Probabilistic Modeling of Computing Demand for Service Level Agreement

Saahil Shenoy, *Student Member, IEEE*, Dimitry Gorinevsky, *Fellow, IEEE*, and Nikolay Laptev
Email: (saahils,gorin)@stanford.edu, nlaptev@ucla.edu

Abstract—Cloud computing applications must be allocated sufficient resources to comply with Service Level Agreements (SLAs). This paper considers data-driven probabilistic modeling of application resource demand for resource allocation. The modeling method is focused on peak demand and SLA violations and relies on a branch of statistics known as extreme value theory (EVT). Rigorous statistical validation of the proposed model shows that it generalizes better than the alternative. The paper presents a resource allocation algorithm using the model to ensure a given small SLA violation rate. For resource allocation in Yahoo data center, over 50% savings are demonstrated using the proposed approach.

1 INTRODUCTION

TODAY'S data centers improve hardware utilization by using virtualization. Hosting multiple Virtual Machines (VMs) on the same system allows for better resource management. The data center utilization is constrained by tenant requirements, such as Service Level Agreements (SLAs), which are considered in this paper. Violating the SLAs leads to penalties. (This is not the case if Service Level Objective (SLO) contract is used instead of the SLA).

Data centers employ multiple levels of capacity allocation and VM placement, e.g., see [1]. First, resource requirements are encoded by SLA constraints for each software application. Second, the *resource allocation* problem is solved to select the number and type of VMs that are allocated for the application. The last step is *placement of VMs* in the physical machines to make use of their capacity. This paper focuses at the resource allocation problem.

The work described in this paper was motivated by Yahoo's effort in consolidating multiple bare metal (BM) systems into VMs running in the cloud. The BM to VM migration required forecasting resources needed to meet the SLA requirements. Automating the resource allocation helps to avoid *over-provisioning*.

The contributions of this paper are as follows. First, the paper develops a data-driven approach to modeling of the resource demand for an application running on one or several VMs. The proposed model generalizes better than the alternatives. Second, the paper shows how the model can be used to determine VM resource allocation taking into account the SLA. Third, the modeling approach and the model-based resource allocation are applied to Yahoo data center data demonstrating over 50% savings.

- *This work was supported by Yahoo Faculty Research and Engagement Program (FREP) award.*
- *S. Shenoy is with Stanford University, Stanford, CA 94305, USA,*
- *D. Gorinevsky is with, Stanford University, Stanford, CA 94305, USA, and Mitek Analytics LLC, Palo Alto, CA 94306,* `dimitry@mitekan.com`
- *N. Laptev was with Yahoo Labs, he is now with Uber*

2 RELATED WORK

Most prior work on capacity allocation assumes that VMs require deterministic and fixed resources. Examples include optimization of VM allocation for data center energy consumption [2], [3], [4], [5].

The resource demand is highly variable; its randomly appearing peaks might violate the SLA. This creates the need for modeling and analysis of the SLA violation, see [6] for survey of work in this area.

Prior work on probabilistic models for resource allocation is limited. Load forecasting models for resource allocation are considered in [7], [8], [9]. More relevant are probabilistic models of demand distribution tail that describe the over-provisioning costs and SLA violation risk. Most earlier papers consider normal distribution models, see [10], [11], [12]. One exception is log-normal model in [13].

This paper considers a different model that is based on a branch of statistics known as extreme value theory (EVT). It accurately models the tail of the distribution that is involved in the SLA violations. The probability distribution is estimated from the historical utilization data and used to trade between the long-term SLA violations and over-provisioning.

3 Modeling

The proposed resource allocation approach is model-based. Its first step is to build probabilistic model of the resource demand. This section develops probabilistic modeling approach based on historical data.

3.1 Yahoo Dataset

The initial motivation for this research came from Yahoo data center applications. The presentation below is illustrated by the Yahoo data.

The data set used for algorithm development and illustrative examples in this paper included 355 computing applications that were migrated from bare metal hosts to the VMs by Yahoo in 2015. This work is focused on CPU data because other channels (e.g., memory, disk) had little impact on the SLA violations. This paper looks at historical CPU resource demand time series X_t for these Yahoo applications, where t is the sample number. The data was sampled at one hour interval and collected over several months. Each application (host), is supported by several servers with the total CPU capacity C. Machine types in the dataset range from 1-core to 64-core production servers and are used to serve both internal and external customers. The applications hosted on these machine include Yahoo! mail, messenger, and various properties (e.g., Sports, Finance). The operational profiles of the machines depended on the application and ranged from CPU-intensive to network-intensive workloads.

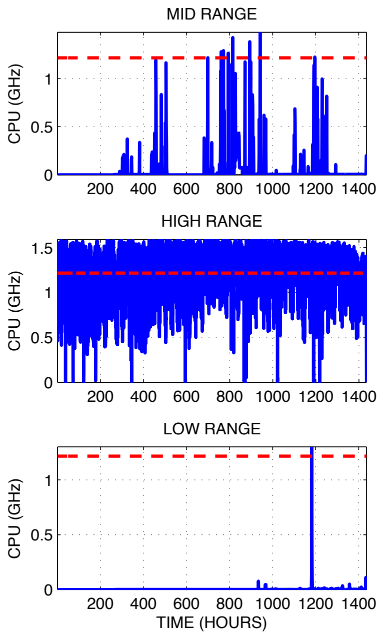


Fig. 1. Examples of three different dataset types.

The approach was informed by viewing the Yahoo CPU resource demand data. Three different types of encountered resource demand time series illustrated in Figure 1 require different features of the modeling approach. The top plot in Figure 1 shows what we call *mid range* data. The time series consists of several peaks and allows convenient modeling of the demand. The middle plot in Figure 1 shows *high range* data. This is data for severely under-provisioned application that has a large number of SLA violations. The demand exceeds the allocated capacity much of the time. This complicates demand modeling from the observed data. The *low range* data in Figure 1 bottom plot might have a few peaks, the rest of the data is near zero. This is a severely over-provisioned application. Such data might be insufficient for estimating a reasonable model for making allocation decisions.

The Yahoo data set includes 321 applications with low range data where the capacity is overallocated, 2 high-range applications where the capacity is under-allocated, and 32 mid-range applications.

3.2 Empirical Distribution Model

Consider a random variable X that models the resource demand for a single application. We denote the time series for historical demand for the application as

$$D = \{X_t\}_{t=1}^{T}, \tag{1}$$

where X_t is the demand data, t is the time sample index, and T is the total number of time samples.

Let $X_{(s)}$ be demand data (1) sorted in the ascending order of index s. Distribution of random variable X with realization X_t can be modeled by empirical survival function $P_{emp}(X \geq x)$ defined at nodes $X_{(s)}$,

$$P_{emp}\left(X \geq X_{(s)}\right) = \frac{T - s + 1}{T + 1}. \tag{2}$$

Between the nodes, $P_{emp}(X \geq x)$ can be interpolated.

Full-order empirical model (2) overfits the data. Such high-order model has low bias but high variance (poor generalization). The bias-variance trade off is well known in statistics. To reduce the variance, next section introduces a low order model.

3.3 Low Order Model

Consider variables y,

$$y = X - a, \tag{3}$$

where a is a given threshold. The positive values of y are known as peaks over threshold (POT) data in Extreme Value Theory (EVT), see [14]. The POT data can be used to describe the tail of the demand distribution. Modeling of the tail distribution is discussed below.

Given a threshold a, the POT data in (1) are

$$y_t = X_t - a, \tag{4}$$
$$Y = \{y_t | y_t > 0\}, \tag{5}$$
$$n = \text{card } Y, \tag{6}$$

where n is the number of POT data points in set Y.

In what follows, the positive values of y are assumed i.i.d.. A low order model assumes that positive POT exceedances (3) follow an exponential distribution, which is one of the distributions predicted by the EVT for the POT data, see [14]. As shown below, this distribution describes the Yahoo CPU demand data well. The number n of the positive exceedances y_t is binomially distributed and the survival function of the exponential distribution is exponential. This model can be expressed as

$$p(y|y > 0, \theta) = \theta \cdot \exp\left(-\theta \cdot y\right), \tag{7}$$
$$P(y > 0) = q, \tag{8}$$

where tail rate θ and quantile level q are the model parameters. Conditional probability formula yields

$$P(y > x) = P(y > a) \cdot P(y > x|y > a), \tag{9}$$

where the conditional dependence on parameters θ and q is omitted for brevity. For $x > a$, the probabilities in (9) can be expanded by using (4)–(8). This yields the probabilistic exponential model for the tail distribution of the raw demand variable X.

$$P(y > x) = q \cdot e^{-\theta(y-a)}. \tag{10}$$

Threshold parameter a in the exponential tail model (10) is assumed to be given and fixed. Model parameters θ and q can be then obtained from maximum likelihood estimate (MLE). Computing the log-likelihood of θ assumes that data (5) are independently distributed. The likelihood is then the product of the likelihoods for individual data points; the log-likelihood is the sum of the log-likelihoods. From (1), (4), (5), and (10) the log-likelihood of tail rate θ is

$$\begin{aligned} L_\theta &= \log p(Y|\theta) \\ &= \sum_{y_t>0} (\log\theta - \theta \cdot y_t) \\ &= n \log\theta - \theta \cdot \text{sum } Y. \end{aligned} \tag{11}$$

Log-likelihood of the quantile level parameter q is

$$\begin{aligned} L_q &= \log p(n|q) \\ &= \sum_{y_t>0} \log q + \sum_{y_t\le 0} \log(1-q) + C_q \\ &= n \log q + (T-n)\log(1-q) + C_q, \end{aligned} \tag{12}$$

where C_q is a constant and n is given by (6).

The MLE estimates of θ and q maximize the log-likelihoods (11) and (12) and are

$$\hat\theta = \arg\max_\theta L_\theta = (\text{mean } Y)^{-1}, \tag{13}$$
$$\hat q = \arg\max_q L_q = n/T, \tag{14}$$

where T comes from (1), n is given by (6), and Y is given by (5).

3.4 Censoring

Modeling procedure of Section 3.3 is suitable for the mid range data. High range data example in Figure 1 illustrates that model estimation must account for data censoring. The used resource is bounded by capacity C_T, even if the actual demand is higher, exceeds C_T.

MLE formulation (11) needs to be updated to address the censoring. For POT variable y (4), the capacity bound C_T is transformed into

$$c_T = C_T - a. \tag{15}$$

For censored data, equations (5) and (6) take the form

$$Y^{(u)} = \{y_t | 0 < y_t < c_T\}, \tag{16}$$
$$m = \text{card } Y^{(u)}. \tag{17}$$

Similar to (11), the log-likelihood for parameter θ is the sum of the log-likelihoods for n individual POT points in (6). The sum over m uncensored data points looks like (11) with m in place of n and $Y^{(u)}$ in place of Y. For the remaining $n-m$ of the censored data points, the underlying demand $y \ge c_T$ and the observed data is $y_t = c_T$. For each censored point, the likelihood $P(y > c_T|y > 0, \theta)$ can be computed by integrating the probability density function (7) from c_T to infinity. This yields the likelihood $\exp(-\theta \cdot c_T)$ and the log-likelihood $-\theta \cdot c_T$ for each of the $n-m$ censored points.

With the data censoring, the log-likelihood function (11) for parameter θ becomes $L_{\theta,C}$ of the form

$$\begin{aligned} L_{\theta,C} &= \log p\left(Y^{(u)}|\theta\right) + (n-m)\log P(y > c_T) \\ &= \left(m\log\theta - \theta \cdot \text{sum } Y^{(u)}\right) - (n-m)\cdot\theta\cdot c_T. \end{aligned} \tag{18}$$

The MLE estimate with the data censoring is obtained by maximizing $L_{\theta,C}$ (18) as

$$\hat\theta = m \cdot \left(\text{sum } Y^{(u)} + (n-m)\cdot c_T\right)^{-1}. \tag{19}$$

Formula (19) is still valid if there is no censored data. In that case, (16) yields $Y^{(u)} \equiv Y$, (17) yields $m = n$, and estimate (19) is exactly the same as (13).

The MLE estimate of q (14) does not change with the data censoring since the POT threshold a is always selected lower than the censoring threshold C_T.

3.5 MAP Estimation

The modeling approach must be able to deal with the low range data sets illustrated by the lower plot in Figure 1. Such data sets contain little demand data, sometimes all zero data. When the data is insufficient for reliable estimation, a standard Bayesian statistics approach is to include the priors for the estimated distribution parameters θ in (7) and q in (8).

It is convenient to use conjugate priors. For exponential distribution (7) with parameter θ, the conjugate is the Gamma distribution

$$p_{Gamma}\left(\theta|\alpha,\beta\right) = \frac{\beta^{\alpha}\theta^{\alpha-1}e^{-\beta\cdot\theta}}{\Gamma\left(\alpha\right)}, \tag{20}$$

where α, β are the distribution parameters, and $\Gamma(\cdot)$ is the standard Gamma function. A conjugate prior can be be described as using pseudo-observations in addition to the actual observations in the parameter estimation. In (20), parameter α has the meaning of the pseudo-observation number. Parameter β in (20) characterizes the tail rate that the pseudo-observations follow. Setting the mode of distribution (20) to θ_* gives

$$\theta_* = (\alpha-1)/\beta. \tag{21}$$

The prior mode θ_* gives the optimal Bayesian estimate if no data, except the prior, is available.

For the binomial distribution with parameter q (8), the conjugate is Beta distribution given by

$$p_{Beta}(q|\nu,\eta) = \frac{\Gamma(\nu+\eta)}{\Gamma(\nu)\cdot\Gamma(\eta)}\cdot q^{\eta-1}\cdot(1-q)^{\nu-1}, \tag{22}$$

where ν and η are the distribution parameters. Parameter ν has the meaning of the number of pseudo-observations that are below threshold a in (4) and have $y < 0$. Parameter η has the meaning of the number of pseudo-observations that have nonnegative POT exceedances, $y \geq 0$, and belong to the modeled distribution tail. The total number of the pseudo-observations is then $\nu + \eta$. Assume that q_* provides the prior knowledge of the tail quantile parameter q. Setting the mode of distribution (20) to q_* yields

$$q_* = (\eta-1)/(\eta+\nu-2). \tag{23}$$

The priors are incorporated in the estimation through Maximum A posteriori Probability (MAP) Bayesian formulation. The posteriors optimized by the proposed MAP formulation combine the priors (20)–(22) and the likelihoods used in the censored MLE in Section 3.4. The MAP estimation procedure works for all three types of data shown in Figure (1).

To estimate θ, log-posterior $L_{\theta,MAP}$ is obtained by adding log-likelihood $L_{\theta,C}$ (18) and log of prior (20). Keeping only the terms that depend on θ yields

$$L_{\theta,MAP} = L_{\theta,C} + (\alpha-1)\log\theta - \beta\cdot\theta. \tag{24}$$

MAP estimate of θ is computed from (18), (24) as

$$\hat{\theta}_{MAP} = \arg\max L_{\theta,MAP}, \tag{25}$$

$$\hat{\theta}_{MAP} = \frac{m+\alpha-1}{\text{sum}\, Y^{(u)} + (n-m)c_T + \beta}. \tag{26}$$

To estimate q, log-posterior $L_{q,MAP}$ is obtained by adding log-likelihood L_q (12) and log of the prior (22). Keeping only the terms that depend on q yields

$$\begin{aligned} L_{q,MAP} = {} & (n+\eta-1)\log q \\ & + (T-n-\nu-1)\log(1-q). \end{aligned} \tag{27}$$

MAP estimate of q is computed from (27) as

$$\hat{q}_{MAP} = \arg\max L_{q,MAP}, \tag{28}$$

$$\hat{q}_{MAP} = (n+\eta-1)\cdot(T+\eta+\nu-2)^{-1}. \tag{29}$$

In formulas (26), (29), data point number T comes from (1), n from (6), $Y^{(u)}$ from (16), m from (17), c_T from (15), α and β from (20), ν and η from (22).

3.6 Prior Tuning

Computing MAP estimates (26), (29) requires setting (tuning) four prior parameters α, β, ν, and η. The approach to doing so is discussed below.

Parameters α, β in (20) and ν, η in (22) are chosen to address the following engineering requirements.

R1: Priors shall dominate the estimate when the data set is smaller than a given minimum size. If there is more data, the results are less sensitive to the prior. (The examples below assume that the minimum data set size is one week worth of data).

R2: In the absence of actual data, the prior shall predict the distribution tail rate θ_* that is an average for a representative historical data set.

R3: In the absence of actual data, the prior shall predict a given SLA level for the existing capacity allocation. (In the examples below, the prior predicts that 85% of the capacity is violated 1% of time).

Engineering requirements **R1**–**R3** can be formulated in terms of prior parameter values. Let T_* be the number of samples in the minimal data set. Week worth of the historical data sampled at one hour interval yield $T_* = 7\cdot 24 = 168$. Requirement **R1** can be satisfied by setting the numbers of pseudo-observations in the priors (20) and (22) to T_*. In accordance with the discussion in Section 3.5,

$$\alpha = T_*, \tag{30}$$

$$\eta + \nu = T_*. \tag{31}$$

Let θ_* be the average tail rate parameter observed in historical data across a representative set of the applications. Requirement **R2** is fulfilled if α and β in (20) satisfy (21). Substituting (30) into (21) gives

$$\beta = (T_* - 1)/\theta_*. \tag{32}$$

Assume that the SLA specifies probability p_* for the demand exceeding the capacity level S_*. Satisfying **R3** requires that (10) holds for the prior model parameters, $x = S_*$ and $P(X > S_*) = p_*$. Assuming that $\theta = \theta_*$, see (21), and $q = q_*$, see (23) we get

$$(\eta - 1)/(\eta + \nu - 2) = p_* \cdot \exp(\theta_*(S_* - a)). \tag{33}$$

Solving (31) and (33) yields

$$\eta = 1 + (T_* - 2)p_* \cdot \exp(\theta_*(S_* - a)), \tag{34}$$
$$\nu = T_* - 1 - (T_* - 2)p_* \cdot \exp(\theta_*(S_* - a)). \tag{35}$$

Expressions (30), (32) provide the suggested tuning for parameters α, β in (20), while (34), (35) provide the tuning for ν, η in (22). Parameters T_*, p_*, S_*, and θ_* in these expressions have clear engineering meaning.

3.7 Model Validation

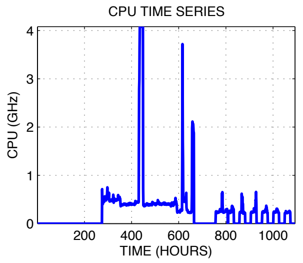
The proposed modeling approach was validated for data set (1) described in Section 3.1. The CPU utilization data were used; these are the percentages of the allocated CPU capacity for the application.

Tuning parameter in (23) was $q_* = 0.5$. The probability of exceeding S_* in (33) was $p_* = 0.01$ for the SLA level S_* at 85% of the capacity. The tail threshold a was set at 10% of the capacity. The selected threshold a was validated to provide low tail model error; this error is discussed later in this section. Tail rate parameter θ_* in (21) was determined from condition $P(X \geq S_*) = q_* \exp(-\theta_*(S_* - a)) = p_*$.

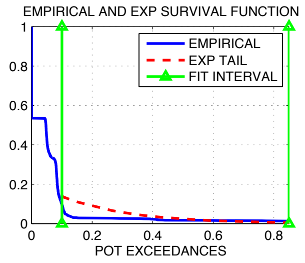
Figure 2 illustrates the modeling results for one selected application from the Yahoo dataset. Figure 2a shows $T = 1080$ samples of the CPU demand time series collected for the application over a period of 45 days. Figure 2b compares the survival function (10) for the MAP-estimated exponential model against the empirical survival function (2). The model fit is shown for the tail of the survival function. The vertical line shows the POT threshold a, where the tail starts. The estimated exponential model fits quite well for the tail. A part of the empirical survival function in Figure 2b below the POT threshold is a flat horizontal line. This is because about 20% of the data show zero CPU use.

The CPU demand in the Yahoo dataset was logged over 3 month period. For validation of the proposed modeling approach the dataset was divided into a training and test set. The model is estimated using the training set, which is the first 1.5 months of the data.

The errors of the exponential model fit were computed both for the training and test set for each of the 355 applications. For a given application, the error is the difference between the theoretical and empirical quantile. These errors are computed from the quantiles of the theoretical inverse CDF $F^{-1}(\cdot)$ and empirical inverse CDF $F^{-1}_{emp}(\cdot)$, respectively.



(a) CPU time series.



(b) Comparison of models.

Fig. 2. Model fit from MAP estimation on empirical data.

$$F(x) = 1 - P(X \geq x), \tag{36}$$
$$F_{emp}(x) = 1 - P_{emp}(X \geq x), \tag{37}$$

where $P(X \geq x)$ is the survival function in (10) and $P_{emp}(X \geq x)$ is the survival function in (2).

Positive errors imply that for a given quantile p, the exponential model predicts lower capacity used than the empirical CDF shows. Such under-estimation of the SLA violations is undesirable. The worst relative error of under-estimating capacity allocation is

$$\epsilon_{exp} = \sup_{p \in [q_*, Q]} \left(\frac{F^{-1}_{emp}(p) - F^{-1}(p)}{F^{-1}_{emp}(p)} \right), \tag{38}$$
$$F^{-1}(p) = a + \theta^{-1} \log(q/p), \tag{39}$$

where (39) follows from (10) and (38) is computed on the tail data interval $[q_*, Q]$. The quantile level $q*$ corresponds to the POT threshold (quantile) a; the results below are obtained for $Q = 0.99$. Function $F^{-1}_{emp}(p_t)$ can be computed from (2) by interpolating $X_{(t)}$ values on the grid of $F_{emp}(X_{(t)}) = 1 - P_{emp}(X \geq X_{(t)})$.

The empirical CDF $F_{emp}(\cdot)$ can be considered as a high-order model that is estimated along with the MAP-estimated exponential tail model (10) for the training set data. Once a model is estimated for the training set, the maximum absolute relative difference error of the model can be computed for the training set. The error of the exponential tail model (38) is given by (39). For the test set, $F_{emp}(\cdot)$ in (39) should be replaced by the empirical CDF $F_{emp,test}(\cdot)$ computed based on the test set data. The error of the empirical CDF $F_{emp}(\cdot)$ model for the test set data is $F^{-1}_{emp,test}(p) - F^{-1}_{emp}(p)$. The worst case relative error of under-estimating capacity allocation is then

$$\epsilon_{emp} = \sup_{p \in [q_*, Q]} \left(\frac{F^{-1}_{emp,test}(p) - F^{-1}_{emp}(p)}{F^{-1}_{emp,test}(p)} \right), \quad (40)$$

where interval $[q_*, Q]$ is the same as in (38).

Figures 3 and 4 show the maximum absolute relative difference errors ϵ_{exp} (38) and ϵ_{emp} (40) computed for a selected subset of applications in the Yahoo data set. The scatter plots show the error vs the mean CPU demand for each application. The results are shown for mid-range applications, as described in Section 3.1. The selected mid-range applications have tail rate parameter θ in the range of $0.1 < \theta^{-1} \cdot \log 10 < 0.5$.

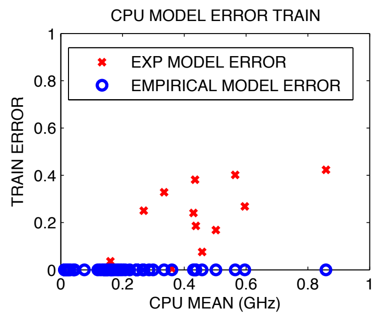


Fig. 3. Training set errors for mid-range applications.

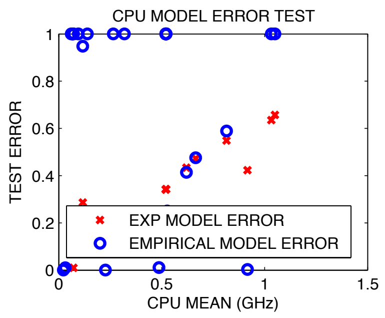


Fig. 4. Test set errors for mid-range applications.

Figure 3 shows the results for the test set data. The errors ϵ_{exp} (38) are marked by 'x'. The 'o' markers show the errors ϵ_{emp} (40) computed for the training set rather than test set; these errors are all zeros.

Figure 4 shows the results for the training set data. The exponential tail models errors ϵ_{exp} (38) computed for the test set are marked by 'x'. These errors are in the same range as the training set errors(38) in Figure 3: the exponential tail model generalizes well.

The 'o' markers in Figure 4 show the errors ϵ_{emp} (40) for the test set data. These errors are large. The results illustrate the bias/variance trade-off well known in statistics. The full-order empirical model has zero errors (variance) for the training set, see Figure 3. Yet, such model has large errors (bias) for the test set, see Figure 4. The exponential tail model has much smaller bias because it has just two parameters: θ (7) and q (8).

4 Decision Logic

A typical product launch cycle involves the following steps. First, a software application is developed. Next, the application is tested, benchmarked, and optimized on representative hardware. Finally, a request for hardware resources is made based on the predicted requests/queries per second (RPS/QPS). Often, such requests make allowances for expected growth and various buffers. As a result, most of the time, a good chunk of the hardware remains under-utilized. This is a fairly common problem in most data centers that have to deal with large *overprovisioning* cost. For example at Yahoo, a service engineer might periodically review the hardware utilization and re-balance the resources manually. The experience shows that this labor intensive approach requires improvement.

The modeling approach of Section 3 and the decision logic based on this model, which is described below, allow to automate the resource capacity allocation for each application. The capacity can be periodically reallocated as new demand data is accumulated and becomes available to the model estimation algorithm. This section presents describes the decision logic for resource capacity allocation and results of using it for the 355-application data set described in Section 3.1.

4.1 Optimal Capacity Allocation

For a given risk of SLA violation r, the data-driven model of the tail distribution described in Section 3 can be used to compute the minimum capacity allocation x_* needed to achieve this risk. The capacity required to maintain the SLA violation risk r is given by

$$x_* = F^{-1}(r), \quad (41)$$

For the exponential model, $F^{-1}(\cdot)$ is defined in (39) .

The optimal allocation will satisfy the constraint of not being below the VM capacity x_*. Assuming that the capacity scales linearly with the number of the allocated VMs, this number can be computed from an optimization problem for given SLA violation risk r

$$\begin{aligned} &\underset{n \in \mathcal{Z}}{\text{minimize}} \quad n \\ &\text{subject to} \quad n \cdot v \geq x_*, \end{aligned} \quad (42)$$

where v is the capacity of one VM. The optimizer solution to (42) is

$$n_* = \lceil x_*/v \rceil, \quad (43)$$

where $\lceil \cdot \rceil$ is the ceiling function, the smallest following integer. The formulation (42) allows different types of VMs having different capacities. An alternative version of (42) can be introduced to minimize the cost given the prices of different VM resources.

4.2 Estimated Savings

The modeling approach of Section 3 and decision logic of Subsection 4.1 was applied to Yahoo effort in migration of bare metal (BM) systems into virtual machines (VM). In the BM to VM migration, the model was used in decision logic suggesting what VM resources will be required for the application to meet the SLA requirements. The model was estimated from historical usage data for the application BM host.

The suggested allocation can be compared with the initial (current) allocation. The latter is described by current number n_c of the VMs that would be allocated based on the current host capacity,

$$n_c = \lceil C/v \rceil, \tag{44}$$

where C is the current capacity of the host.

Current allocation cost S and savings ΔS are

$$S = A \cdot n_c, \tag{45}$$

$$\Delta S = A \cdot \Delta n, \tag{46}$$

$$\Delta n = n_c - n_*, \tag{47}$$

where A is the unit cost of deploying one VM and n_* is the optimized capacity allocation (43). The cost savings come from reducing the capacity for the over-allocated hosts.

Figure 5 shows current VM allocation n_c (marked by 'x') and optimized allocation n_* ('o' marker) for $r = 5\%$. Figure 6 shows individual percentage savings for each dataset host in accordance with Figure 5.

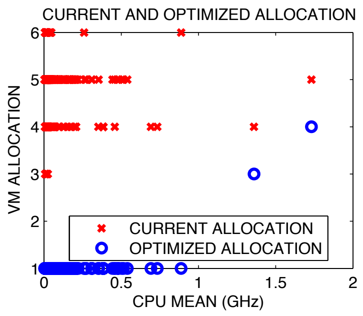


Fig. 5. Current and optimized allocation for each host at 5% SLA violation risk.

Table 1 summarizes the cost savings for all the hosts. We chose A =\$12 per VM and $v = 1$ GHz. The results are shown for three different SLA risk levels r. The saving are roughly \$16,000 over all the machines.

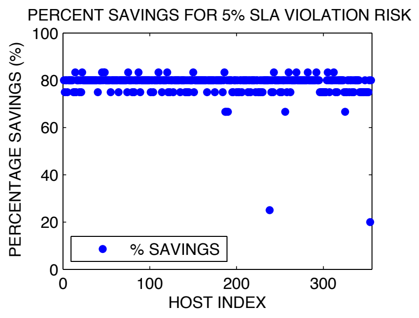


Fig. 6. Percentage savings for each host at 5% SLA violation risk.

TABLE 1
Cost Savings

r	sum Δn	(sum ΔS)/(sum S)
10%	1351	79.10%
5%	1350	79.04%
1%	1348	78.92%

4.3 Use of Exponential vs Empirical Model

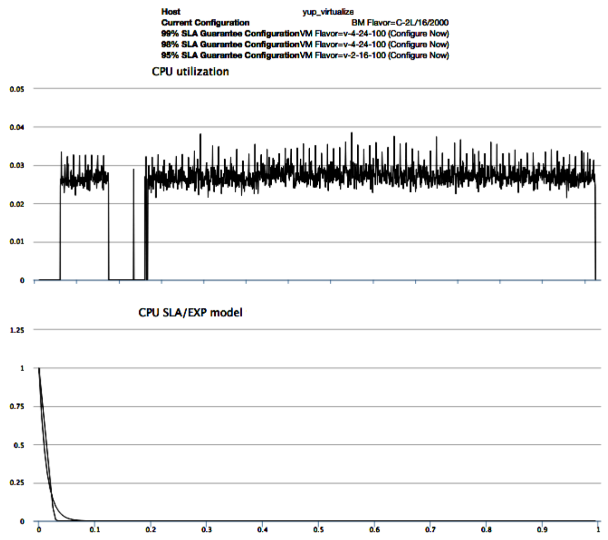
The capacity allocation logic (41), (43) is based on the low order exponential distribution model described in Section 3. The logic is based on the model CDF $F(x)$ (36). An alternative approach could be to use the full-order empirical distribution model described in Subsection 3.2. In that case, the CDF $F(x)$ in (41) is replaced with the empirical CDF $F_{emp}(x)$ (37).

The two capacity allocation algorithms - one based on the exponential, another based on the empirical distribution model - were compared using the framework similar to Subsection 3.7. The models were trained using the first 1.5 month of the data. Each allocation algorithm was then backtested using the second 1.5 month of the data for SLA risk $r = 0.01$. As described in Subsection 3.7, the empirical model always results in smaller capacity requirements. This difference is mostly lost because of the rounding in (43) and only one or two VMs are allocated in either case. The result is then the same for most of the 355 applications. However for five of the applications using the empirical model causes between 72% and 95% of SLA violations, and hence large penalties. For two of these applications, the exponential model correctly allocates three and four VM's respectively while the empirical model allocates only one VM, which is greatly inadequate.

5 Algorithm Deployment

The described data-driven probabilistic modeling and model-based optimal allocation approach were implemented as a software tool for initial demonstration and test of the concept. The software was used by Yahoo engineers during the BM to VM migration. The tool helped to optimize data center costs by converting historical resource utilization data for BM hardware to VM allocation requirements.

The developed software tool extracts time-series data from existing Yahoo monitoring tool for various resources, specifically the CPU load. It then converts the raw data to utilization time-series by doing scaling and data cleanup. The off-line part of the software fits the models to the time-series for each BM host and stores them on disk. The on-line part of the software runs as an Apache server application with a RESTful API. Given a host, current configuration, and required SLA, the software consults the model previously stored on disk and outputs a web page with suggested configuration.



Host	yup_virtualize
Current Configuration	BM Flavor=C-2L/16/2000
99% SLA Guarantee Configuration	VM Flavor=v-4-24-100 (Configure Now)
98% SLA Guarantee Configuration	VM Flavor=v-4-24-100 (Configure Now)
95% SLA Guarantee Configuration	VM Flavor=v-2-16-100 (Configure Now)

Fig. 7. The UI of the tool showing the three recommended VM configurations for three SLA levels. The bottom figure shows both the SLA and the EXP model fit.

The output web page of the deployed on-line software tool is shown in Figure 7. The page includes the plots of the historical demand time series for the selected host and two survival functions computed for the time series: the empirical survival function (2) and the exponential survival function (10).

The parameters for the exponential model are estimated in accordance with (26) and (29) as discussed in Section 3.5. The exponential model is then used to compute the optimized VM allocation as discussed in Section 4.1. The computed VM allocation is shown as suggestion at the page that provides user interface for selecting the VM allocation.

Because of how the tool was used in the migration process, the savings were not documented. Subsection 4.2 provides an estimate of the achieved savings.

6 Conclusions

This paper presents an automated and robust approach to probabilistic modeling of computing resource allocation based on historical data for an application. The probability distribution model is estimated from historical utilization data. The model can be used to trade between the SLA violations and over-provisioning and enables substantial savings.

References

[1] A. Beloglazov, R. Buyya, Y. C. Lee, A. Zomaya, *et al.*, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," *Advances in computers*, vol. 82, no. 2, pp. 47–111, 2011.

[2] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proc. 2010 IEEE/ACM Int. Conf. on cluster, cloud and grid computing*, pp. 826–831, IEEE Computer Society, 2010.

[3] M. Cardosa, M. R. Korupolu, and A. Singh, "Shares and utilities based power consolidation in virtualized server environments," in *Integrated Network Management, 2009. IM'09. IFIP/IEEE Int. Symp. on*, pp. 327–334, IEEE, 2009.

[4] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future generation computer systems*, vol. 28, no. 5, pp. 755–768, 2012.

[5] D. Minarolli and B. Freisleben, "Utility-based resource allocation for virtual machines in cloud computing," in *2011 IEEE Symp. on Computers and Communications (ISCC)*, pp. 410–417, IEEE, 2011.

[6] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez, and W. Wang, "Quality-of-service in cloud computing: modeling techniques and their applications," *Journal of Internet Services and Applications*, vol. 5, no. 1, pp. 1–17, 2014.

[7] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *2011 IEEE Int. Conf. on Cloud Computing (CLOUD)*, pp. 500–507, IEEE, 2011.

[8] S. K. Garg, A. N. Toosi, S. K. Gopalaiyengar, and R. Buyya, "SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter," *Journal of Network and Computer Applications*, vol. 45, pp. 108–120, 2014.

[9] R. N. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using arima model and its impact on cloud applications QoS," *IEEE Tran. on Cloud Computing*, vol. 3, no. 4, pp. 449–458, 2015.

[10] D. Niu, H. Xu, B. Li, and S. Zhao, "Quality-assured cloud bandwidth auto-scaling for video-on-demand applications," in *Proc. IEEE INFOCOM, 2012*, pp. 460–468, IEEE, 2012.

[11] H. Jin, D. Pan, J. Xu, and N. Pissinou, "Efficient VM placement with multiple deterministic and stochastic resources in data centers," in *Global Communications Conf. (GLOBECOM), 2012 IEEE*, pp. 2505–2510, IEEE, 2012.

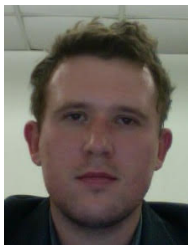
[12] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *Proc. IEEE INFOCOM 2011*, pp. 71–75, IEEE, 2011.

[13] A. Alasaad, K. Shafiee, H. M. Behairy, and V. Leung, "Innovative schemes for resource allocation in the cloud for media streaming applications," *IEEE Tran. on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1021–1033, 2015.

[14] L. de Haan and A. Ferreira, *Extreme Value Theory: An Introduction*. New York: Springer, 2006.

Saahil Shenoy (SM14) is a graduated Ph.D student in Physics and an ARCS Foundation Scholar at Stanford University. He completed the MS requirements in Particle Physics at the age of 20. His Stanford research is in big data analytics for extreme events, with applications in energy, finance, climate change, cloud computing, and risk analysis. He worked on portfolio optimization with Charles Schwab. He was Best Student Paper Award Finalist at 2015 American Control Conference and authored several papers in IEEE journals and conferences.

Dimitry Gorinevsky (M'91–SM'98–F'06) received a Ph.D. from Moscow (Lomonosov) University, and a M.Sc. from Moscow Institute of Physics and Technology (Phystech). He has been a Consulting Professor in Electrical Engineering with Information Systems Laboratory at Stanford University since 2003. His interests are in analytics applications for Industrial Internet of Things (IIoT). He is a founder of Mitek Analytics, an IIoT company in Palo Alto, CA. Over the last two decades, he has been working on data analytics applications in aerospace systems, energy, computing, and other industries. He has authored a book, 170+ papers, and many patents. He received Control Systems Technology Award, 2002, and Transactions on Control Systems Technology Outstanding Paper Award, 2004, of the IEEE Control Systems Society. He received Best Paper Award (Senior Award), 2013, of the IEEE Signal Processing Society. He is a Fellow of IEEE.



Nikolay Laptev completed his PhD in Computer Science at UCLA. The focus of his research is on Big Data machine learning and system design. Besides Big Data machine learning, during his PhD years Nikolay also conducted research on approximation approaches with an error guarantee for general machine learning algorithms. At Yahoo Labs his research focused on ranking, forecasting and anomaly detection. Currently at Uber, he conducts research on system design and machine learning over massive data-streams and stored data.