Markovian Workload Characterization for QoS Prediction in the Cloud

Sergio Pacheco-Sanchez[*,‡], Giuliano Casale[†], Bryan Scotney[‡], Sally McClean[‡], Gerard Parr[‡] and Stephen Dawson[*]
[*] *SAP Research Center Belfast, Belfast BT3 9DT, UK, {sergio.pacheco-sanchez, stephen.dawson}@sap.com*
[†] *Imperial College London, Dept. of Computing, London SW7 2AZ, UK, g.casale@imperial.ac.uk*
[‡] *Univ. of Ulster, School of Comp. and Inf. Eng., Coleraine BT52 1SA, UK, {bw.scotney, si.mcclean, gp.parr}@ulster.ac.uk*

***Abstract*—Resource allocation in the cloud is usually driven by performance predictions, such as estimates of the future incoming load to the servers or of the quality-of-service (QoS) offered by applications to end users. In this context, characterizing web workload fluctuations in an accurate way is fundamental to understand how to provision cloud resources under time-varying traffic intensities. In this paper, we investigate the Markovian Arrival Processes (MAP) and the related MAP/MAP/1 queueing model as a tool for performance prediction of servers deployed in the cloud.**
MAPs are a special class of Markov models used as a compact description of the time-varying characteristics of workloads. In addition, MAPs can fit heavy-tail distributions, that are common in HTTP traffic, and can be easily integrated within analytical queueing models to efficiently predict system performance without simulating. By comparison with trace-driven simulation, we observe that existing techniques for MAP parameterization from HTTP log files often lead to inaccurate performance predictions. We then define a maximum likelihood method for fitting MAP parameters based on data commonly available in Apache log files, and a new technique to cope with batch arrivals, which are notoriously difficult to model accurately. Numerical experiments demonstrate the accuracy of our approach for performance prediction of web systems.

***Keywords*-Workload prediction; Quality-of-Service; Performance prediction; Markov models**

I. Introduction

Web workload modeling and prediction are fundamental to support service management tasks such as deployment and provisioning, to design cloud computing solutions, to select load balancing and scheduling policies, and for capacity planning exercises. As data centers become larger and their workloads more complex, performing such activities through trial and error becomes clearly intractable. At the large scale these systems operate on, there is an increasing need for effective models to provide insights on expected performance and future resource usage levels. Not surprisingly, the lack of performance predictability is listed as one of the critical obstacles to the growth of cloud computing [2].

In this work we focus on web workloads, which are usually modeled as a stochastic process in order to capture the uncertainty in their future evolution. An important class of processes used for web traffic modeling is the Markov Modulated Poisson Process (MMPP) [14], which is a class of continuous-time hidden Markov models that can be easily integrated into a queueing model for performance prediction. In a MMPP, the active state of a hidden Markov chain determines the current rate of arrival of requests to a system, e.g., a web server. In state k of the hidden Markov chain, arrivals follow a Poisson process with rate λ_k. However, since the active state changes over time, the modulation of Poisson streams with different rates λ_k allows MMPPs to describe complex time-varying workloads that are *not* exponentially distributed. If the MMPP parameters are fitted appropriately, the arrival events generated by the model can match in statistical terms (e.g., same probability distribution and correlations) those observed in a measured trace. Markovian Arrival Processes (MAP) are a compact, tractable and expressive class of stochastic models that generalizes the MMPP framework. Similarly to MMPPs, MAPs capture not only the moments of a probability distribution, but also the autocorrelation and, more generally, the temporal dependence of a time series. However, MAPs are a superset of the MMPP model since they can fit a larger variety of time-varying patterns, e.g., negative autocorrelations that cannot be fitted by MMPPs. Good fitting methods for MAPs have only been developed in recent years [4], yet little work has been done towards applying these methods to HTTP traffic. A clear advantage of MMPPs/MAPs with respect to other workload models is that, being Markovian, they can be readily integrated within analytical models of queueing systems. Such queueing models can then be used to *inexpensively* compute performance metrics, such as server utilization, queue length of requests waiting to be served and expected response times at server side. Queueing-based performance models that require minutes to be simulated accurately require just a few seconds or milliseconds to be solved exactly with the Markov models we use. In this paper we investigate further the effectiveness of these models and show how to apply MAPs for web workload characterization and fitting. This provides a quantitative methodology for QoS prediction and performance evaluation of servers in the cloud.

The main contributions of this paper are two-fold:

The work of Sergio Pacheco-Sanchez has been co-funded by the Northern Ireland government within the InvestNI/SAP MORE and VIRTEX projects. The work of Giuliano Casale has been supported by the Imperial College Junior Research Fellowship. We thank Giuseppe Serazzi, Politecnico di Milano, for providing the logfile traces used in the paper.

1) A maximum likelihood method for fitting a MAP to the web traffic measurements collected in commonly-available HTTP web server traces.
2) A methodology to parameterize the MAP/MAP/1 queueing model for web server performance prediction. The methodology supports the handling of short traces during the modeling and simulation activities, the different requests types in HTTP workloads, and can account for batches of requests that arrive simultaneously to the system.

The remainder of the paper is organized as follows. Section II gives an overview of related work, and highlights the relevance of our work for cloud computing. Section III gives an overview of MAPs. Section IV presents the reference system for our study together with a description of the data analyzed and the underlying analytical model. The proposed methodology is presented in Section V. In Section VI we show experimental results. Section VII offers conclusions.

II. Related Work

Markovian workload characterization is motivated by the fact that an effective analysis technique, the matrix geometric method, is available for the evaluation of the related queueing models [15]. In that respect, a study similar to ours proposes a methodology to construct a MAP/PH/1 queueing model fitted to web server data coming from stationary intervals of HTTP traffic that consider only static requests [17]. In [1] it is proposed to model the arrival stream of requests by means of MMPPs parameterized with just two states. Markov models with a small number of states are usually insufficient to capture the complex characteristics of traces collected in web access logs, as we show in Section VI.

The proposed MAP-based quantitative models may be utilized as a tool for resource allocation in the cloud. Research done in QoS management is vast and frequently accounts for various aspects, namely charges for service, commitment to provide a specified level of service, and penalties [8], [21]. In a real deployment, both server capacity and admission thresholds should change dynamically in response to system workload variations. In this respect, [22] proposes a hybrid cloud computing model which routes the incoming requests to shared infrastructures in cases where the base system is under a certain load. Its prediction capability relies on the estimation of the request rates to the corresponding objects. Approaches exist in the literature to provision resources accounting for workload variations [1], [18], however models are not learned from real traffic which restricts predictive capabilities.

Summarizing, a key research challenge of the QoS management area lies in developing workload models that may help in taking decisions based on accurate predictions. This leads us to use the MAP/MAP/1 model as a simple analytical technique for modeling complex time-varying workloads.

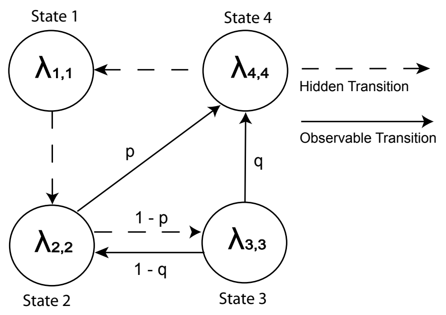


Figure 1. Example of Markovian Arrival Process (MAP)

III. Overview of Markovian Arrival Processes

We now give definitions required to understand the Markovian arrival process (MAP). This is used in the rest of the paper to describe in a compact mathematical model the characteristics of a time series of values, for instance the inter-arrival times (IATs) of HTTP requests. An example MAP is given in Figure 1. In this specific case, the model is composed by $J = 4$ states. The active state at time t is $X(t) \in \{1, 2, \ldots, J\}$. Assume that the model is in state k, then it spends t_k time before moving into state $j \neq k$; we assume that t_k follows an exponential distribution $\Pr(t_k = t) = \lambda_{k,k} e^{-\lambda_{k,k} t}$, thus $X(t)$ is a continuous-time Markov chain (CTMC). The destination state j after a jump is selected according to probabilities $p_{k,j}$, $\sum_{j=1}^{J} p_{k,j} = 1$.

A MAP extends a CTMC by introducing the semantics for defining IATs. Upon jumping from state k to j a MAP defines probabilities $p^h_{k,j}$ and $p^o_{k,j}$, $p^h_{k,j} + p^o_{k,j} = p_{k,j}$, that the state transition will be *hidden* or *observable*, respectively. A hidden transition has the only effect of changing the active state $X(t)$. An observable transition additionally leads to the emission of a *sample*. In other words, an IAT sample of a measured trace is modeled in a MAP as the time elapsed between successive activation of any two observable transitions. For example, in Figure 1, if the MAP is first initialized in state 1 where it spends t_1 time, jumps to state 2 waiting t_2 units of time and then takes the transition to state 4, then the sample value generated is $s_0 = t_1 + t_2$. The next sample s_1 is similarly generated starting from state 4; thus, the time spent in state 4 is included in s_1. Successive visits to the same state are all cumulatively accounted for in the sample value generated. Note also that since sample s_i is generated according to the target state of the observable transition that defines s_{i-1}, a careful definition of observable state transitions can create statistical correlations between consecutive samples generated by a MAP.

Following these definitions, a MAP is able to generate as the time passes an increasing number of samples s_i, $i \geq 0$. If the MAP parameters are appropriately chosen, one can impose the statistical properties of these samples in order to fit the characteristics of a measured trace. Once this is achieved, the MAP provides a mathematical model for the trace. For example, if s_i is interpreted as the IAT between successive HTTP requests arriving to a server, then the MAP is a model for the incoming web traffic. Alternatively, a MAP can model the service times (SVCTs) of requests, in this case the samples s_i are seen as service times. After fitting MAP models for HTTP requests IATs and SVCTs, one can study by efficient means the MAP/MAP/1 queueing system[1] that models the expected performance of the web server. This can be done much faster than with simulation and thus helps in optimization-based QoS management to explore a wider set of decision alternatives.

A. Mathematical Description of MAPs

The following compact notation is often used to summarize the relevant parameters of a MAP. A MAP is represented by a matrix pair $(\boldsymbol{D}_0, \boldsymbol{D}_1)$, where both matrices have order J equal to the number of states. For the example in Figure 1

$$\mathbf{D_0} = \begin{bmatrix} -\lambda_{1,1} & p^h_{1,2}\lambda_{1,1} & 0 & 0 \\ 0 & -\lambda_{2,2} & p^h_{2,3}\lambda_{2,2} & 0 \\ 0 & 0 & -\lambda_{3,3} & 0 \\ p^h_{4,1}\lambda_{4,4} & 0 & 0 & -\lambda_{4,4} \end{bmatrix}$$

$$\mathbf{D_1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & p^o_{2,4}\lambda_{2,2} \\ 0 & p^o_{3,2}\lambda_{3,3} & 0 & p^o_{3,4}\lambda_{3,3} \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

where $p^h_{1,2} = p^h_{4,1} = 1$, $p^h_{2,3} = 1 - p$, $p^o_{2,4} = p$, $p^o_{3,2} = 1 - q$, $p^o_{3,4} = q$. Thus $\mathbf{D_0}$ provides the rates of hidden transitions, while $\mathbf{D_1}$ gives those for observable transitions.

The $(\boldsymbol{D}_0, \boldsymbol{D}_1)$ notation enables the compact description of the statistical properties of the samples generated by the MAP. Here, we always refer to the statistical properties defined on a stationary time series of MAP samples. It is known that this series can be obtained by initializing the MAP in state j according to probability $\pi_j \in \vec{\pi}$, where the row vector $\vec{\pi}$ is the left eigenvector of $\boldsymbol{P} = (-\boldsymbol{D}_0)^{-1}\boldsymbol{D}_1$ such that $\vec{\pi}\boldsymbol{P} = \vec{\pi}$, $\vec{\pi}\vec{1} = 1$, where $\vec{1} = (1, 1, \ldots, 1)^T$ is a vector of ones of length J. The statistics of a sample s generated by a (stationary) MAP are:

- Cumulative distribution function

$$F(X) = \Pr[s \leq X] = 1 - \vec{\pi}\, e^{\boldsymbol{D}_0 X}\, \vec{1}, \qquad (1)$$

- Moments of the sample distribution

$$E[X^k] = k!\, \vec{\pi}\, (-\boldsymbol{D}_0)^{-k}\, \vec{1}, \quad k \geq 1, \qquad (2)$$

  (In particular the mean arrival rate is $\lambda = 1/E[X]$.)

- Joint moments of the sample distribution

$$E[X_0 X_k] = \vec{\pi}\, (-\boldsymbol{D}_0)^{-1}\boldsymbol{P}^k(-\boldsymbol{D}_0)^{-1}\, \vec{1}, \qquad (3)$$

  where X_0 and X_k are samples that are $k \geq 1$ lags apart.

- Autocorrelation function coefficient at lag k (ACF-k)

$$\rho_k = \frac{E[X_0 X_k] - E[X]^2}{E[X^2] - E[X]^2} \qquad (4)$$

If $\rho_k = 0$, for all k, there are no correlations between the samples. In this special case, the MAP reduces to a *phase-type* (PH) distribution. A PH distribution can model the moments or cumulative distribution function of a time series, but not time-varying patterns. Thus, a trace T and a trace T' obtained by the random shuffling of T would have the same PH distribution model, but different MAP models.

IV. System Characteristics

A. Web Server Environment

The Apache logfile trace captures the incoming HTTP traffic to a departmental web server of the Politecnico di Milano university. The trace covers a period of a week from the 3^{rd} September 04:27am to the 10^{th} September 03:53am, 2006. The web server access log has information that includes the timestamp T_n of the n-th request with a default resolution value of one second, and the size of the object returned to the client in bytes. In addition, we are able to determine whether the content served to clients is static or dynamic. We assume the server handles requests for static content from main memory, while dynamic requests are forwarded first to the back-end before replying to the client. We assume the time to transmit an object to the client is accurately estimated by its size [23], and in our case it fully represents the resource consumption of static requests. On the other hand, the resource consumption of dynamic requests has been approximated by aggregating the time to generate the content, including database and application server activity, and the time to transfer it through the network. The time to generate dynamic content is drawn from a Lognormal distribution with mean $\mu = 3.275$ ms, and squared coefficient of variation (i.e., square of the ratio of the standard deviation to the mean) $c^2 = 11.56$, parameters obtained from [23] for a TPC-W workload.

B. Web Server Data

The average incoming traffic to a web server changes considerably with the time of the day, an effect that does not comply to stationarity assumptions used in statistical modeling. A common approach that is followed in these cases is to break down traces into smaller datasets which are representative of a period where the average behavior of a server may be assumed stationary. We have decided to adopt a strategy in our analysis for splitting the web trace into blocks of one hour of traffic. Thus, we define the i-th 60-minute traffic block $B^{(i)}$ as an ordered sequence of $len(B^{(i)})$ HTTP requests sent to the web server. The download of a web page is broken down into individual

[1]Kendall's notation $A/B/c$ stands for a queue where arrivals follow process A, service times follows process B and there are c servers (e.g., CPUs) processing requests. For instance, M/M/1 stands for a single-server model where arrival and service processes are both Poisson.

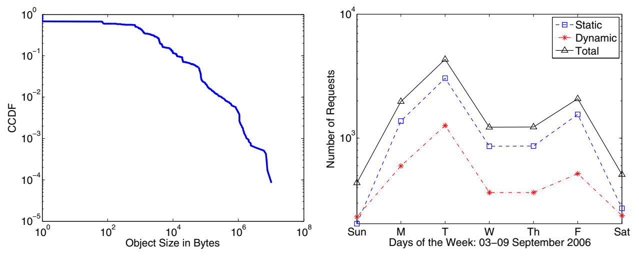
Table I
SUMMARY OF TRACE CHARACTERISTICS.

$B^{(i)}$	Size dataset	Stationary IAT	Stationary SVCT	ACF-1 IAT	ACF-1 SVCT	c^2 IAT	c^2 SVCT
1	437	Yes	No	0.28	0.89	24	0.88
2	1972	No	No	0.42	0.81	17	2.29
3	4313	No	No	0.14	0.70	30	2.57
4	1228	No	No	0.16	0.77	14	2.27
5	1229	Yes	No	0.11	0.68	28	2.48
6	2077	No	No	0.13	0.78	20	2.87
7	510	Yes	No	0.08	0.80	39	1.11

HTTP requests to the objects that compose the page. Each 60-minute block of traffic is represented by the time of arrival of the first request, indicated with $time(B^{(i)})$, and by the set of inter-request times $V_1^{(i)}, V_2^{(i)}, \ldots, V_{len(B^{(i)})-1}^{(i)}$ occurring between the following requests. Computing inter-request times between adjacent requests whose timestamps are logged at a coarse one-second resolution frequently results in the generation of sequences of zeros. This is expected for web sites that receive at least tens or even hundreds of requests within a second. To reduce the noise introduced by such a coarse resolution, we randomize the inter-arrival time between the last request falling in a period of one-second and the first request in the next one by a uniform random number in $[0, 1]$ seconds. This results in smoother probability distributions which are easier to fit using mathematical models.

To illustrate our methodology, we have focused on a heterogeneous set of seven blocks representing the traffic for each day of the week observed approximately between 7:37am to 8:37am, in total consisting of 11,766 requests. In Figure 2(a) we plot the CCDF of the size of objects, which appears to follow a heavy-tailed distribution. Heavy-tails in object sizes are then inherited by request SVCT distributions thus complicating the modeling.

Figure 2(b) depicts the variation in the type of incoming requests to the system. On average, the traffic is heavily dominated by static content which accounts for the 65% vs. 35% of dynamic requests. A summary of the statistical characteristics of the blocks analyzed, that include the sequences of zeros within the IAT dataset, is shown in Table I. Stationarity is assessed by the KPSS test [13]; the squared coefficient of variation $c^2 > 1$ indicates more variability than in a Poisson process. To cope with the residual non-stationarity in the blocks, we assume in the rest of the paper that simulation-based predictions are obtained by a cyclic concatenation of the trace in order to run one long simulation experiment for observing the convergence of performance metric estimates. This is more acceptable than operating the same type of repetition on the original 24-hour trace, since 1 hour is a more representative time scale for cloud resource allocation decisions.



(a) CCDF of file sizes (b) Workload mix

Figure 2. (a)Heavy-tailed distribution; (b) Load of request types per block.

C. Web Server Performance Model

The performance of the web server system under study is modeled by the MAP/MAP/1 queue. The MAP model fitted to an IAT block is represented by $(\boldsymbol{D}_0, \boldsymbol{D}_1)$, whereas we denote by $(\boldsymbol{D}'_0, \boldsymbol{D}'_1)$ the MAP model fitted to the corresponding SVCT block. According to this notation, the MAP/MAP/1 queue is a continuous-time Markov chain with infinitesimal generator

$$\mathbf{Q} = \begin{bmatrix} \bar{\boldsymbol{A}}_0 & \bar{\boldsymbol{A}}_1 & & & \\ \boldsymbol{A}_{-1} & \boldsymbol{A}_0 & \boldsymbol{A}_1 & & \\ & \boldsymbol{A}_{-1} & \boldsymbol{A}_0 & \boldsymbol{A}_1 & \\ & & \ddots & \ddots & \ddots \end{bmatrix} \quad (5)$$

where $\boldsymbol{A}_1 = \boldsymbol{D}_1 \otimes \boldsymbol{I}$, $\boldsymbol{A}_0 = \boldsymbol{D}_0 \otimes \boldsymbol{D}'_0$,$\boldsymbol{A}_{-1} = \boldsymbol{I} \otimes \boldsymbol{D}'_1$, $\bar{\boldsymbol{A}}_0 = \boldsymbol{D}_0 \otimes \boldsymbol{I}$, being $\boldsymbol{I}$ the identity matrix and $\otimes$ the Kronecker product. The above matrix is called a quasi-birth-death (QBD) process since it generalizes by block transitions the classic birth-death processes of the M/M/1 queue which has scalar transition rates. This enables the integration of complex workload descriptions obtained for IAT and SVCT from the Apache logfile traces into the queueing analysis of the web server. The probability for states that pertain to the block row $k = 0, 1, \ldots$ is described by a vector $\vec{v}_k$ such that $\vec{v}_k\vec{1} = v_k$ is the probability of observing k requests in queue. In particular, v_0 describes the probability for the queue being empty, thus $v_0 = 1-\rho$ where ρ is the utilization of the server. The matrix geometric method proves under mild assumptions that there exist a matrix $\boldsymbol{R}$ such that [15]

$$\vec{v}_k = \vec{v}_0\mathbf{R}^k, \quad k > 0. \quad (6)$$

It is found that $\mathbf{R}$ is the minimal non-negative solution of equation[2] $\boldsymbol{A_1} + \mathbf{R}\boldsymbol{A_0} + \mathbf{R}^2\boldsymbol{A_{-1}} = 0$ and $\vec{v}_0$ is the solution to equations

$$\vec{v}_0(\bar{\boldsymbol{A}}_0 + \mathbf{R}\boldsymbol{A}_{-1}) = 0, \quad \vec{v}_0(\boldsymbol{I} - \mathbf{R})^{-1}\vec{1} = 1. \quad (7)$$

Thus, (6) provides a simple way to study the queue-length probability distribution in a model for a web server with

[2]We have used the free SMCSolver for MATLAB to compute $\mathbf{R}$ and the vectors v_k, $k \geq 0$ [5]. Alternatively, they can be computed with the MAMSolver tool available in C++ [16]. Simple algorithms for computing $\mathbf{R}$ are described in [6, pp. 133].

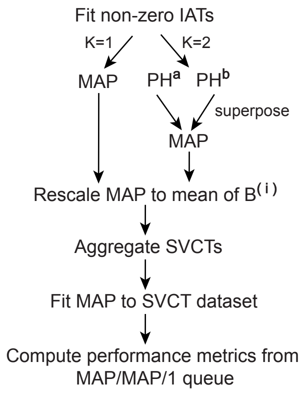


Figure 3. Model parameterization methodology

IATs $(\boldsymbol{D}_0, \boldsymbol{D}_1)$ and SVCTs $(\boldsymbol{D}'_0, \boldsymbol{D}'_1)$ fitted from a real logfile trace. Furthermore, we can compute various performance measures with ease. The tail distribution of the number of web server requests in the system which is given by

$$\boldsymbol{P}[Q > x] = \sum_{k=x+1}^{\infty} \vec{v}_k \, \vec{1}, \quad x \geq 0, \tag{8}$$

The mean queue length is computed as

$$\boldsymbol{E}[\,Q\,] = \sum_{k=0}^{\infty} k \; \vec{v}_0 \mathbf{R}^k \, \vec{1} = \vec{v}_0 \mathbf{R}(\boldsymbol{I} - \mathbf{R})^{-2} \, \vec{1}, \tag{9}$$

The mean response time of system requests is obtained by Little's law [6] and (9) as

$$\boldsymbol{E}[R] = \lambda^{-1}(\vec{v}_0 \mathbf{R}(\boldsymbol{I} - \mathbf{R})^{-2} \, \vec{1}), \tag{10}$$

which is a fundamental metric used in QoS prediction.

V. Model Parameterization Methodology

Figure 3 illustrates the methodology we propose for fitting logfiles into MAPs that can be used to analyze the MAP/MAP/1 queue. Two separate fitting methods are used for IATs of HTTP requests and their SVCTs. Here, we develop multiclass models ($K = 2$), which distinguish between static and dynamic requests, and single class models ($K = 1$). Details on the individual steps of the methodology are given in the next subsections.

A. Maximum Likelihood Estimation

Maximum likelihood (ML) estimation is a classical approach for fitting workload models [6]. This technique is particularly useful for estimating models from short datasets, such as the ones considered in this study where a single block $B^{(i)}$ provides only few hundreds or thousands samples. Before outlining how we apply MAPs to the problem under study, we overview the ML fitting method and focus the discussion on IATs between incoming requests.

Let us first focus on the single class case $K = 1$. Using the MAP notation, a PH distribution is a special case of MAP where it can be shown that $\boldsymbol{D}_1 = -\boldsymbol{D}_0 \vec{1} \vec{\pi}$, thus a pair $(\vec{\pi}, \boldsymbol{D}_0)$ is sufficient to specify a PH distribution. As observed earlier, this mathematical definition implies that the samples generated by the model are independent and identically distributed (i.i.d.), thus $\rho_k = 0$, $k \geq 1$. Stemming from the above properties, the maximum likelihood method seeks for a pair $(\vec{\pi}, \boldsymbol{D}_0)$ which maximizes the probability of observing the dataset obtained, that is, for block i

$$\max_{(\vec{\pi}, \boldsymbol{D}_0)} \mathbb{P}[V_1^{(i)}, \ldots, V_2^{(i)}, \ldots, V_{len(B^{(i)})-1}^{(i)} | \vec{\pi}, \boldsymbol{D}_0], \tag{11}$$

subject to $\vec{\pi}\vec{1} = 1$ and the sign constraints for the entries in $\boldsymbol{D}_0$. Approximating the inter-arrival times as independent random variables and taking the logarithm of the resulting expression we get

$$\max_{(\vec{\pi}, \boldsymbol{D}_0)} \sum_{j=1}^{len(B_i)-1} \log \mathbb{P}[V_j^{(i)} | \vec{\pi}, \boldsymbol{D}_0], \tag{12}$$

where the argument is called the *likelihood function* for the IATs in block i. In particular, for a PH distribution it is

$$\mathbb{P}[V_j^{(i)} | \vec{\pi}, \boldsymbol{D}_0] = \vec{\pi} e^{\boldsymbol{D}_0 V_j^{(i)}} (-\boldsymbol{D}_0) \vec{1}, \tag{13}$$

Summarizing, the maximum likelihood method obtains a PH distribution describing the IATs by maximizing (12) using expression (13) and standard nonlinear solvers[3]. The corresponding MAP used in the MAP/MAP/1 queue has the same $\boldsymbol{D}_0$ matrix and $\boldsymbol{D}_1 = -\boldsymbol{D}_0 \vec{1} \vec{\pi}$ and is re-scaled to the mean of each trace B^i. This is obtained easily by multiplying all rates in $\boldsymbol{D}_0$ and $\boldsymbol{D}_1$ by $c = E[X_{old}]/E[X_{new}]$, being $E[X_{old}]$ the current mean of the MAP and $E[X_{new}]$ the desired value. This provides a single class approach for fitting a concatenated interval of traffic containing the sets of IATs $V_1^{(i)}, V_2^{(i)}, \ldots, V_{len(B^{(i)})-1}^{(i)}$, for all $1 \leq i \leq I$, where I is the total number of blocks of traffic analyzed. We have chosen $I = 7$, each $B^{(i)}$ representing a block of the same hour of traffic for every day of the week. The rationale is that those blocks represent short traces, therefore the need to concatenate them. A larger interval of traffic may represent the IATs of one day. Note that the single class case ignores the correlation between requests in the arrivals. It is here preferred to a fitting of a general MAP process since we found the latter to underestimate performance systematically. We conjecture this to be the case due to the fact that autocorrelation estimates are quite unreliable for short traces such as the ones we considered in this study; thus, it is hard to fit appropriate values for the autocorrelation coefficients. We describe below a more accurate approach that uses MAPs to fit the time-varying patterns arising from interleaving static and dynamic requests. The resulting MAPs can be autocorrelated and thus are inherently more

[3]In this paper, we have maximized (12) using the fmincon function integrated in MATLAB's optimization toolbox.

expressive than the single class modeling approach we have described so far.

Multiclass. In contrast to the single class, for $K = 2$ we distinguish two classes of requests, namely static and dynamic. We follow the same fitting process as above, except that we fit a model $\boldsymbol{PH}^a = \{\boldsymbol{D}_0^a, \boldsymbol{D}_1^a\}$ for the static IAT dataset and another model $\boldsymbol{PH}^b = \{\boldsymbol{D}_0^b, \boldsymbol{D}_1^b\}$ for its dynamic counterpart. In both cases, the inter-arrival times $V_j^{(i)}$ refer to the time between two static requests in the static model and between two dynamic requests in the dynamic model.

Intuitively, we have now to describe the aggregate flow of static and dynamic request types to the servers. In general, two options are available to this aim. In the first approach, one may consider multiclass QBD processes which have been studied only in recent years and are still poorly understood, thus requiring simulation for their evaluation [7]. In the second approach, one defines a new MAP which represents the superposition of the two flows treated as independent of each other. Classic theory shows that the superposition of two PH distributions is the MAP

$$\boldsymbol{MAP} = \boldsymbol{PH}^a \oplus \boldsymbol{PH}^b = \{\boldsymbol{D}_0^a \oplus \boldsymbol{D}_0^b, \boldsymbol{D}_1^a \oplus \boldsymbol{D}_1^b\}, \quad (14)$$

where $\oplus$ denotes the Kronecker sum operator. This operation describes the inter-arrival times between activation of observable transitions either in $\boldsymbol{PH}^a$ or in $\boldsymbol{PH}^b$. In other words, superposing the flows from the two types of workloads represents IAT of requests originating from two independent sources, namely $\boldsymbol{PH}^a$ and $\boldsymbol{PH}^b$ [9], and the result is not a PH model, but in general a MAP. This is because the superposition of i.i.d. arrival processes is not in general an independent flow of requests, but may show autocorrelation and burstiness that can only be captured by a MAP and cannot be represented with PH distributions. To the best of our knowledge, this superposition approach has never been applied to web server analysis. Thus, it provides an innovative way to parameterize performance models of web servers from real logfile traces.

B. Trace transformation

Computing inter-request times between adjacent requests whose timestamps are logged at one-second resolution, frequently results in the generation of sequences of zeros in the IATs. This is a very problematic effect to model in real traces because we have found that these sequences are highly autocorrelated, but we are not aware of Markov-modulated processes that can describe inter-batch autocorrelations. The Batch Markovian Arrival Processes (BMAPs), which are a generalization of MAPs, can describe batches of arrivals [14], however batch sizes are i.i.d. variables. Results of experiments presented in section VI-A demonstrate BMAPs do not give satisfactory results for modeling our web trace.

To cope with the difficulty, we have applied the methodology described in Section V-A to blocks $B^{(i)}$ after having removed temporarily the sequences of zero IAT values. The idea we have followed to account for the zeros in the performance model is to merge requests falling into the same second as a single logical request. The SVCTs for these requests are correspondingly merged. Thus, in this transformed model the arrival trace is exactly the trace without zeros that we have fitted in Section V-A and the effect of the zeros is seen only in the SVCT traces. The resulting SVCT trace is a transformed trace, which we name 'aggregated trace' in the remainder of this paper. To account for this transformation, it is sufficient to scale the mean queue length and throughput of the queueing results obtained using the aggregated trace (or its fitted MAP model). The scaling factor for $B^{(i)}$ is the ratio

$$R_f = \frac{len(B^{(i)})}{len(B^{(i)}) - zeros(B^{(i)})}, \quad (15)$$

where $zeros(B^{(i)})$ is the number of zero inter-arrival times in block $B^{(i)}$ before the filtering. We verified that the difference of mean queue length results between the original and aggregated trace re-scaled by (15) varies at most by 9%. This makes our transformation a good approximation to avoid the difficulty of modeling autocorrelated batches.

C. Service Process

We describe the procedure for deriving the service process from the aggregated trace, whose SVCTs exhibit ACF-1 values varying in the range $[.20, .41]$. To model the temporal dependence in the time series, we use a MAP model with 2 states. The script used is integrated in the KPC-Toolbox, a collection of MATLAB scripts to automatically fit traces using MAPs [9]. It takes as input our SVCT and derives a two-state MAP capturing the first three moments of the distribution and ACF-1.

Frequently, difficulties arise in the fitting process because two-state MAPs impose constraints on the moments and autocorrelation values that can be fitted by the model. In such cases, we split the distribution in equal parts and fit independently the corresponding datasets, and compose the individual models into a final MAP again via the superposition technique. In simple words, this means that we fit separate MAPs for small and large SVCTs and we then roughly approximate the model for the original trace by the superposition technique, yielding 4-state MAPs. According to the patterns observed, we have found that two-equally sized datasets, e.g. separated by the median, yield good results. This stems from the fact that the matching of the moments tends to fit better the tail than the body, resulting in an overestimation of the queue-length. Moreover, the situation is aggravated when fitting such short traces, often containing < 500 data points. In the absence of longer traces, splitting the trace has proven to be a good strategy.

VI. Experimental Results

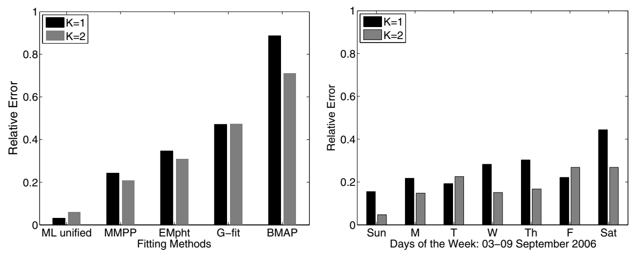
We represent the system under study as a queue processing an open workload, i.e., a workload coming from an external source independent of the state of the system. We consider only one service station that corresponds to one web server for system utilization $U = .90$, which represents a heavily loaded system. Heavy load prediction is more important than light-load prediction with the aim of sizing in the cloud. Moreover, it can be harder to estimate performance accurately in heavy-load. The methodology simulates the MAP/TRACE/1 queue and makes use of intermediate results to determine the final configuration of the model. At the end, we utilize a MAP/MAP/1 queue to model web server performance. On the grounds that the traces have been transformed, the number of observations in each of the blocks analyzed is often < 500. It is not only challenging to model such short traces, but also to obtain stable simulation results. We have succeed in tackling this problem by concatenating the same block of traffic to sum up to ten million elements. For each run, we log response time, and queue length. Thus, we evaluate estimation accuracy by the relative error

$$\Delta_\omega = \left| \frac{Q_{len}^{model} - Q_{len}^{trace/trace/1}}{Q_{len}^{trace/trace/1}} \right|, \qquad (16)$$

which is the relative error of the predicted mean queue length with respect to the trace mean queue length estimated by the TRACE/TRACE/1 simulation.

A. MAP/TRACE/1 results

We evaluate the accuracy of diverse state-of-the art approaches in constructing models for the IAT process. We consider a class of methods that fit model parameters to our web server data via the maximum likelihood estimation approach (EMpht, G-fit), and a formalism that derives the parameters analytically without numerical optimization. The methods in the first class both use the Expectation-Maximization (EM) algorithm for PH distributions. The algorithm iteratively estimates model parameters to measured data that maximizes the likelihood that the data has been sampled from the model [10]. We set the order of the MAP models to $J = 8$ states for $K = 1$ and for each class when $K = 2$, as we found it provides a good balance between tractability and accuracy. That is, for the multiclass case we obtain a model with $J = 8 * 8 = 64$ states. On the one hand, EMpht is an early development for which we have fitted a hyper-exponential distribution to values of web traces [3]. The rationale is that we note a variability of $c^2 > 13$ in the IAT datasets analyzed, whereas for the theoretical hyper-exponential is $c^2 > 1$. On the other hand, G-fit has been developed more recently and its algorithm is customized to fit the parameters of hyper-Erlang structures, which are mixtures of Erlang distributions [19]. In addition,



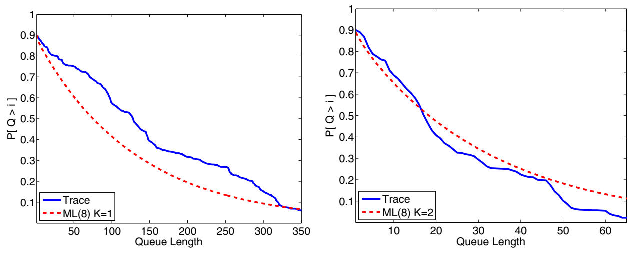
(a) MAP/TRACE/1 models (b) MAP/MAP/1 models

Figure 4. (a) Intermediate results; (b) Final results.

we use a MMPP model with $J = 2$ states, since it can be fit analytically from traces using simple methods [11]. The MMPP model is parameterized for the first three moments of the distribution and ACF-1. The methods used have been implemented in tools freely available. We compare these techniques against our method, called *ML unified*, that fits PH distributions with $J = 8$ states and possibly compose them in the multiclass case into a MAP. We compare the accuracy of the aforementioned approaches by fitting the inter-request times of web trace $B^{(4)}$. We integrate the derived processes into a MAP/TRACE/1 queue that is solved by simulation, under server utilization $U = .90$. We compute accuracy by (16), and show the relative error for all approaches in Figure 4(a). We clearly observe that ML unified delivers the best fit across all approaches with low prediction errors of 3% and 6% for the single class and the two class cases, respectively. The other approaches deliver the best prediction effort $> 20\%$ relative error. The 2-state MMPP model returns an error of approximately 21% for both classes. G-fit underestimates the trace, and attains a 47% of error for both classes. As apposed to G-fit, EMpht does slightly better by positioning between 31% and 35% error. Based on these intermediate results, we investigate a method capable of generating the sequences of zeros temporarily neglected that need to be reintroduced in the trace. Thus, we evaluate the accuracy of the Batch Markovian Arrival Processes (BMAPs), which are a generalization of MAPs that allow for batch arrivals [14]. That is, a MAP is a BMAP with all arrivals that consist of a batch of size 1. We then transform our ML unified models in order to account for batch arrivals of zeros, for which we use the probabilities of observing N number of consecutive zeros after a non zero IAT value. Finally, we simulate the BMAP/TRACE/1 queue and quantify accuracy by (16). Evidently, the BMAP largely overestimates the trace queue length [20], as its relative error emerges above the one of G-fit.

B. MAP/MAP/1 results

We show in Figure 5(a) the queue length tail distribution of model MAP/MAP/1 for block $B^{(3)}$, $K = 1$, and in Figure 5(b) the corresponding for block $B^{(4)}$, $K = 2$. Our predictions are a close match with the corresponding tail



(a) Web trace $B^{(3)}$ (b) Web trace $B^{(4)}$

Figure 5. Queue length tail distribution of the MAP/MAP/1 queue.

distributions obtained from simulation. We describe in detail the application of the methodology to $B^{(3)}$ as it has been challenging to model. We fit ML unified models of order $J = 8$, by inversely characterizing the IAT. The resulting model is re-scaled, if needed, to the mean of block $B^{(3)}$; smaller or larger mean values can be used to evaluate the system's response under increased or decreased utilizations, respectively. The SVCT of the aggregated trace is fitted by utilizing the KPC-Toolbox [9]. The resulting model, of order $J = 2$, captures closely not only the first three moments of the distribution, but also the autocorrelation with ACF-1= .32, whereas the real value is ACF-1= .38. In the end, we integrate both models and simulate the MAP/MAP/1 queue, whose resulting mean queue length is re-scaled to the original trace by (15). In the aggregated trace representation of $B^{(3)}$ a job queuing is equivalent to observe 4.76 jobs in the queue of the baseline experiment. We quantify accuracy by (16), and realize the difference between the simulation, and analytical mean queue length prediction is minimal at 0.95%. This discrepancy is explained due to simulation inaccuracies. A simulation run, including sampling ten million elements for each of the processes, takes on average five minutes. In contrast, analytical results are obtained immediately, often in a fraction of a second, given the MAP models and the exact system utilization parameter as input. The results for traces $B^{(3)}$ and $B^{(4)}$ are presented in the third and fourth groups of columns in Figure 4(b), respectively. For $B^{(3)}$ we observe a competitive estimation at 19% of error for K= 1, whereas in $B^{(4)}$ it attains 15% error for $K = 2$.

We notice that high variability present in the short traces poses a challenge for the single class case. However, estimation accuracy improves when moving to a multiclass workload. For $K = 2$, we note than on average prediction accuracy stays $< 18\%$ error, and in many cases even $< 16\%$ error. Actually, the striking prediction for trace $B^{(1)}$, $K = 2$ is at 4.6% error. Thus, it seems more practical to decompose the workload in an increased number of classes.

VII. Conclusions

We have provided an example that shows state-of-the-art fitting methods are often inaccurate in approximating queueing behavior of a trace. On the contrary, our ML unified method has demonstrated potential to achieve higher accuracy in predictive models for web workloads. We believe this study demonstrates the MAP/MAP/1 queue is a useful and versatile tool for performance prediction of web servers deployed in the cloud, which is significantly faster to solve analytically than by simulation. In the future, we intend to investigate the use of the proposed models for resource allocation and online web performance prediction.

References

[1] M. Andersson, J. Cao, M. Kihl, and C. Nyberg. Performance modeling of an apache web server with bursty arrival traffic. In *Proc. of ICOMP*, 2003.

[2] M. Armbrust, *et al*. A view of cloud computing. *Communications of the ACM*, 53(4):50–58, 2010.

[3] S. Asmussen and O. Nerman. Fitting phase-type distributions via the EM algorithm. *Scand. J. Stat.*, 23(4):419–441, 1996.

[4] F. Bause, P. Buchholz, and J. Kriege. A comparison of Markovian Arrival and ARMA/ARTA processes for the modeling of correlated input processes. In *Proc. of WSC*, 634–645, 2009.

[5] D. Bini, B. Meini, S. Steffé, and B. Van Houdt. Structured Markov chains solver: software tools. In *Proc. of SMCTOOLS Workshop*. ACM, 2006, http://win.ua.ac.be/~vanhoudt/.

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

[7] P. Buchholz, P. Kemper, and J. Kriege. Multi-class Markovian arrival processes and their parameter fitting. *PEVA*, 67:1092–1106, 2010.

[8] R. Calinescu, L. Grunske, M. Kwiatkowska, R. Mirandola, and G. Tamburrelli. Dynamic QoS management and optimisation in service-based systems. *IEEE TSE*, (99):1, 2010.

[9] G. Casale, E. Zhang, and E. Smirni. Kpc-Toolbox: Simple yet effective trace fitting using markovian arrival processes. In *Proc. of QEST, pp. 83–92*. IEEE, 2008.

[10] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc.*, 39(1):1–38, 1977.

[11] A. Heindl, K. Mitchell, and A. van de Liefvoort. Correlation bounds for second-order MAPs with application to queueing network decomposition. *PEVA*, 63(6):553–577, 2006.

[12] R. Horn and C. Johnson. *Topics in matrix analysis*. Cambridge University Press, 1994.

[13] D. Kwiatkowski, P. Phillips, and P. Schmidt. Testing the null hypothesis of stationarity against the alternative of a unit root. *J. of Econometrics*, 54(1-3):159–178, 1992.

[14] G. Latouche and V. Ramaswami. *Introduction to matrix analytic methods in stochastic modeling*. ASA-SIAM, 1999.

[15] M. Neuts. *Structured stochastic matrices of M/G/1 type and their applications*. Marcel Dekker, 1989.

[16] A. Riska and E. Smirni. MAMsolver: A matrix analytic methods tool. In *Proc. of TOOLS, pp. 205–211*. Springer-Verlag, 2002, http://www.cs.wm.edu/MAMSolver/.

[17] A. Riska, M. Squillante, S. Yu, Z. Liu, and L. Zhang. Matrix-analytic analysis of a MAP/PH/1 queue fitted to web server data. In *MAM4, pp. 335–356*, 2002.

[18] R. Singh, U. Sharma, E. Cecchet, and P. Shenoy. Autonomic mix-aware provisioning for non-stationary data center workloads. In *Proc. of ICAC, pp. 21–30*. ACM, 2010.

[19] A. Thummler, P. Buchholz, and M. Telek. A novel approach for fitting probability distributions to real trace data with the EM algorithm. In *Proc. of DSN, pp. 712–721*. IEEE, 2005.

[20] C. H. Xia, Z. Liu, M. S. Squillante, L. Zhang, and N. Malouch. Traffic modeling and performance analysis of commercial web sites. *SIGMETRICS PER*, 30:32–34, 2002.

[21] C. Xu, B. Liu, and J. Wei. Model predictive feedback control for QoS assurance in webservers. *IEEE Computer*, 41(3):66–72, 2008.

[22] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena. Intelligent workload factoring for a hybrid cloud computing model. In *Proc. SERVICES, pp. 701–708*. IEEE, 2009.

[23] Q. Zhang, A. Riska, W. Sun, E. Smirni, and G. Ciardo. Workload-aware load balancing for clustered web servers. *IEEE TPDS*, 16(3):219–233, 2005.