-degree automation in DevOps. In addition, the description languages of architectural models in previous works are independent of deployment, which brings complexity to frequent deployment and automatic calibration of performance models responding to new alternatives during this process. Compared to the above models, TOSCA [4] provides a directed topological description of applications that allows deployment and lifecycle management via dedicated orchestrators. As such, TOSCA is increasingly widespread to describe microservice-based applications.

To enable simulation and prediction with TOSCA, model-to-model transformations are required to decompose an architectural model into an analytical model that can be solved with analytical solvers via simulation. Among the parameters of a performance model generated in this way, service demand is a critical aspect that should be specified [8, 33]. Service demand generally refers to the cumulative time a request spends receiving service from system resources, such as CPU or disk, accumulated over all visits. The accuracy of service demand specification is decisive for the predictive effectiveness of performance models. Therefore, it is critical to specify accurate service demand distribution in TOSCA models.

Service demand can be estimated with measurements of CPU utilization and response time collected via system monitoring. Several different approaches for service demand estimation have been proposed over the years, such as utilization law [10], response time approximations based on linear regression [26], non-linear optimization [34] and also machine learning methods [12]. However, most of the existing approaches for service demand estimation mainly focus on estimating the *mean* service demand. Restricting attention to the mean can limit the accuracy of service demand estimation. This issue as the higher-order moments can affect the accuracy of critical metrics such as higher percentiles of the response time.

In this paper, we propose to estimate the service demand distribution. To learn the service demand, we first represent a microservice as a closed queueing system, with the finite population representing the maximum parallelism level within the microservice, and in which the service demand for queue nodes is characterized with a general acyclic phase-type (APH) distribution. After generating the continuous-time Markov chain (CTMC) for this model, we filter the departure transitions into a Markovian arrival process (MAP) to characterize the departure process at the queue node. The problem of service demand estimation can then be formulated as an optimization problem to infer the service demand distribution that maximizes the likelihood of the collected trace data with the departure process MAP. The optimal parameters of the service demand distribution can be obtained from matching moments of the APH distribution by a global search with maximum likelihood estimation.

To address the high cost of global optimization, we then propose a heuristic estimation method. In this method, the problem of service demand distribution estimation is divided into sub-problems of fitting different moments, using a collection of estimation methods. The required given data for fitting parameters for estimation with MAP consists of the inter-departure times, response times and the time of departure instant,

which can be directly collected with network traffic sniffing from pairs of arrival and departure events. To evaluate our method, we apply it to the analysis of real traces from deploying and monitoring a microservice-based application. The results show that our method can fit the distribution of real traces with a high degree of accuracy.

The rest of the paper is organized as follows. In Section 2, we recall necessary background and definitions. In Section 3 the inter-departure time model and problem formulation are introduced. In Section 4, we discuss our proposed service demand distribution estimation method based on global optimization with maximum likelihood. In Section 5, we introduce the heuristic method for service demand distribution estimation. We present our experimental results in Section 6. Related work is summarized in Section 7. Finally, we draw conclusions in Section 8.

2 Preliminaries

Acyclic Phase-type Distribution. A phase-type (PH) distribution [23] can be defined as the distribution of absorbing time in a continuous-time Markov chain (CTMC) with finite states $\{1, 2, ..., m, m+1\}$, where the first m states are transient and the last state is absorbing. The infinitesimal generator matrix of the CTMC $\boldsymbol{G}$ is

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{T} & \boldsymbol{t} \\ \boldsymbol{0^T} & 0 \end{bmatrix}$$

The sub-generator $\boldsymbol{T}$ with dimension $m \times m$ specifies the transition rate from state i to state j. We also define $\boldsymbol{t} = -\boldsymbol{T}\boldsymbol{e}$, where $\boldsymbol{e}$ denotes a column vector of 1 with appropriate dimension. We can further describe the stationary distribution of the transient states with $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$, subject to $\boldsymbol{\alpha e} = 1, \alpha_i \geq 0$. A PH distribution may thus be compactly specified as $PH(\boldsymbol{\alpha}, \boldsymbol{T})$.

An acyclic PH (APH) distribution [5] is a subset of PH distributions with acyclic underlying Markov chain. This implies that any state in the underlying Markov chain cannot be visited more than once before absorption. If a random variable Y has APH distribution with parameter $\boldsymbol{\alpha}'$ and $\boldsymbol{T}'$, we write $Y \sim APH(\boldsymbol{\alpha}', \boldsymbol{T}')$.

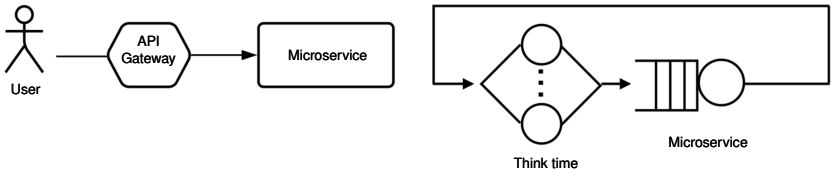
Service demand distribution modeling. In this paper, the service demand distribution is modeled as an APH distribution. The parameters of $APH(\boldsymbol{\alpha}, \boldsymbol{T})$ can be obtained with various PH distribution fitting methods [15]. In this work, we will use the method of moment matching, which can fit the parameters to match an arbitrary number of moments of a reference on empirical distribution. In particular, we consider using the first three moments to study the APH distribution for service demand. The third moment (skewness, S_k) is considered for its characterization on the fitting performance of the end of the tail.

$$S_k = \frac{E[(X-\eta)^3]}{(E[(X-\eta)^2])^{3/2}} \tag{1}$$

In equation (1), η denotes the mean value. The first three moments can be described with the mean value (η), the squared coefficient of variance (SCV, c^2) and the skewness (S_k).

$$E[X^2] = (1+c^2)\eta^2 \tag{2}$$

$$E[X^3] = S_k(c^2)^{3/2}\eta^3 + 3\eta^3 c^2 + \eta^3 \tag{3}$$



(a) (b)

Fig. 1: The structure (a) and the queueing model (b) for of the example applicaton

Thus, we can write the service demand distribution as function of the first three moments $E[X]$, $E[X^2]$ and $E[X^3]$ with parameters η, c^2, and S_k.

Markovian Arrival Processes. MAPs [22] are able to incorporate correlations between successive inter-arrival times. An n-state MAP consists of two stochastic processes, referring to a counting process and a phase process modeled by a finite state (n states) CTMC with infinitesimal generator $\boldsymbol{Q}$. Let $\boldsymbol{D}_0$ be a matrix associated with transitions without arrivals with non-negative off-diagonal elements; $\boldsymbol{D}_0$ and $\boldsymbol{D}_1$ satisfy $\boldsymbol{Q} = \boldsymbol{D}_0 + \boldsymbol{D}_1$ and $(\boldsymbol{D}_0 + \boldsymbol{D}_1)\boldsymbol{e} = \boldsymbol{0}$.

3 Problem formulation

We propose to observe the departure process of a microservice and determine parameters of service demand distribution, modeled as an APH, using maximum likelihood estimation.

Microservice-based applications can be abstracted as a queueing model. Here we take a simple microservice as an illustrating example[1]. This is a simple microservice exposing a body mass index (BMI) calculation service. The calculation service is a minimalistic microservice that only receives requests and posts responses without external processing. We generate a closed workload to simulate the microservice clients in the system – the structure of the example is given in Figure 1(a). Figure 1(b) illustrates the analytical model for the application, consisting of a closed queueing network describing the microservice buffer and server, as well as the think time of clients. The model features N concurrent users, each modeled as a job. Scheduling could be either first-come first-served (FCFS) or Processor-sharing (PS) order depending on the implementation details of the web server handling the requests within the microservice. We assume exponentially distributed user think times at the delay station. The problem is to determine the APH service demand distribution in the queueing station. Note that since we focus on a single class of jobs, the model admits a product-form solution for the steady-state distribution, while no specific product-form simplifications are available to analyze the departure process of this queue. As such, the service distribution identification problem does not satisfy a simple analytical closed-form to conduct inference.

Departure process modeling with MAP. Referring to [1], the inter-event times in queueing models can be captured with a quasi birth-and-death process (QBD). We

[1] https://github.com/go-chassis/go-bmi

can generate the infinitesimal generator $\boldsymbol{Q}$ of the underlying CTMC and then filter the events associated with job departures from $\boldsymbol{Q}$ as $\boldsymbol{D}_1$. That is, all departure transitions from the queue are tagged in $\boldsymbol{D}_1$. Then, a MAP can be used to model the departure process with representation $\boldsymbol{D}_0$ and $\boldsymbol{D}_1$, where $\boldsymbol{D}_0 = \boldsymbol{Q} - \boldsymbol{D}_1$.

We consider a MAP $= \{\boldsymbol{D}_0, \boldsymbol{D}_1\}$ that represents the departure process of the queueing station, our objective is to estimate the parameters for service demand distribution as $APH(\boldsymbol{\alpha}, \boldsymbol{T})$ from the observable inter-departure times (IDTs). We denote the time between two successive departure events i and $i-1$ as $X_i = a_i - a_{i-1}$. Thus, the IDTs of jobs are $\boldsymbol{X} = [X_1, X_2, \ldots, X_{n-1}]$. Since the departure process is modeled as a MAP, the IDTs follows a PH distribution $PH(\boldsymbol{\pi}, \boldsymbol{D}_0)$, where $\boldsymbol{\pi} = \boldsymbol{\pi}(-\boldsymbol{D}_0)^{-1}\boldsymbol{D}_1$ indicating the stationary distribution of the embedded chain. If $\boldsymbol{D}_0$ is acyclic, then the PH distribution specializes into an APH one. This distribution produces an interval stationary initialization for the MAP.

For the MAP described above, the joint probability density function (PDF) of IDTs $\boldsymbol{X} = [X_1, \ldots, X_n]$ is

$$f(\boldsymbol{X}) = \boldsymbol{\pi} e^{\boldsymbol{D}_0 X_1} \boldsymbol{D}_1 e^{\boldsymbol{D}_0 X_2} \boldsymbol{D}_1 \ldots e^{\boldsymbol{D}_0 X_n} \boldsymbol{D}_1 \boldsymbol{e} \tag{4}$$

For computational convenience, we assume that the given departure events are independent. The logPDF of the IDTs can be approximated as

$$\log f(\boldsymbol{X}) = \sum_{i=1}^{n} \log(\boldsymbol{\pi} e^{\boldsymbol{D}_0 X_i} \boldsymbol{D}_1 \boldsymbol{e}) \tag{5}$$

In general, let $\boldsymbol{\theta}$ be the parameter set of the service demand distribution to be estimated. The log-likelihood for the IDTs is

$$\log f(\boldsymbol{\theta}|\boldsymbol{X}) = \sum_{i=1}^{n} \log(\boldsymbol{\pi} e^{\boldsymbol{D}_0(\boldsymbol{\theta}) X_i} \boldsymbol{D}_1(\boldsymbol{\theta}) \boldsymbol{e}) \tag{6}$$

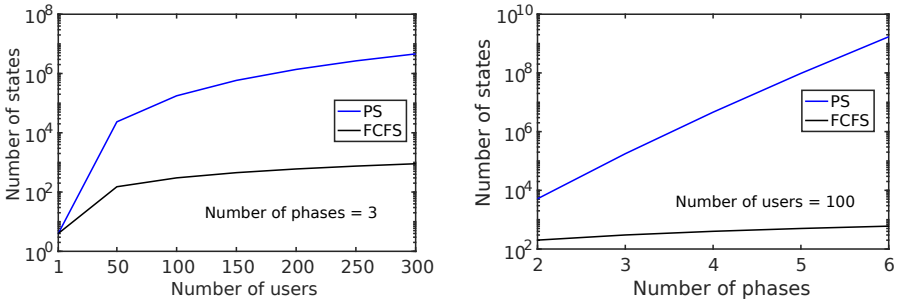
In (6), $\boldsymbol{D}_0(\boldsymbol{\theta})$ and $\boldsymbol{D}_1(\boldsymbol{\theta})$ describe the functional dependencies between $\boldsymbol{D}_0$, $\boldsymbol{D}_1$ and the service demand distribution parameters $\boldsymbol{\theta}$ such as its moments, e.g., η, c^2, and S_k mentioned in Section 2. Then our problem of service demand distribution estimation can be formulated as finding the parameters that maximize the log-likelihood of the IDTs measured from the monitoring traces.

$$f_{obj}(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta} \geq 0} \quad \log f(\boldsymbol{X}|\boldsymbol{\theta}) \tag{7}$$

4 Global optimization based estimation

The parameter estimation of service demand distribution is based on observations of real system trace. In this paper, we consider a finite observation with n samples. Our measured observation consists of the IDTs, the timestamps of each departing instant and the response times.

Data preprocessing. For a real system, there could be a large number of requests from the users arriving within a very short period. If we directly take all of the samples in the trace, it could be quite time-consuming to calculate the likelihood function in (6), due to the cost of evaluating the matrix exponential.



(a) (b)

Fig. 2: The number of states in the CTMC state space with PS and FCFS

To address the above issue, we observe that the inter-departure times of jobs can be grouped into different patterns. In order to accelerate the execution times, we apply clustering based on k-means [17] to partition the IDTs to obtain K groups of data with cluster centroids $\boldsymbol{C} = [C_1, C_2, \ldots, C_K]$. Then, the log joint PDF in (5) can be approximated based on the IDT clusters as

$$\log f(\boldsymbol{C}) = \sum_{i=1}^{K} L_i \log(\boldsymbol{\pi} e^{\boldsymbol{D}_0 C_i} \boldsymbol{D}_1 \boldsymbol{e}) \tag{8}$$

where L_i denotes the number of points in cluster i.

CTMC state space explosion. Increasing of the number of concurrent users, the state space of the CTMC can easily suffer state-space explosion. Assume a single-server queue where the service demand SD of the queue node is APH distributed and a delay node as shown in the example in Figure 1, and there are N users in the queueing network. First, we consider the jobs in the queue are processed with a first-come-first-served (FCFS) order. Only one job can be served by the server at one time. Let P denote the number of phases in the service process, i.e., the number of columns in $\boldsymbol{\alpha}$. The number of states in the state space is

$$s = N \cdot P + 1 \tag{9}$$

Instead, if the server follows a processor sharing (PS) scheduling strategy, i.e., multiple jobs can be served simultaneously, the number of states in the state space is

$$s = \sum_{i=0}^{N} (N + 1 - i) \binom{i + P - 2}{i} \tag{10}$$

Compared to FCFS, we can see that the state space for PS grows combinatorially, as shown in Figure 2, making the analysis of the CTMC intractable.

To mitigate the complexity of dealing with PS scheduling, we propose to capture the behaviours of the original model with a simpler model. We focus on the mean queue length that can be approximated by using mean-value analysis (MVA). Instead of considering all the individual jobs circulating in the delay and queue nodes as usual, we propose a modified model with only an estimated number of jobs $N^{'}$ perpetually looping within the queue.

$$N^{'} = \frac{N-1}{N} E[U(N)] \tag{11}$$

Let $E[U(N)]$ denote that there are averaged $E[U(N)]$ users at the queue when N users in the system. Note that the number of jobs $N^{'}$ looping in the new model is decided by the expected number of users at an arrival instant in the queue based on Schweitzer's Approximation [31].

For a real system with a large number of users, this approximation can lead to a significant reduction of the computational cost, while providing adequately accurate results. For example, $N^{'} = 2$ is obtained with $N = 100$ and a 3-phase service distribution, the number of states in the state space is only 10, which is much smaller compared to the one shown in Figure 2(a) with $N = 100$ under PS schedule.

MLE for service demand distribution. Our objective is to search optimal parameters for approximating service demand distributions. Given the observed trace data, a common approach for parameter estimation is maximum likelihood estimation (MLE) [24], which casts the estimation as a global optimization problem. We propose an estimation method that combines MLE with simulations of queueing models to approximate the APH distribution for service demand.

Algorithm 1 describes the implementation of this method in details. The algorithm requires a set of clustered IDTs and the searching boundaries. In each iteration, the algorithm generates a set of moments satisfying the bound constraints and then the APH distribution is fitted from the current moments. The conditions of convergence for the algorithm are set as follows. There are two conditional parameters, including the maximum number of evaluation on the objective function ($MaxEvalFunc$) and a last step tolerance for ending iteration (Tol). For moment matching we use the BuTools package [16], pointing to *APHFit* in Line 3. Note that we need to satisfy that the APH distribution is feasible with given parameters, i.e., both $\boldsymbol{\alpha}$ and $\boldsymbol{T}$ are not empty or zero. After obtaining the service demand distribution SD, a queueing model is generated with a queue node of SD. The current queueing model can be solved by analyzing the underlying CTMC, obtaining the infinitesimal generator $\boldsymbol{Q}$. By analyzing the transitions in $\boldsymbol{Q}$, the transition rates of departure events on the queue node can be filtered for $\boldsymbol{D}_1$ as shown in Lines 6-8.

In Algorithm 1, the optimal parameters of service demand distribution are obtained with the maximum likelihood value of the monitoring traces. However, the computation of the infinitesimal generator involves the computation of a matrix exponential, which is computationally expensive and rises numerical instability. To mitigate these issues, we use CTMC uniformization [29] which is well-known to be an effective numerical method for computing transient measures involving matrix exponential. For transient analysis, uniformization techniques can be applied with sub-generator $\boldsymbol{D}_0$ and the initial distribution $\boldsymbol{\pi}$ of the MAP. Since the transient rate in $\boldsymbol{D}_0$ of a real system could be large, to guarantee stable calculations, we adopt the scaling method from [32], involving a scaling factor q to avoid floating-point errors. In Line 10, the scaling CTMC uniformization method is defined as *ctmc_uniform*, which takes π, $\boldsymbol{Q} - \boldsymbol{D}_1$ and the centroid of the cluster as the input. The approximated transient probability is obtained as $\boldsymbol{\beta}$ for $\boldsymbol{\pi} e^{\boldsymbol{D}_0}$. Then the log-likelihood value can be computed using (8) at Line 11.

Algorithm 1 Global optimization based estimation method

Input: C $\leftarrow$ Set of clustered inter-departure times $[c_1, c_2, \ldots, c_n, l_1, l_2, \ldots, l_n]$, where c_i is the centriod value and l_i is the number of points in cluster i.
$LB \leftarrow$searching lower bound
$UB \leftarrow$searching upper bound
Output: $SD \leftarrow$ Estimated service demand with APH distribution
1: Random initialize $M_0 \leftarrow [m_1, m_2, m_3], LB < M_0 < UB$
2: **while** $MaxEvalFunc \leq 10^{10}$ and $Tol \geq 10^{-8}$ **do**
3: Service demand distribution $APH(\boldsymbol{\alpha}, \boldsymbol{T}) \leftarrow APHFit[m_1, m_2, m_3]$
4: **if** $APH(\boldsymbol{\alpha}, \boldsymbol{T})$ is feasible **then**
5: Generate a queueing network model QNM with service demand $APH(\boldsymbol{\alpha}, \boldsymbol{T})$
6: $\boldsymbol{Q} \leftarrow$ solve(QNM)
7: Filter $\boldsymbol{D}_1$ from the infinitesimal generator $\boldsymbol{Q}$
8: MAP $\leftarrow \{\boldsymbol{Q} - \boldsymbol{D}_1, \boldsymbol{D}_1\}$, generate $\boldsymbol{\pi}$
9: **for** $i = 1$ to n **do**
10: $\boldsymbol{\beta} \leftarrow ctmc_uniform(\boldsymbol{\pi}, \boldsymbol{Q} - \boldsymbol{D}_1, c_i)$
11: $L \leftarrow L + \log(\boldsymbol{\beta}\boldsymbol{D}_1\boldsymbol{e})l_i$
12: **end for**
13: **end if**
14: **end while**
15: Get optimal parameter set $[m_1, m_2, m_3]$ with maximum likelihood value
16: **return** $SD \leftarrow APHFit[m_1, m_2, m_3]$

5 Heuristics-based estimation

As illustrated in Algorithm 1, the global optimization method for service demand distribution estimation needs to consider a large search space. It could be time-consuming to obtain the optimal parameter set maximizing the likelihood value. The method presented in this section estimates the parameters sequentially, rather than jointly, offering a heuristic estimation that trades accuracy for speed.

Mean service demand estimation. The mean value of service demand can be efficiently estimated based on performance measurements from monitoring traces. We refer to the work in [26], which allows estimating the expected value of service demand with queue length and response times. Both queue length and response times are easily measured with system monitoring. Since in this work we target the departure process, the input dataset of the estimation method contains the following data by calculating from system monitoring at departing occurrence.

- The timestamp of a job departing from the queue node (DT)
- The response times from the monitoring traces (R)
- The queue length seen upon arrivals (A)

Considering a single class of jobs in the system, let N be the size of the population in the closed queueing network. Therefore, the mean service demand $E[D]$ for the single-class case can be estimated as [26]:

$$E[D] = \frac{E[R]}{1 + E[A]} \tag{12}$$

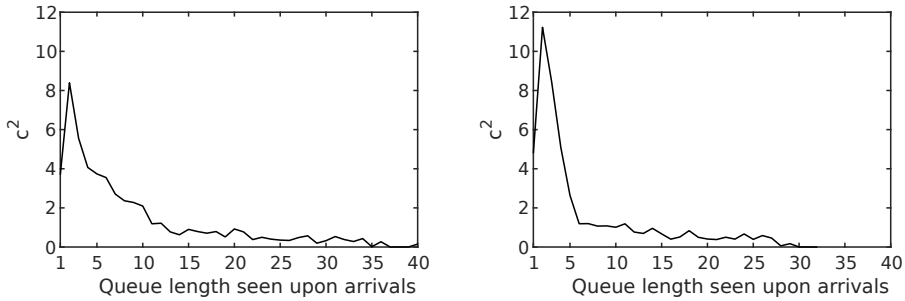


Fig. 3: c^2 conditional on the queue length seen upon arrivals for simulation (a) and the real trace (b)

where $E[R]$ and $E[A]$ is the expected value of response times and the queue length seen upon arrivals, respectively.

SCV estimation. To estimate the second moment of service demand, we investigate the state-dependent behavior of the system. The estimation formulation is derived from the SCV on the mean queue length seen upon arrival. In a queueing system, the response time of a job is related to the number of jobs in the queue node either waiting or receiving service. We propose to estimate c^2 using the following heuristic expression

$$c^2 = \max \frac{E[(R_{ij} - E[R_i])^2]}{E[R_i]^2} \tag{13}$$

where i is a value for the length of queue seen upon arrival, with $i = 0, \ldots, max(A)$. R_i denotes a set of response times of which the queue length seen upon arrival is i, and R_{ij} denotes the j^{th} response time belonging to R_i. In detail, at each departure instant, we can collect the response time of the current completed job and the number of jobs remaining in the queue. Then the queue length is sorted and the response times can be grouped according to different numbers of the queue length as R_{ij}.

To demonstrate the accuracy of the approximation for SCV, we conduct an experiment with $N = 100$. We analyze real trace data from monitoring and compare the c^2 of real traces to the simulated queueing model with estimated parameters based on MLE. We can see from Figure 3(a) that the c^2 conditional on the queue length first increases to the maximum and then decrease with the growth of queue length. The same pattern can be also observed for the real trace as shown in Figure 3(b). The simulated c^2 value is 8.3. It can be seen that the $max(c^2)$ of the simulation is close to the one for the real trace with $c^2 = 11.2$.

Heuristic service demand distribution estimation. Our heuristic method to accelerate the MLE estimation is shown in Algorithm 2. Compared to the global optimization in Algorithm 1, the heuristic-based method is used to estimate the service demand distribution with three separate estimation methods to fit the first three moments, including mean service demand (η) and SCV (c^2) estimation in the beginning of Section 5 and a MLE-based skewness (S_k) estimation as shown in Algorithm2.

Algorithm 2 Heuristics based estimation method

Input: $C \leftarrow$ Set of clustered inter-departure times $[c_1, c_2, \ldots, c_n, l_1, l_2, \ldots, l_n]$, where c_i is the centriod value and l_i is the number of points in cluster i.
$R \leftarrow$ Set of response times $[r_1, r_2, \ldots, r_n]$
$DT \leftarrow$ Set of times on the departure instant $[dt_1, dt_2, \ldots, dt_n]$
$S_k = [S_{k1}, S_{k2}, \ldots, S_{kj}] \leftarrow$ searching set of S_k
Output: $SD \leftarrow$ Estimated service demand with APH distribution
1: Compute A from DT and R
2: $\eta \leftarrow meanEstimate(A, R)$
3: $c^2 \leftarrow scvEstimate(A, R)$
4: **for** $i = 1$ to j **do**
5: Compute the first three moments $[m_1, m_2, m_3]$ from m, c^2 and S_{ki}
6: $APH(\boldsymbol{\alpha}, \boldsymbol{T}) \leftarrow APHFit[m_1, m_2, m_3]$
7: **if** $APH(\boldsymbol{\alpha}, \boldsymbol{T})$ is feasible **then**
8: Repeat execution of lines 5 to 12 in Algorithm 1
9: **end if**
10: **end for**
11: Get optimal skewness set of service demand, $S_k \leftarrow S_{ki}$ with $max(L)$
12: Compute the first three moments $[m_1, m_2, m_3]$ from η, c^2 and S_k
13: **return** $SD \leftarrow APHFit[m_1, m_2, m_3]$

The algorithm requires a set of IDTs, the departure times, the response time of each job, and the queue lengths seen upon arrivals. In Line 1, we first compute the arrival times with the departure times DT and response times R and then calculate the queue length seen upon arrivals for each job. The mean service demand is estimated based on response times and the queue length seen upon arrivals. Then the algorithm estimates c^2 by (13). The only search parameter for our method now is the skewness S_k. Here we perform MLE on the departure process with Equation (6). As in the search-based global optimization, we first generate a set of candidate skewness values $S_k = [S_{k1}, S_{k2}, \ldots]$. For each S_{ki}, we generate a queueing model with corresponding service demand and then calculate the likelihood value as Lines 5-12 in Algorithm 1. The algorithm will search on all candidates in the set, and the process is repeated with multiple candidate points for robustness. The estimated result of the skewness is finally decided on the max likelihood value. Therefore, the final result is obtained with a collection of three-parameter estimation methods.

6 Evaluation

This section introduces the experimental setup and evaluation metrics, and a comparison of our method against the baseline algorithms.

6.1 Experimental setup

To evaluate the proposed service demand distribution estimation, we conduct several experiments and compare the results to our baseline algorithm using an open-source

Table 1: Workload pattern

CPU level	Low	Medium	High
CPU utilization (U)	33%	43%	95%
Number of users (N)	50	100	300

microservice-based application called Sock Shop[2]. Sock Shop consists of 13 different services and all services communicate using REST APIs over HTTP. We use Docker Compose for the multi-container orchestration. We then generate closed workloads with different intensity using Locust[3]. In the experiment, we target a service that does not interact with a database, which avoids indirect drifts in the response time due to the state of the database.

The experimental environment is as follows. For the deployment of the application, we use a server running Ubuntu 16.04.7, and our target service is pinned to a separate CPU core. Locust is running on 6 different servers, including one host node and 5 distributed nodes to simulate the concurrent users. We experimented with populations of different numbers of users to assess the corresponding CPU utilization level as shown in Table 1.

In all the following experiments, the users' think time is exponentially distributed with a mean of 0.1 seconds.

We conduct experiments with each population in Table 1 and capture the network traffic with a dockerized tcpdump that is triggered over HTTP. During the experiments, we monitor HTTP traffic on the source and destination nodes of our target service. Then the traffic data is parsed to extract the request and response information of each request. For each different population size, we collect 10,000 HTTP request and response pairs to build up the trace dataset. Every dataset consists of the response time of each request and the time of departure instant.

Baseline algorithm. The service demand is usually modeled as exponential with the estimated mean value [33]. In our experiments, we also fit as baseline an exponential distribution for service demand.

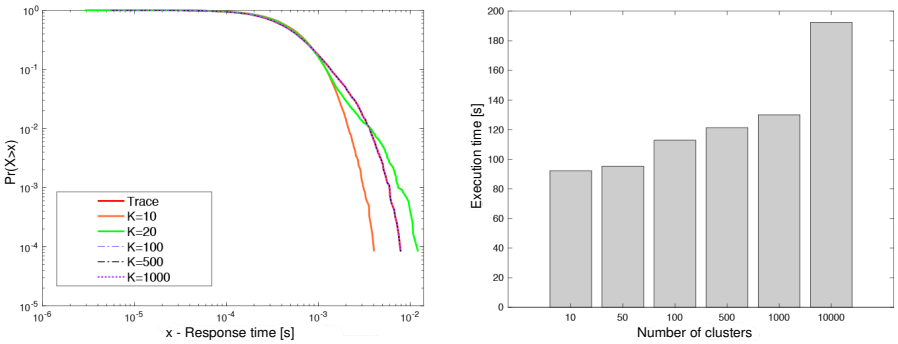
Metrics. Since the real service times of systems are usually difficult to measure, we opt to measure the response time via monitoring and use the complementary cumulative distribution function (CCDF) of the response times as our metrics. We construct the queueing model parameterized with the estimated service demand distribution and use the simulation-based JMT solver in LINE [27] to compute the corresponding response times to compare with. Thus, the only difference between the baseline and our experiments is the service demand distribution at the queueing node.

6.2 Data preprocessing and clustering

To demonstrate the trade-off between computational complexity and the approximation accuracy due to the choice of the number of clusters K, we consider a simulated experiment with 50 users in a single-server queueing system. In this experiment, we estimate

[2] https://microservices-demo.github.io/

[3] https://locust.io/



(a) (b)

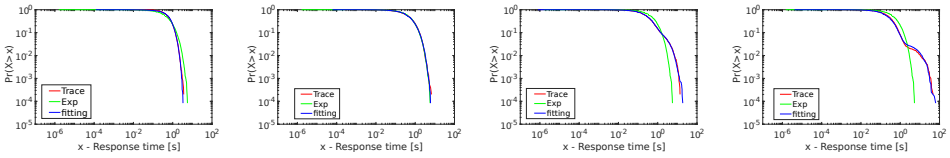
Fig. 4: The CCDF of response times (a) and the execution times (b) of maximum likelihood estimation under different number of clusters

the service demand distribution with the original data and clustered data for different values of K. Figure 4 shows the simulation results. It can be observed from Figure 4(b) that with clustered IDTs the execution time of departure process MAP modeling and log-likelihood calculation drops by almost 33% compared to the initial execution time with the whole trace. The accuracy of parameter estimation with the clustered data is evaluated in Figure 4(a) by means of CCDF diagrams. While small values of K lead to a coarse approximation, increasing K the CCDF for the clustered data rapidly converges to the CCDF estimated from the whole dataset without clustering. As can be noted from Figure 4(a), the curves become indistinguishable for $K \geq 100$. We can thus conclude that our clustering heuristics does not significantly reduce the accuracy of the estimation for K large enough (K ≥ 100 in our experiment), while significantly reducing the computational cost of the estimation.

6.3 Numerical experiment results

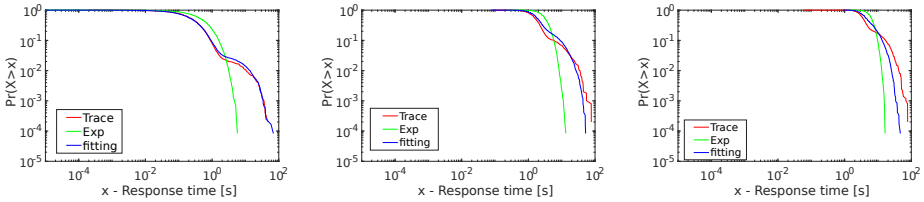
We conduct the following numerical experiments to assess the effectiveness of our global-search based method for the service demand distribution estimation. We first generate single-server queueing models that can simulate the behaviours of a simplistic microservice, setting different known values for service demand at the queue node. Then we generate samples of inter-departure times by simulating. The mean service demand η is fixed at 0.7 and different SCV are selected from $c^2 \in \{0.5, 1, 4, 16\}$. After generating sample traces via simulation, we execute Algorithm 1 and compare the estimated SD with the known values. We also compare with the baseline algorithm on the same simulated traces. The results of fitting with different service demand distributions are plotted as CCDF diagrams in Figure 5.

It can be seen from the response times distributions that our estimation method achieves good fits for all setting of c^2 and outperforms the exponential distribution especially for the tail of the distributions. For larger c^2, we can observe that only estimating the mean η of the exponential distribution cannot capture the full distribution of



(a) $c^2 = 0.5$ (b) $c^2 = 1$ (c) $c^2 = 4$ (d) $c^2 = 16$

Fig. 5: CCDF of response time for different setting of c^2



(a) $N = 1$ (b) $N = 5$ (c) $N = 10$

Fig. 6: CCDF of response time for different numbers of users N

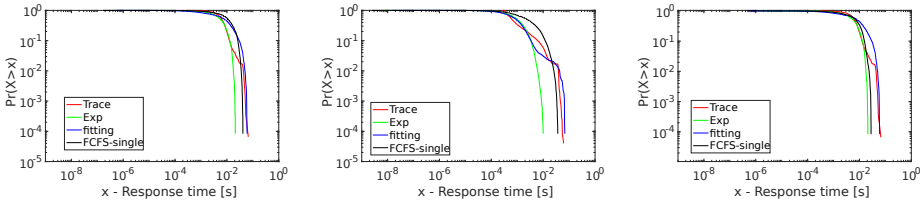
service demand, with the baseline algorithm decreasing much faster than our method, resulting in an inaccurate prediction of the response times.

We also evaluate the estimation results for different numbers of users. In this experiment, we create the simulation models for sample generation with users $N \in \{1, 5, 10\}$. The assigned parameters of service demand in the simulation models are $\eta = 0.7$ and $c^2 = 16$. We notice that with $N = 10$, the CPU utilization from simulation can reach over 95% which is close to saturation. However, compared to the baseline, the fitting results of our method under different values of N are much closer to the simulated response time distribution in the tail than the baseline estimation, as shown in Figure 6.

6.4 Analysis of Results on Measured Traces

Scheduling policy – FCFS. We now turn our attention to experiments on the real system. First, we present our experimental results for different numbers of users with FCFS scheduling strategy. For a single server system with FCFS, the service times can be obtained from sampling IDTs when the server is not idle. Therefore, for FCFS, we can directly calculate the first three moments for these sampled service times. Here we define this direct measured method as FCFS-single method. We evaluate the simulation results with FCFS-single method, baseline algorithm, and our method. The CCDF of response times for low, medium, and high CPU utilization are plotted in Figure 7, respectively. The parameter estimation results with FCFS are summarized in Table 2, and the results based on FCFS-single measurement are in Table 3.

We first observe that the FCFS-single method yields a similar accuracy of fitting the tail of the distribution under low and medium utilization level, whereas the curve based on the FCFS-single is farther away than our method for the body of the distribution.



(a) $N = 50, U = 33\%$ (b) $N = 100, U = 43\%$ (c) $N = 300, U = 95\%$

Fig. 7: CCDF of response time with different utilization for the fitted models with FCFS.

Table 2: Service demand distribution parameter estimation results for FCFS

N	$\eta \times (10^{-4})$	c^2	S_k	Likelihood $\times(10^4)$
50	8.33	8.24	11	6.62
100	6.21	11.22	57	14.31
300	3.50	8.03	12	10.64

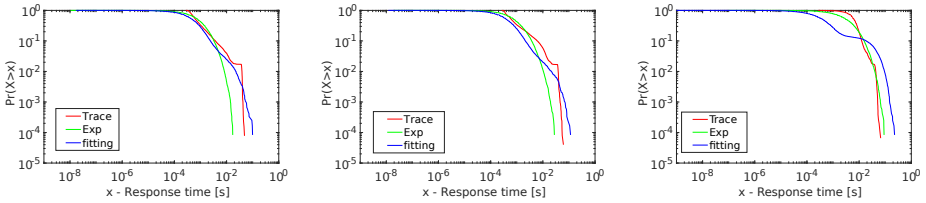
Table 3: Service demand distribution parameter estimation results for FCFS with FCFS-single method

N	$\eta \times (10^{-4})$	c^2	S_k
50	13.61	2.31	4
100	7.96	2.85	5
300	3.72	2.87	7

For higher CPU utilization, our method has a better fit for the tail of the distribution. We interpret that the real system is not precisely served by FCFS scheduling, while it exhibits some state-dependent degree of CPU sharing, and the heuristic we propose captures instances of large service variance that occur in certain system states, which can be inferred from c^2 in Table 3.

Compared to the baseline algorithm with exponential distribution, it can be seen that for all 3 different N of users our method produces a closer fit. As one can see for the body fitting, both baseline and our proposed model yield good performance. For $N = 50$ and $N = 100$, our model fits the body with better accuracy, whereas the baseline method is outperformed for $N = 300$. However, for the fitting of the tail, the baseline method is worse in terms of accuracy for all values of N. Overall, our estimated distribution achieves a better fit throughout the distribution curve, producing an approximation of the real trace distribution accurately. Compared to the baseline, our proposed method can model the heavy-tail behaviors, which is significant to model different intensive workloads.

Scheduling policy – PS. To analyze the impact of scheduling policies, we conduct experiments on the queueing model with $N \in \{50, 100, 300\}$ with PS scheduling policy. The parameter estimation results with PS is shown in Table 4 and the fitting comparison is plotted in Figure 8. We can see from the figure that switching from FCFS to PS impacts the results of the baseline in a significant manner. First, comparing the estimated



(a) $N = 50, U = 33\%$ (b) $N = 100, U = 43\%$ (c) $N = 300, U = 95\%$

Fig. 8: CCDF of response time with different utilization for the fitted models with PS.

Table 4: Service demand distribution parameter estimation results for PS

N	$\eta \times (10^{-4})$	c^2	S_k	Likelihood $\times(10^4)$
50	8.33	8.24	9	6.11
100	6.21	11.22	10	14.53
300	3.50	8.03	4	10.68

skewness to the results for FCFS, we can observe smaller values of skewness. As it can be seen in Figure 8(a) and (b), the baseline method first fits well at the beginning, and it decays faster from the middle body, ultimately not able to capture the tail. On the other hand, our method fits the body of the distribution and achieves a slower decay for the tail, showing a more accurate fit for the distribution.

While for low and medium CPU utilization our method outperforms the baseline, especially for the tail of the distribution, Figure 8(c) shows that with the increasing number of simulated users, and CPU utilization close to 100%, the baseline method achieves a closer fit to the data with PS scheduling.

It is not difficult to see from Table 2 and Table 4 that the likelihood values for $N = 50, 100, 300$ with PS are close to the one with FCFS. In reality, the actual system scheduling will factor in several elements such as caching, memory bandwidth, and operating system scheduling, which are neither perfectly PS nor FCFS. Thus, our results indicate that either models provide a reasonable approximation to the observed system behaviour, but FCFS appears more suitable to model heavy loads.

Summarizing, the previous results indicate that in the majority of instances the proposed method is able to fit the service process characteristics of microservices with higher fidelity than simple exponential models, especially capturing tail behaviors with higher accuracy.

7 Related work

To enable automatic generation for accurate performance models, model parameterization brings out an important problem of resource demand estimation. There are several works using regression methods. Rolia [30] introduce the resource demand estimation problem with linear regression techniques. In general, linear regression has been employed to solve the service demand estimation mostly based on utilization [9, 10] and

response time [26]. Neural networks like recurrent neural networks (RNN) can be applied to estimate resource demands for the ability to predict time series data like the works in [12]. Machine learning can also help to select the optimal approaches for estimation on account of varieties of existing resource demand approaches [19]. However, most of the mentioned approaches based on regression only enable to obtain the mean value of the resource demand instead of full distributions, lacking higher-order properties of the demand. These time-series based prediction is not suitable for assigning the required parameters for performance models.

There are many existing works on modeling with a queueing system in which the arrivals of users are characterized with a Poisson arrival process and an exponentially distributed service time [6, 18]. However, they are not accurate enough to capture and represent the behaviors of real systems, especially for the tail of the service demand distribution. To characterize more accurate service demand, PH distribution provides possibilities to approximate arbitrary distributions, which has been used in studies of modeling and simulation like [11, 25].

8 Conclusion

In this paper, we have introduced a service demand distribution estimation method by using Markovian arrival processes for microservices. We have first presented a closed queueing model for a microservice and characterized the service demand of the queue node with an APH distribution. We have then proposed to model the departure process at the queue node of this model with a MAP, in which the parameters of service demand can be exposed. The service demand distribution can be estimated with maximum likelihood based on the departure process MAP. We applied the global search and a heuristic estimation method on solving the optimization problem. The proposed heuristic estimation method can effectively reduce the computational cost compared to the global search approach. We further showed that the proposed estimation method yields a better performance on fitting real traces of microservices compared to the baseline.

As we consider a single-server queue with a single-class workload in this work, our future research aims at generalizing the estimation method with multi-classes models. In the next step, we plan to model the departure process with a marked MAP, which may increase the model complexity and the likelihood of CTMC explosion. To deal with the problems of multi-class queues, multi-class classification methods like the one-against-all algorithm may provide the feasibility to reduce the number of classes. Aggregation methods may be explored to address this issue.

Acknowledgement

The work of Giuliano Casale has been partly funded by the EU's Horizon 2020 program under grant agreement No. 825040.

References

1. Andersen, A.T., Neuts, M.F., Nielsen, B.F.: Lower order moments of inter-transition times in the stationary QBD process. Methodology and Computing in Applied Probability **2**(4), 339–357 (2000)
2. Bass, L., Weber, I., Zhu, L.: DevOps: A software architect's perspective. Addison-Wesley Professional (2015)
3. Bezemer, C.P., Eismann, S., Ferme, V., Grohmann, J., Heinrich, R., Jamshidi, P., Shang, W., van Hoorn, A., Villavicencio, M., Walter, J., et al.: How is performance addressed in devops? In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering. pp. 45–50 (2019)
4. Binz, T., Breitenbücher, U., Kopp, O., Leymann, F.: TOSCA: portable automated deployment and management of cloud applications. In: Advanced Web Services, pp. 527–549. Springer (2014)
5. Bobbio, A., Horváth, A., Telek, M.: Matching three moments with minimal acyclic phase type distributions. Stochastic models **21**(2-3), 303–326 (2005)
6. Bramson, M., Lu, Y., Prabhakar, B.: Randomized load balancing with general service time distributions. ACM SIGMETRICS performance evaluation review **38**(1), 275–286 (2010)
7. Brosig, F., Kounev, S., Krogmann, K.: Automated extraction of palladio component models from running enterprise java applications. In: Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools. pp. 1–10 (2009)
8. Casale, G., Artač, M., van den Heuvel, W.J., van Hoorn, A., Jakovits, P., Leymann, F., Long, M., Papanikolaou, V., Presenza, D., Russo, A., et al.: Radon: rational decomposition and orchestration for serverless computing. SICS Software-Intensive Cyber-Physical Systems **35**(1), 77–87 (2020)
9. Casale, G., Cremonesi, P., Turrin, R.: How to select significant workloads in performance models. In: CMG Conference Proceedings. pp. 58–108. Citeseer (2007)
10. Casale, G., Cremonesi, P., Turrin, R.: Robust workload estimation in queueing network performance models. In: 16th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP 2008). pp. 183–187. IEEE (2008)
11. Dudin, S., Dudina, O.: Retrial multi-server queuing system with PHF service time distribution as a model of a channel with unreliable transmission of information. Applied Mathematical Modelling **65**, 676–695 (2019)
12. Duggan, M., Mason, K., Duggan, J., Howley, E., Barrett, E.: Predicting host cpu utilization in cloud computing using recurrent neural networks. In: 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST). pp. 67–72. IEEE (2017)
13. Franks, G., Al-Omari, T., Woodside, M., Das, O., Derisavi, S.: Enhanced modeling and solution of layered queueing networks. IEEE Transactions on Software Engineering **35**(2), 148–161 (2008)
14. Garetto, M., Cigno, R.L., Meo, M., Marsan, M.A.: A detailed and accurate closed queueing network model of many interacting TCP flows. In: Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213). vol. 3, pp. 1706–1715. IEEE (2001)
15. Horváth, A., Telek, M.: Phfit: A general phase-type fitting tool. In: International Conference on Modelling Techniques and Tools for Computer Performance Evaluation. pp. 82–91. Springer (2002)
16. Horváth, G., Telek, M.: Butools 2: a rich toolbox for markovian performance evaluation. In: VALUETOOLS (2016)
17. Krishna, K., Murty, M.N.: Genetic k-means algorithm. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **29**(3), 433–439 (1999)

18. Krishnamoorthy, A., Manikandan, R., Lakshmy, B.: A revisit to queueing-inventory system with positive service time. Annals of Operations Research **233**(1), 221–236 (2015)
19. Liao, S., Zhang, H., Shu, G., Li, J.: Adaptive resource prediction in the cloud using linear stacking model. In: 2017 Fifth International Conference on Advanced Cloud and Big Data (CBD). pp. 33–38. IEEE (2017)
20. Marsan, M.A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with generalized stochastic Petri nets, vol. 292. Wiley New York (1995)
21. Mazkatli, M., Koziolek, A.: Continuous integration of performance model. In: Companion of the 2018 ACM/SPEC International Conference on Performance Engineering. pp. 153–158 (2018)
22. Neuts, M.F.: Models based on the Markovian arrival process. IEICE Transactions on Communications **75**(12), 1255–1265 (1992)
23. O'Cinneide, C.A.: Characterization of phase-type distributions. Stochastic Models **6**(1), 1–57 (1990)
24. Pan, J.X., Fang, K.T.: Maximum likelihood estimation. In: Growth curve models and statistical diagnostics, pp. 77–158. Springer (2002)
25. Parini, A., Pattavina, A.: Modelling voice call interarrival and holding time distributions in mobile networks. In: Proc. of ITC 2005. pp. 729–738 (2005)
26. Pérez, J.F., Pacheco-Sanchez, S., Casale, G.: An offline demand estimation method for multi-threaded applications. In: 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems. pp. 21–30. IEEE (2013)
27. Pérez, J.F., Casale, G.: Line: Evaluating software applications in unreliable environments. IEEE Transactions on Reliability **66**(3), 837–853 (2017)
28. Petriu, D.C., Shen, H.: Applying the UML performance profile: Graph grammar-based derivation of lqn models from uml specifications. In: International Conference on Modelling Techniques and Tools for Computer Performance Evaluation. pp. 159–177. Springer (2002)
29. Reibman, A., Trivedi, K.: Numerical transient analysis of markov models. Computers & Operations Research **15**(1), 19–36 (1988)
30. Rolia, J., Vetland, V.: Correlating resource demand information with arm data for application services. In: Proceedings of the 1st international workshop on Software and performance. pp. 219–230 (1998)
31. Schweitzer, P.: Approximate analysis of multiclass closed networks of queues. J. ACM **29**(2) (1981)
32. Sidje, R.B., Burrage, K., MacNamara, S.: Inexact rouniformization method for computing transient distributions of markov chains. SIAM Journal on Scientific Computing **29**(6), 2562–2580 (2007)
33. Spinner, S., Casale, G., Brosig, F., Kounev, S.: Evaluating approaches to resource demand estimation. Performance Evaluation **92**, 51–71 (2015)
34. Wang, W., Huang, X., Qin, X., Zhang, W., Wei, J., Zhong, H.: Application-level cpu consumption estimation: Towards performance isolation of multi-tenancy web applications. In: 2012 IEEE Fifth International Conference on Cloud Computing. pp. 439–446. IEEE (2012)