Spatial-Temporal Prediction Models for Active Ticket Managing in Data Centers*

Ji Xue, Robert Birke, Lydia Y. Chen, and Evgenia Smirni

***Abstract*—Performance ticket handling is an expensive operation in data centers, where physical boxes host multiple virtual machines (VMs). A large body of tickets arise from the resource usage warnings, e.g., CPU and RAM usages that exceed predefined thresholds. The transient nature of CPU and RAM usage as well as their strong correlation across time among co-located VMs within boxes drastically increase the complexity of ticket management. Based on a large resource usage data collected from production data centers, with 6K physical boxes and more than 80K VMs, we first discover patterns of spatial and temporal dependencies among/within the usage series of co-located resources. Leveraging our key findings, we develop an Active Ticket Managing (ATM) system that consists of (i) a spatial-temporal dependency based time series prediction methodology and (ii) a proactive capacity planning policy for CPU and RAM resources for VMs co-located within a box and boxes within a single data center client, that aims to drastically reduce usage tickets. ATM exploits the spatial-temporal dependency across/within multiple resources of co-located VMs and single-client boxes for usage prediction, and then actuates proactive capacity planning. Evaluation results on traces of 6K physical boxes from operating data centers show that ATM is able to provide accurate prediction of usage series in cloud data centers with low computational overhead, and at the same time ATM achieves significant ticket reduction up to 60% for both VM and box usage series.**

***Index Terms*—Cloud Data Center, Reliability Analysis, Spatial-Temporal Prediction, Capacity Planning**

I. Introduction

PERFORMANCE ticketing provides the means to data centers to interactively improve user experience by maintaining performance at tails. Typically, tickets can be issued by users or by system monitoring tools when performance violations are encountered, e.g., unresponsive service, high resource usage due to transient load dynamics, or persistent insufficient provisioning. Ticket resolution is unfortunately very expensive [2], [3] as a significant amount of manual labor is required for root-cause analysis and to remedy the detected problem [4]. Prior work has shown that there is strong correlation of ticket issuing with resource usage exceeding certain predefined thresholds [5]. In today's data centers, with physical resources being aggressively multiplexed across multiple virtual machines (VMs), the likelihood of issuing performance tickets due to physical or virtual machines crossing predefined usage thresholds dramatically increases.

Past work has established that resource usage at data centers exhibits strong temporal patterns [6], [7]. Beyond temporal dependencies that are established by usage time series [8], it is common for co-located VMs to simultaneously compete for limited physical resources. We illustrate a motivating example in Figure 1 depicting the CPU usage time series[1] of 4 VMs co-located within the same physical box, where performance tickets are issued automatically when a VM utilization exceeds a threshold of 60%. One can easily see the *spatial* dependency of VM1 and VM3, i.e., usage series move up and down synchronously. So do VM2 and VM4. The above time series are obtained by an IBM data center production system and are quite representative of typical patterns encountered in such systems.

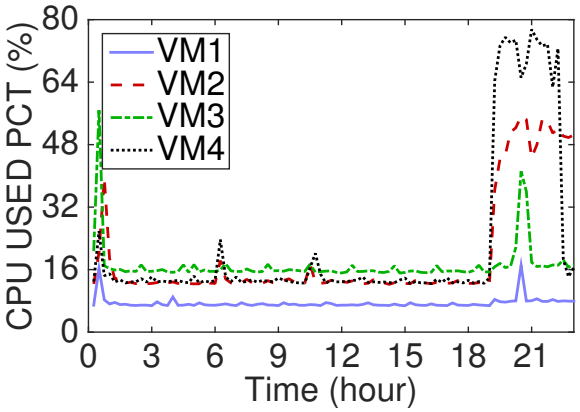


Fig. 1: An illustration of spatial dependency across usage time series for 4 VMs co-located on a box.

The focus of this paper is to develop a methodology that harnesses spatial and temporal dependencies of usage time series and increases the data center dependability by using a *proactive* approach: reduce the number of tickets by predicting *when* they will occur in the future and by employing dynamic capacity planning to adjust resource usage to avoid triggering of future tickets. To this end, we first conduct a detailed, post-hoc workload characterization study of usage time series in IBM production data centers corresponding to 80K VMs hosted on 6K physical servers. We develop an Active Ticket Managing (ATM) system that predicts future VM and box resource usage, and then proactively adjusts capacity on the co-located VMs and single-client boxes. The research challenges are numerous and outlined as follows.

Effective usage prediction is a prerequisite to the development of any management policy. Indeed, in our past work we have shown that neural networks can be effectively employed for prediction [8], but their effective usage remains prohibitively expensive in practical situations as it suffers by high

* A preliminary revision of this paper appeared at DSN'2016 [1].

[1]We interchangeably use the terms time series and series.

training costs. In practice, in a large-scaled data center, with more than tens of thousands of physical boxes and hundreds of thousands of VMs, it is infeasible to rely on neural networks to predict future resource usage. We solve this first problem by developing a prediction methodology that discovers spatial dependencies across usage series and exploits them to develop an agile methodology for prediction. To this end, we introduce the concept of *signature series*, a subset of usage series that are representative of all other usage series. We are able to predict usage series not in the signatures set and the usage violation tickets of co-located VMs, via a linear combination of signature VM series, which provides a time series prediction model with as low as 15% of the original time series. More significant series reduction (i.e., as low as 10%) is observed when predicting box usage series within each single client. Second, based on predicted resource usage, we define a multi-choice knapsack problem and develop a greedy algorithm to dynamically adjust (resize) virtual resources across co-located VMs, or to perform efficient capacity planning of physical resources across boxes within the same client. ATM is evaluated on production traces of 6K boxes and 80K VMs. Our extensive evaluation results show that ATM has remarkably high accuracy in prediction, i.e., its relative prediction errors are as low as 20% for VM usage series and only 7% for box usage series while using only 20% of the original time series. Usage prediction results in significant ticket reductions, i.e., up to $60\% - 70\%$ fewer tickets. The contributions of this paper are as follows:

1) We do post-hoc characterization of usage ticket issuing in a commercial data center setting. We focus on discovering the distribution of usage tickets and spatial-temporal patterns of resource usages across/within co-located VMs and single-client boxes. We find that usage tickets are mostly due to a small set of VMs or boxes, and that both co-located VMs and single-client boxes show significant auto- and cross-correlation among their CPU and RAM usage series.

2) Motivated by the strong spatial patterns across resource usage series of co-located VMs and of single-client boxes, we argue that a small number of signature usage time series can be used as predictors to represent well the entire set of resource usage time series. Moreover, we also provide a customized interface for users to determine the trade-off between prediction accuracy and prediction cost. This prediction methodology is the basis of ATM.

3) We develop a capacity planning policy to reduce usage tickets by setting the upper limits of CPU and RAM allocations when several VMs are co-located or when boxes belong to the same client, a problem which is shown to be NP-hard. We rigorously formulate the ticket minimization problem subject to the physical capacity constraints. We propose a greedy algorithm to solve it and compare its performance to the max-min fairness algorithm.

The outline of this work is as follows. Section II provides a characterization study on the usage tickets as well as the spatial patterns among usage series of co-located VMs and single-client boxes. We propose spatial-temporal prediction methods for resource usage series in Section III. In Section IV, we formulate the ticket minimization problem and demonstrate a greedy resizing algorithm to reduce usage tickets. An extensive evaluation of ATM on both production traces is discussed in Section V. Section VI presents related work, followed by summary and conclusions in Section VII.

II. Statistics and Observations

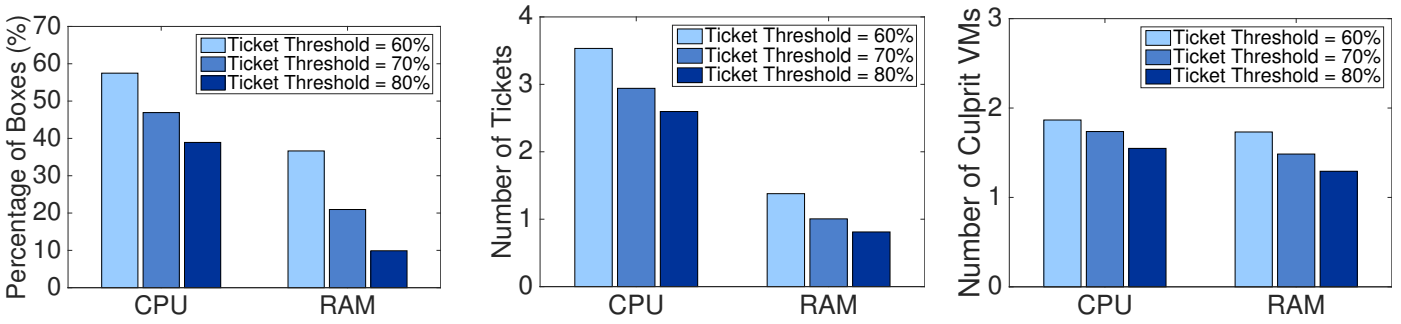
The motivation for the design of ATM is the need to reduce usage tickets that are typically issued when resource utilizations exceed certain thresholds. The trace that we consider here comes from IBM production data centers serving various industries, including banking, pharmaceutical, IT, consulting, and retail. The majority of VMs in the trace are VMware VMs. The trace contains CPU and RAM capacity but also utilization data taken at a time granularity of 15 minutes for 6K physical boxes within 300 clients, hosting more than 80K VMs, during a 7-day period from April 3, 2015 to April 9, 2015. In a private cloud data center, which is a "single client environment", the hardware, storage and network are dedicated to a single company. A client is assigned a unique cluster of boxes, which allows customized placement of VMs and assignment of resources. Within this environment, there is no sharing of boxes across different clients. On average, 10 VMs are consolidated within a single physical box, and 15 physical boxes are assigned to each client [6]. Both VMs and boxes are very heterogeneous in terms of their resource configuration.

In the following, we first show the distribution of usage tickets under different ticket thresholds for VMs and physical boxes, followed by a more detailed analysis on the spatial and temporal patterns of usage series of both co-located VMs and single-client boxes. We aim to uncover how usage tickets are distributed across resources and most importantly how usage patterns trigger usage tickets. We anticipate that the design principles of the proposed ATM system leverage this characterization analysis.

A. Usage Tickets

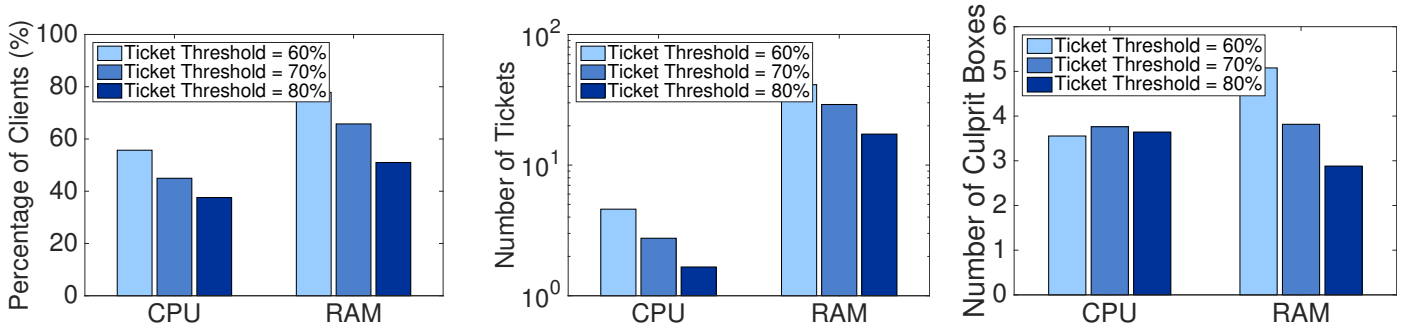
Usage tickets are generated when utilization values exceed target thresholds. Naturally, lower thresholds trigger a higher number of usage tickets and increase the cost of resolution, whereas higher thresholds result into fewer tickets but at a higher risk of performance degradation. To quantify the effect of different thresholds, we consider three threshold levels, namely 60%, 70%, and 80%. Such values are commonly adopted in production systems [9]. Figures 2-3 illustrate quantitative information on the issued tickets for the CPU and RAM usage series of April 3, 2015, in terms of VMs and boxes respectively. We illustrate how many boxes have VM tickets and how these tickets are distributed across co-located VMs and their resources. The similar characterization on box usage tickets is also presented.

1) VMs: First, we provide an overview on the VM usage tickets within each physical box for CPU and RAM. Figure 2(a) plots the percentage of boxes that have at least one VM usage ticket under the selected different thresholds. Even with the highest ticket threshold of 80%, almost 40% of boxes obtain at least one ticket due to CPU violation and 10% due to RAM violation, these percentages increase to 57% and 38%,



(a) Percentage of boxes having VM tickets (b) Average number of tickets per VM (c) Average number of culprit VMs

Fig. 2: Characterization of usage tickets for CPU and RAM of VMs per box.



(a) Percentage of clients having box tickets (b) Average number of tickets per box (c) Average number of culprit boxes

Fig. 3: Characterization of usage tickets for CPU and RAM of boxes per client. Notice that in (b), the y-axis is in log-scale.

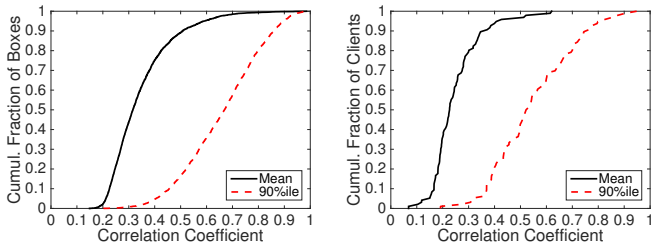
respectively, when the threshold is 60%. Figure 2(b) illustrates the average number of VM tickets for CPU and RAM. The average number of CPU usage tickets per VM are 4, 3, 2.5, for the three thresholds of 60%, 70%, and 80%, respectively, showing a relatively minor decreasing trend. A similar trend is observed for RAM usage tickets in VMs. Overall, RAM tickets are less likely to occur and in lower amount compared to CPU tickets in VMs, independently of the threshold. This can be explained by the fact that RAM tends to be over-provisioned for performance reasons. The next natural question is whether tickets are evenly distributed across all co-located VMs. To this end, we compute the number of VMs that accounts for the majority of tickets, where the majority is defined to 80% of usage tickets per box (this is an ad-hoc value). Figure 2(c) shows that on average one to two VMs per box cause the majority of tickets, irrespective of the three threshold values. A further interesting observation is that since the culprit VMs are few, if we increase the capacity allocation of the culprit VMs by removing resources from other co-located VMs, then tickets may reduce. On the contrary, if tickets are evenly distributed, such resizing does not help.

2) Boxes: We provide statistics on the box usage tickets across all clients for CPU and RAM, see Figure 3. Figure 3(a) exhibits similar trends as VM usage tickets, compared with Figure 2(a). The trend here on the severity of RAM vs CPU tickets is reversed, i.e., a significant percentage of clients experience RAM tickets. Figure 3(b) measures the average number of box usage tickets for CPU and RAM. The average number of CPU(RAM) usage tickets per box are 5(40), 3(30), 2(18) for the three thresholds, and this again suggests that RAM is more over-utilized than CPU in physical boxes. The above observations suggest the need to reduce/avoid the box usage tickets, especially for RAM. Finally, we aim to quantify the culprit boxes, which result in the majority (i.e., more than 80%) of box usage tickets within each client, see Figure 3(c). Across CPU and RAM, we observe that no more than 5 physical boxes in each client cause more than 80% of box usage tickets, recall that on average each client is assigned 15 boxes. This is similar as what we observe for VM usage tickets, and again suggests the opportunity to reduce/avoid the box usage tickets by efficient capacity planning of physical resources among the single-client boxes.

After the overview of usage tickets, in the following, we characterize the spatial and temporal dependency of usage tickets within co-located VMs and single-client boxes.

B. Do Spatial Dependencies Exist?

To better understand the spatial patterns of usage tickets, we estimate the magnitude of spatial dependency by computing the Pearson's correlation coefficients [10] over each pair of CPU and RAM usage series of co-located VMs within each box as well as boxes within each client. For co-located VMs, we compute the mean and 90%ile of the above measures and present the cumulative distribution functions (CDFs) across all the boxes in Figure 4(a). A similar analysis of spatial dependency is also done for box usage series across all the clients, see Figure 4(b).



(a) VM (b) Box

Fig. 4: Cumulative distribution of correlation within co-located (a) VMs and (b) single-client boxes.

Comparing Figures 4(a) and 4(b), one can immediately see from the shapes of the CDFs that co-located VMs have slightly stronger correlations (spatial dependency) than physical boxes within the same client. Indeed, the mean values for correlation measures (mean and 90%ile) of resource usage series of co-located VMs are 0.35 and 0.66, while the mean values for single-client boxes are 0.25 and 0.55. Still, the observations give a clear message: across both co-located VMs and single-client boxes, resource usage series exhibit strong spatial dependency. This is a fact that we take advantage of when we attempt to use clustering to reduce the cost of prediction.

C. Do Temporal Dependencies Exist?

Besides of spatial dependencies, another interesting aspect to explore is temporal dependency within each usage series, which is key to prediction. Other work [11] combines fast fourier transforms and autocorrelation to capture temporal patterns. Similarly to [11], we leverage autocorrelation to capture temporal dependencies within usage series. To motivate our discovery of temporal dependency, we first present two representative CPU usage series throughout 2 months for a certain VM and box in Figures 5(a) and 5(b), respectively. We observe that the workloads within the IBM private cloud exhibit clear periodic patterns over time. To capture the temporal dependency within each usage series, we show the autocorrelation of each usage series for the selected VM and box, see Figures 5(c) and 5(d). Autocorrelation is a mathematical representation of the degree of similarity in a time series and a lagged version of itself. As such, it is ideal for discovering repeating patterns by quantifying the relationship between different points of a time series as a function of the time lag [12]. The autocorrelation metric is in the range of $[-1, 1]$. Higher positive values indicate that the two points between the computed lag distance are "similar", i.e., have stronger temporal dependency. Zero values suggest no temporal dependency. Negative values show that the two points lag elements apart are diametrically different. In Figures 5(c) and 5(d), it is clear that the autocorrelation becomes high at certain lag values and that these lag values can be different for different usage series. Again, this observation verifies the strong temporal dependency within the selected usage series. Notice that similar periodic patterns are also observed across most of usage series in IBM private cloud data centers [8].

Although autocorrelation is effective to quantify the temporal dependency in most of the cases, autocorrelation alone cannot always represent temporal dependency, if for example, the usage series is constant overtime. In this case, autocorrelation cannot be computed. Yet, this kind of series still exhibits very strong temporal dependency, thus it can be predicted easily. To this end, we use the coefficient of variation (C.V.) of series. Figure 6(a) presents the CDFs of C.V. of VMs and boxes for CPU and RAM usage series. We observe that RAM usage series of boxes have the highest line, with mean C.V. equal to 0.05, referring to the high consistency of box RAM usage series overtime and unnecessary use of autocorrelation to measure its temporal dependency. For usage series of VMs and CPU usage series of boxes, where the C.V. is much higher, there it is possible to leverage autocorrelation to measure their temporal dependencies.

With the exception of RAM usage series of boxes, we quantify temporal dependency of usage series in cloud data centers, using a metric called goodness of temporal dependency (GTD):

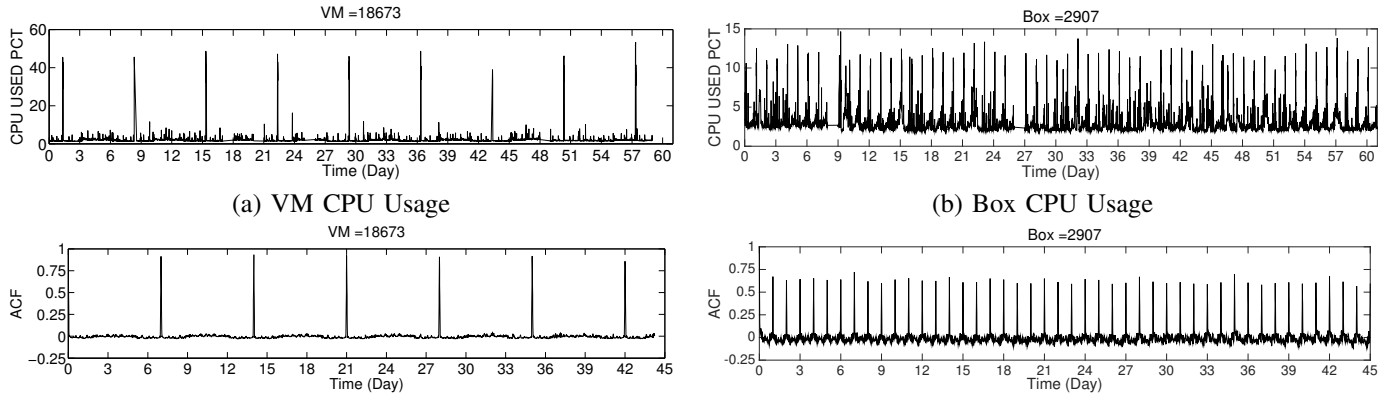
$$GTD = \beta * max(ACF_{short}) + (1 - \beta) * max(ACF_{long}). \quad (1)$$

Here ACF_{short} is a list of autocorrelation coefficients for lags that correspond to time stretches of less than 1 day, while ACF_{long} consists of autocorrelation coefficients for lags that are more than 1 day. We capture the highest autocorrelations in both short- and long-term, i.e., $max(ACF_{short})$ and $max(ACF_{long})$ respectively. The term $\beta \in [0, 1]$, is a weight that represents how important the short-term behavior is for time series prediction, which is inversely related to the prediction length. In our application scenario, the favored prediction length is at least 1 day ahead. In our experiments, β is set as 0.2 to satisfy such long-term prediction demand. The higher the GDT, the stronger temporal dependency within the series. In Figure 6(b), the CDFs of GTD of VM usage series for CPU and RAM, as well as box usage series for CPU are presented. In general, the graph shows that box CPU usage series have the lowest line, which indicates the overall highest GTD values, followed by VM RAM and then VM CPU usage series.

To summary Sections II-B and II-C, we discover strong spatial and temporal dependencies of resource usage series for both VMs and boxes. With the above observations, in the next section, we propose spatial-temporal prediction models to accurately and cheaply predict the usage series in cloud data centers.

III. Spatial-Temporal Prediction Models

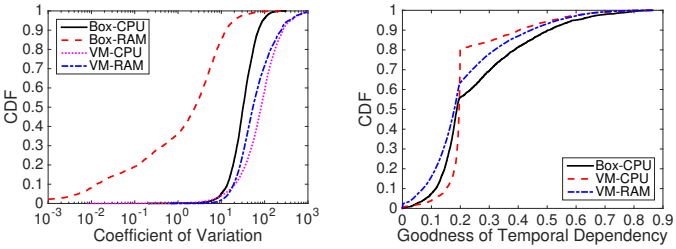
We first elaborate on the challenges for concurrent prediction for a large number of time series representing multiple resource usages from co-located VMs and boxes at production data centers. The immediate obstacles of prediction given a large number of demand series are accuracy, training overhead, and model scalability. Typically, temporal models [10], such as ARIMA are not able to capture bursty behavior well. More sophisticated temporal models such as neural networks, capture irregular patterns better but at much higher computational



(a) VM CPU Usage (b) Box CPU Usage

(c) Autocorrelation function of VM CPU Usage (d) Autocorrelation function of box CPU Usage

Fig. 5: CPU utilization over time for one representative (a) VM and (b) box, with their autocorrelation functions presented in (c) and (d), respectively.



(a) C.V. (b) GTD

Fig. 6: Comparison of temporal characteristics between VM and box usage series. Note that x-axis in (a) is in log-scale.

overheads. Given such restrictions, it is important to come up with efficient and accurate prediction models that also scale well.

We propose a new prediction methodology that combines both temporal and spatial models to predict the resource demand time series[2] $\boldsymbol{D_i}$ $(\forall i \in [1, M \times N])$ where M is the number of co-located VMs(boxes) and N is the number of different resources taken into consideration. We introduce the concept of *signature series:* a minimum number of time series that are predicted via *temporal models*. The rest of the demand series, termed as *dependent series*, are predicted through a linear combination of signature series via *spatial models*. Essentially, we divide the demand series, $\boldsymbol{D_i}$, into two sets: the signature set, denoted by Ω_s, and the dependent set, Ω_d.

The novelty of ATM is to derive novel spatial models for dependent series while applying existing temporal models to predict signature series. Many practical techniques exist in the literature for reducing the overhead of temporal models by extracting and storing features of the time series [8], [13]. We stress that any temporal prediction model can be directly plugged into the ATM framework.

To derive the spatial models, we want to express all demand series $\boldsymbol{D_k}, k \in \Omega_d$ by a linear combination f_k of the signature series $\boldsymbol{D_j}, j \in \Omega_s$:

$$\boldsymbol{D_k} = f_k(\boldsymbol{D_j}). \tag{2}$$

As every demand series can be either a signature or a dependent series, a brute force solution to find the minimum signature set is to explore all $2^{N \times M}$ combinations of regression models. For boxes hosting an average number of M VMs, i.e., $M \approx 10$ and expected to grow as servers become more powerful, it is clear that this method is not viable. To address this issue, we devise an efficient searching algorithm that can quickly find signature series without exhaustive search, by leveraging time series clustering techniques and signature selection methods.

A. Searching for Signature Demand Series

Key to the discovery of signature series is clustering. We propose a three-step algorithm to identify the signature set Ω_s. *Step 1* defines the initial clusters across all usage series. This is achieved using time series clustering, specifically dynamic time warping (DTW) [14] or correlation based clustering (CBC) that we propose here. *Step 2* chooses the initial set of signature series via either *spatial-* or *temporal-driven* signature selection methods. *Step 3* defines the final set of signature series by detecting and removing multicollinearity among the initial set of signature series using variance inflation factors (VIF) and stepwise regression. The intent of the last step is to fix the pitfall that although signature series appear independent, it is possible that a combination of certain subsets of the initial signature series can well represent the others. For example, a group of series can be separated into three clusters because of their dissimilarity in the distances or the correlation patterns. If however one of the clusters can actually be well expressed as a linear combination of the other two, then this falls under a classical example of multicollinearity. Figure 7 illustrates the steps of signature set search.

Step 1 - Time Series Clustering: Dynamic time warping is an effective solution for finding clusters of time series where the distance across the series is short. A potential problem is

[2]Demand series is the product of usage series and the allocated virtual capacity. Both demand and usage series share the same correlation characteristics.

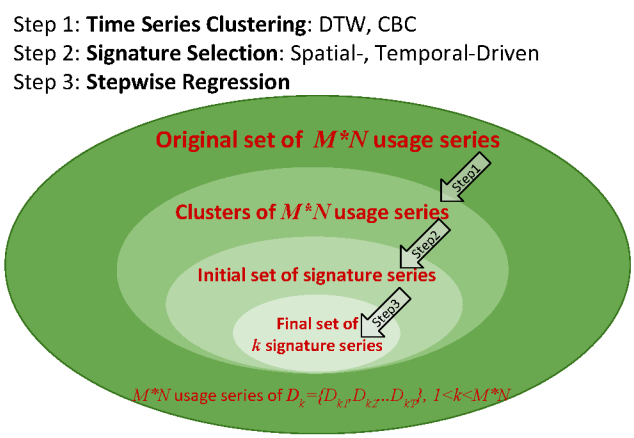


Fig. 7: Overview of searching for signature set.

that DTW falls short in capturing series within the cluster that are of larger distance. Correlation based clustering solves this problem by capturing highly correlated time series that are far enough apart and cannot be captured by DTW. Applying DTW on the four series shown in Figure 1 illustrates how clustering with DTW only offers a partial solution. DTW detects two: cluster 1 (VM1), cluster 2 (VM2, VM3, and VM4). CBC instead puts VM1 and VM3 in the same cluster, and VM2 with VM4 within another cluster. Indeed, the series $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$ of VM1 and VM2 can be well represented as linear models of the series $\boldsymbol{D_3}$ of VM3 and $\boldsymbol{D_4}$ of VM4 respectively, e.g., $\boldsymbol{D_1} = a_0 + a\boldsymbol{D_3}$, and $\boldsymbol{D_2} = b_0 + b\boldsymbol{D_4}$, where a_0, a, b_0, and b are scalars. In the remaining of this section we provide details on DTW and CBC, as well as how to select the signature series.

Dynamic Time Warping Clustering: The high level idea of DTW is to group series that show low *distance dissimilarity*. To obtain the distance dissimilarity between two series $P = \{p_1, p_2, ..., p_i, ..., p_n\}$ and $Q = \{q_1, q_2, ..., q_j, ..., q_m\}$, we first build a matrix that consists of the pair-wise squared distances, i.e., $d(p_i, q_j) = (p_i - q_j)^2$, between each pair of elements p_i and q_j in the two series. The distance dissimilarity $\lambda(n, m)$ of the two series is given by the wrapping path through the matrix that minimizes the total cumulative distance [14] and can be recursively computed as follows:

$$\lambda(i,j) = d(p_i, q_j) + min\{\lambda(i-1, j-1), \lambda(i-1, j), \lambda(i, j-1)\}. \quad (3)$$

Next, we apply hierarchical clustering [15] for any given number of clusters, ranging from 2 to $(M \times N)/2$ since we aim to reduce the original set to at least its half. We determine the optimal number of clusters, based on the average silhouette value [16] of all time series within each cluster. For each series i, its silhouette value $s(i)$ is defined as

$$s(i) = \frac{b(i) - a(i)}{max\{b(i), a(i)\}} \quad (4)$$

where $a(i)$ is the average distance dissimilarity between series i to all the other series within the same cluster using DTW while $b(i)$ is the lowest average distance dissimilarity between series i to all the series in a different cluster. The higher the silhouette value, the better the series lies within its cluster. For each number of clusters, we average the silhouette values of all series as the representative silhouette value. The optimal number of clusters is the one with the maximal silhouette value.

Correlation-based Clustering: CBC focuses on grouping series showing high correlation. For each box, we first compute the pairwise correlation coefficients, denoted as ρ, for all pairs of the $M \times N$ series. For a demand series $\boldsymbol{D_i}$, there are $(M \times N - 1)$ pairs $\rho_{i,l}, \forall l \neq i$. To form the clusters, we rank each series $\boldsymbol{D_i}, i \in [1, M \times N]$ first by the total number of $\rho_{i,l}$ above a threshold ρ_{Th}, and second by the mean value of the $\rho_{i,l}$ above the threshold. Compared with DTW based clustering, our proposed CBC provides customized clustering via setting different values of ρ_{Th}. Lower ρ_{Th} results in more aggressive clustering. After the series have been ranked, we select the topmost one and remove it together with all the series that are correlated with it with a correlation coefficient higher than the threshold. These series are now considered within a new cluster with the top ranked series being the signature series. This procedure continues by selecting the next topmost series still in the ranked list and ends when the ranked list becomes empty.

Step 2 - Signature Selection Methods: After clustering usage series, the next step is to select one signature series from each cluster, to represent all the other usage series in the same cluster. We propose two different methods of signature selection: either *spatial-driven* or *temporal-driven*. The intuition behind the *spatial-driven* method is to find the signature series that can best represent all the other series in the same cluster, while the *temporal-driven* method aims to choose the signature series that can be best predicted within each cluster. Specifically, the *spatial-driven* method selects the signature series with the strongest spatial dependency (i.e., the smallest distance dissimilarities or the highest correlation values) with all the other series in the same cluster, while the *temporal-driven* one selects the signature series with the highest value of goodness of temporal dependency within each cluster. After applying either signature selection method, each cluster will be represented by one signature series.

Step 3 - Stepwise Regression: To further reduce the number of signature series, we calculate the variance inflation factor, a metric that can detect multicollinearity in regression. For each series in the signature set, we regress it on the rest of signature series and obtain its VIF value [17]. A rule of thumb is that a VIF greater than 4 indicates a dependency with the other series in the initial set [17]. After detecting the risk of multicollinearity, i.e., at least one series has a VIF greater than 4, we perform standard stepwise regression to remove the series that can be represented as linear combinations of the other signature series.

B. Prediction Models

To predict all $M \times N$ demand series, we first predict the signature series $\boldsymbol{D_i}\,(i \in \Omega_s)$, using neural network models and their historical data [8]. To predict all dependent series, we regress each dependent series on the set of signature series,

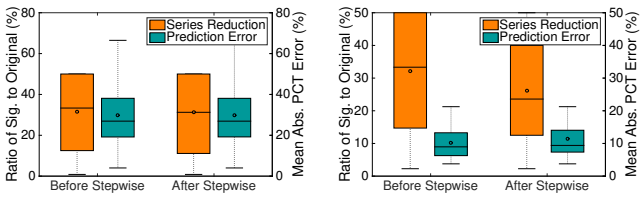
obtaining coefficients using ordinary least square estimates. We stress that the signature series predictions are not tied to the any specific model. Rather, any suitable prediction model can be easily plugged into our ATM framework.

In summary, we first leverage historical data to develop spatial models to define dependent series, and then select their respective signatures. Later, we use temporal models to predict the signature series and inexpensive linear transformation models to predict the dependent series.

C. Results on Spatial Models

Prior to moving to the proposed capacity planning policy, we present evaluation results of the proposed spatial models across the demand series of the trace data (6K boxes and 80K VMs) presented in Section II. The evaluation consists of two main parts. The first part focuses on the comparison between different clustering methods, while in the rest of the evaluation we turn the focus on the analysis of the two signature selection methods. Note that in the first part, we use the *spatial-driven* signature selection method. Since the purpose of spatial models is to use a minimum subset of original series to accurately represent the data center, the metrics of interest are: (I) the percentage of signature series out of the total demand series and (II) the prediction error[3].

In this section we only focus on the effectiveness of the spatial models, i.e., how close the dependent series are from the actual time series counterparts. The overall prediction accuracy of combining spatial models with temporal models is presented in Section V.



(a) VM (b) Box

Fig. 8: Usage series reduction and prediction errors are compared between two steps (before and after stepwise regression) for (a) VM and (b) box usage series. Notice that the left y-axis represents the ratio of the signature set to the original series, and the right y-axis represents the prediction error.

1) Results on DTW: In Figure 8, we present the boxplots of the series reduction and prediction errors on VM and box usage series of RAM and CPU, applying DTW. The boxplots in this figure show the 25^{th}, 50^{th}, and 75^{th} percentiles, the whiskers correspond to the extremes of the distribution, and the dot represents the average. Figure 8(a) compares the results on two steps (before and after stepwise regression) for VM usage series. We observe that DTW is quite aggressive in reducing the number of time series, such that there is almost no further reduction after applying stepwise regression. Both steps reduce the entire set to 31%, with around 30% prediction errors. Turning to prediction results on box usage series in Figure 8(b), we clearly observe that stepwise regression indeed helps further reduce the signature set (from 32% to 26%), with similar prediction error (from 10% to 11%). This result verifies the efficiency of stepwise regression in reducing the signature set without degrading prediction accuracy.

2) Results on CBC: Different from DTW where the number of clusters is determined by the silhouette value (see Eq. 4), CBC provides a customized interface that allows users to determine how aggressively they want to reduce the number of usage series (i.e., save the computational cost), via setting ρ_{Th} within $[0, 1]$. In this section, we focus on understanding the behavior of the proposed CBC, i.e., the trade-off between series reduction and prediction accuracy.

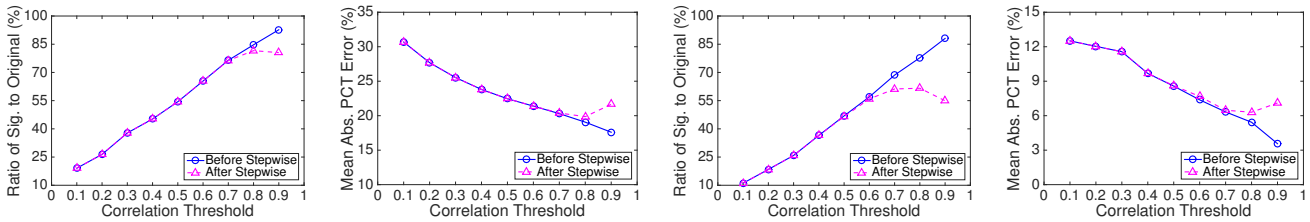
We use different values of ρ_{Th} in CBC, to measure the trade-off between series reduction and prediction accuracy. Figure 9 presents (a) the average percentage of signature series comparing to all series and (b) the mean APEs using CBC, with different levels of correlation thresholds for VM usage series. As expected, the higher the correlation threshold ρ_{Th}, the more accurate the prediction as series reduction is less aggressive. Interestingly, when focused on the case of the lowest correlation threshold $\rho_{Th} = 0.1$ with more than 80% of series reduction, we observe that CBC still achieves satisfactory prediction accuracy, with the mean APE around 27%. Notice that the average coefficient of variation of VM usage series is approximate 53%. In other words, simply using the mean to predict VM usage series will result in 53% prediction error. Compared to the mean-value prediction method, the proposed CBC still provides superb prediction accuracy even for an aggressive reduction of the usage series, The same conclusion is also applied to prediction results on box usage series, shown in Figures 9(c) and 9(d).

Figure 9 further verifies the effectiveness of stepwise regression. When ρ_{Th} is less than 0.6, stepwise regression does not reduce further the signature set, which is an outcome of aggressive clustering. With $\rho_{Th} > 0.6$, the chance of multicollinearity within the signature set becomes much higher, and stepwise regression furthers the reduction of the signature set, with only a small degradation in prediction accuracy. To quantify the effectiveness of stepwise regression, we define a new metric δ for both series reduction and prediction accuracy as follows:

$$\delta = \frac{\mathbf{Metric}_{before\_stepwise} - \mathbf{Metric}_{after\_stepwise}}{\mathbf{Metric}_{before\_stepwise}}. \quad (5)$$

Here $\mathbf{Metric}$ can be either the series reduction or prediction error. $\mathbf{Metric}_{before\_stepwise}$ represents the series reduction or prediction error before stepwise regression while $\mathbf{Metric}_{after\_stepwise}$ represents those after stepwise regression applied. Intuitively, δ quantifies the relative *increment* of series reduction or *decrement* of prediction accuracy after stepwise regression, which allows us to compare the two different metrics quantitatively. Higher increment of series reduction with less decrement of prediction accuracy suggests that applying stepwise regression results in further reduction of the signature set with ignorable errors introduced.

[3]The Prediction error in this paper refers to the absolute percentage error (APE), defined as: $APE = \frac{|Prediction - Actual|}{Actual}$.



(a) VM: Sig. set reduction (b) VM: Prediction error (c) Box: Sig. set reduction (d) Box: Prediction error

Fig. 9: Trade-off between series reduction and prediction accuracy for CBC for (a-b) VMs and (c-d) boxes.

Tables I-II summarize δ for both series reduction and prediction accuracy of three threshold values, i.e., $\rho_{Th} \in \{0.7, 0.8, 0.9\}$, for VM and box usage series, respectively. We observe that stepwise regression always achieves higher series reduction but with minimal reduction in prediction accuracy, which quantifies the effectiveness of stepwise regression.

TABLE I: Comparison of prediction accuracy and series reduction before and after stepwise regression applied for VM usage series prediction with CBC.

	$\rho_{Th} = 0.7$		
	Before Stepwise	*After Stepwise*	δ
Mean APE	20.3%	20.4%	4.1%
Reduction in Series	23.5%	23.7%	**6.6%**
	$\rho_{Th} = 0.8$		
	Before Stepwise	*After Stepwise*	δ
Mean APE	19.6%	19.9%	4.2%
Reduction in Series	15.3%	18.4%	**20.3%**
	$\rho_{Th} = 0.9$		
	Before Stepwise	*After Stepwise*	δ
Mean APE	17.6%	21.6%	22.3%
Reduction in Series	7.3%	19.5%	**166.5%**

TABLE II: Comparison of prediction accuracy and series reduction before and after stepwise regression applied for box usage series prediction with CBC.

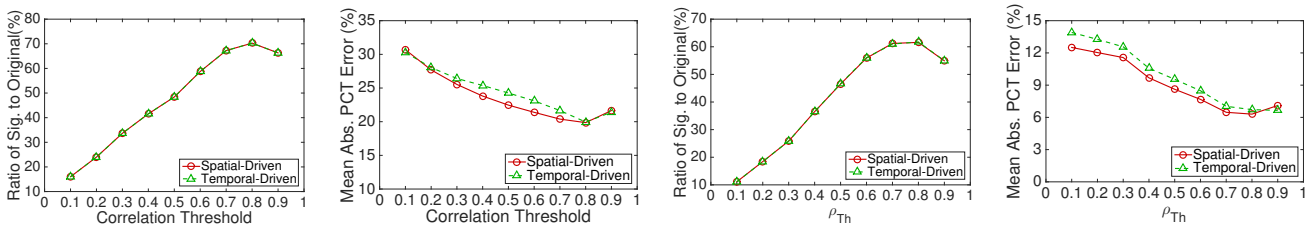
	$\rho_{Th} = 0.7$		
	Before Stepwise	*After Stepwise*	δ
Mean APE	6.3%	6.5%	2.3%
Reduction in Series	31.4%	38.8%	**23.5%**
	$\rho_{Th} = 0.8$		
	Before Stepwise	*After Stepwise*	δ
Mean APE	5.4%	6.3%	16.1%
Reduction in Series	22.2%	38.4%	**73.1%**
	$\rho_{Th} = 0.9$		
	Before Stepwise	*After Stepwise*	δ
Mean APE	3.5%	7.1%	99.3%
Reduction in Series	11.4%	45.1%	**278.4%**

3) Spatial Models - DTW v.s. CBC: Compared with DTW, one advantage of CBC is the *customized* correlation threshold to allow users to dynamically adjust how aggressive is the reduction in series. A possible question is that for a similar level of series reduction, between DTW and the proposed CBC, which one achieves better prediction accuracy.

To compare fairly the prediction accuracy between DTW and CBC, we select $\rho_{Th} = 0.2$ for CBC, where CBC has a similar reduction in series as DTW. DTW achieves around 70% and 75% series reduction for VM and box usage series respectively, while CBC results in more than 75% series reduction for both VMs and boxes on average. At the same time, the average prediction errors of DTW are 30% and 11% for VM and box usage series, while CBC achieves 27% and 11% prediction errors for VMs and boxes. The above observations suggest that, compared to DTW, CBC not only enables *customized* prediction efficiency but also achieves higher prediction accuracy with similar prediction cost.

4) Signature Selection - Spatial- v.s. Temporal-Driven: In Sections III-C1-III-C3, we apply the *spatial-driven* signature selection method. In this section, we compare two different signature selection methods in terms of the prediction efficiency of spatial models. In general, if the signature series lacks strong spatial dependency with other series in the same cluster, it may not represent the cluster well. While if the signature series has weak temporal dependency within itself, the signature series cannot be predicted well by temporal models. Consequently, it is important to understand how the two signature selection methods affect prediction efficiency. As a first step, we compare the effectiveness of *spatial-* and *temporal-driven* signature selection methods with CBC. Figure 10 shows (a) the average reduction in the signature set and (b) the overall prediction accuracy across different correlation thresholds, for *spatial-* and *temporal-driven* signature selection methods, in terms of VM usage series. Results on box usage series are shown in Figures 10(c) and 10(d).

Figures 10(a) and 10(c) show that both signature selection methods share the same amount of reduction in the signature set for VM and box usage series. This is expected because the same time series clustering method (namely CBC) is applied. Figures 10(b) and 10(d) present that the *spatial-driven* signature selection method achieves marginally better accuracy than the *temporal-driven* one, for both VM and box usage series. This is consistent with intuition since the *spatial-driven* method selects the signature series with the strongest spatial dependency from each cluster. Surprisingly, the *temporal-driven* method achieves comparable prediction accuracy. This implies that the selection of the signature series does not play a key role in prediction accuracy of spatial models. Moreover, given that temporal dependency dominates prediction accuracy when temporal models are used to predict the signature series, it is anticipated that *temporal-driven* signature selection would yield better overall prediction accuracy, considering both spatial and temporal models applied. Due to space limits, a comparison between two signature selection methods for



(a) VM: Sig. set reduction (b) VM: Prediction error (c) Box: Sig. set reduction (d) Box: Prediction error

Fig. 10: Comparison between two different signature selection methods with CBC for (a-b) VMs and (c-d) boxes.

DTW is not shown here, but the same conclusion is drawn.

IV. Capacity Planning

Being able to accurately predict future usage enables the very first step to actively manage usage-related tickets. Having future usage knowledge, it is possible to develop a capacity planning policy that can effectively reduce the number of usage tickets. Monitoring systems in modern data centers track resource usages at discrete windows, e.g., 15 minutes, termed as the ticketing window, and compare them with ticket thresholds to determine whether a ticket needs to be issued or not. To avoid incurring overreaction to transient loads, we set the window of capacity planning to be greater than the ticketing window. For the data centers considered here, ticket resolution occurs within a day of the ticket being issued, so setting the window to one day is reasonable and practical. This implies that the prediction horizon of the demand series needs to be also one day. Note that past work has shown that the accuracy of prediction decreases as the prediction horizon increases [8], so setting the prediction window to such a high value makes ATM more conservative than it can actually be. During each window, to reduce the usage tickets on VMs, ATM computes and actuates the virtual resource allocation of co-located VMs on boxes. Similarly, to reduce the box usage tickets, ATM performs capacity planning of physical resources for single-client boxes within the same client. The objective is to find optimal sizes for co-located VMs or single-client boxes to achieve the lowest number of tickets, subject to various resource constraints. The resources considered are: CPU measured in GHz and RAM measured in GB.

There exist a large body of resource allocation studies aiming to satisfy various performance targets, e.g., user response time, system utilization, and fairness. For example, max-min fairness [18], [19] is one of the most applied allocation policies that tries to guarantee the performance of small VMs, given the assumption of known demands. Our capacity planning problem can be viewed similarly but with the objective to minimize the occurrences of target utilization threshold violations.

textcolorblueFor the remaining of this exposition, we assume a fixed resource thresholds that trigger tickets. Yet, we stress that dynamic thresholds can be also used since resource allocation (and ticket reduction) is based on the resource prediction per ticketing window, which depends on the spatio-temporal models. We develop a capacity planning algorithm based on a rigorous optimization formulation, which is later transformed into a multi-choice knapsack problem (MCKP) with tunable discretization parameters. The introduction of such discretization parameters enables us to reduce the complexity and increase the safety margin in resource allocation. In contrast to spatial-temporal prediction models, the capacity planning algorithm treats CPU and RAM separately due to separate constraints on each resource. For simplicity, in the following section, we take the capacity planning problem of co-located VMs (namely virtual resource resizing) as an example to illustrate the proposed capacity planning algorithm.

A. Ticket Optimization Formulation

We formally introduce the problem, including notations and constraints, for resizing all co-located VMs on a single box. The foremost important constraint is that the summation of allocated virtual resources should be less than or equal to the total available virtual resource, i.e., $\sum_i C_i \leq C$, where C_i denotes the virtual capacity allocated to VM i, and C is the total available virtual capacity at the box. The decision variable is C_i and needs to be determined at the beginning of the capacity planning horizon.

The prediction module provides all demand series values for the entire capacity planning window, equal to T ticketing windows, for VM i, $\boldsymbol{D_i} = \{D_{i,1}, \dots D_{i,T}\}$. We introduce an indicator variable, $I_{i,t}$, when $I_{i,t} = 1$ a usage ticket occurs to VM i at ticketing window t, because the demand exceeds a certain threshold of the capacity, say, αC_i (e.g., $\alpha = 0.6$); otherwise $I_{i,t} = 0$. We aim to minimize the total number of tickets occurring on all co-located VMs during the capacity planning window. Thus, we can write the objective function as $\sum_i \sum_t I_{i,t}$. In summary, we can define the ticketing optimization problem as:

$$(\mathbb{R}) \min \quad \sum_i \sum_t I_{i,t} \tag{6}$$

$$s.t. \quad \sum_i C_i \leq C \tag{7}$$

$$D_{i,t} - \alpha C_i \leq D_{i,t} I_{i,j} \tag{8}$$

$$I_{i,t} \in \{0, 1\} \tag{9}$$

Constraint (8) ensures that $I_{i,t} = 1$, when the demand exceeds the ticket threshold αC_i; otherwise the objective function drives $I_{i,t}$ to zero. The problem $\mathbb{R}$ is a classical mixed integer linear programming (MILP), whose complexity greatly depends on the number of integer variables, i.e., the indicator variables $I_{i,t}$ in our case. The number of indicator variables for each box is thus the product of the number of ticketing windows, T, and the number of VMs, M.

1) Capacity Planning Algorithm: Instead of resorting to a standard MILP solvers, such as CPLEX [20], we transform the original problem into a multi-choice knapsack problem by Lemma IV.1: the optimal size for each VM must be equal to one of the demand values in $\boldsymbol{D_i}$ or 0. The advantages of transforming the original problem into a MCKP are twofold: (i) there exist a large number of efficient algorithms for MCKP and (ii) it allows for a reduction of the number of integer variables. We elaborate on the second point after formally introducing the transformation of the original optimization problem to MCKP.

Lemma IV.1. *For VM i, the optimal size $C_i* \in \boldsymbol{D_i} \cup \{0\}$, $\boldsymbol{D_i} = \{D_{i,1}, D_{i,2} \dots D_{i,T}\}$.*

Proof. If there exists an optimal solution (C_i*) for each VM (i) for the capacity planning problem, C_i* has to be in one of the three ranges: $[0, min\{\boldsymbol{D_i}\})$, $[min\{\boldsymbol{D_i}\}, max\{\boldsymbol{D_i}\})$, and $[max\{\boldsymbol{D_i}\}, +\infty)$. If C_i* is less than $min\{\boldsymbol{D_i}\}$, we argue that C_i* could be set to 0 and the objective function stays unchanged while the constraints are not violated. Similarly, it is proven that if C_i* is not less than $max\{\boldsymbol{D_i}\}$, C_i* can be set to $max\{\boldsymbol{D_i}\}$. If C_i* is in $[min\{\boldsymbol{D_i}\}, max\{\boldsymbol{D_i}\})$, sort $\boldsymbol{D_i}$ in a descending order as $\boldsymbol{D_i}^{descend} = \{O_1, O_2, ..., O_p, O_{p+1}, ...\}$. Following the same reasoning, it is possible to determine that $\exists\ q$, $C_i* \in [O_q, O_{q+1})$. In addition, setting C_i* equal to O_q, the minimum objective function can be obtained without any constraint violation. Hence the optimal size C_i* is either in $\boldsymbol{D_i}$ or 0. □

Based on Lemma IV.1, we can transform the original formulation into a multi-choice knapsack problem, whose complexity can be further simplified by reducing the number of indicator variables. We first introduce a reduced demand set with 0 added, denoted as $\boldsymbol{D_i'}$, containing the unique values of the original demands in decreasing order, $D'_{i,v+1} \leq D'_{i,v}$. According to Lemma IV.1, one of them is the optimal capacity. We note that $D'_{i,v}$ is not the same as $D_{i,t}$. The following small example illustrates the difference. Given a specific demand series $\boldsymbol{D_i} = \{30, 30, 40, 40, 23, 25, 60, 60, 60, 60\}$, its reduced series is $\boldsymbol{D_i'} = \{60, 40, 30, 25, 23, 0\}$ containing only the unique values plus 0 in descending order.

We introduce a new binary variable $Y_{i,v}$, denoting that the unique value $D'_{i,v}$ is chosen to be the capacity for VM i. The next step to reduce the problem into MCKP is to define the number of tickets, denoted $P_{i,v}$, seen by VM i when the value of $D'_{i,v}$ is chosen as capacity, i.e., $Y_{i,v} = 1$. Following the previous example of reduced demand set, we show an example of ticket calculation. Let us assume the current capacity is 70 and the ticketing threshold for issuing usage tickets is 60%. We thus know that demands greater than $70 \times 60\% = 42$ at any ticketing window will result into tickets. We can then obtain $\boldsymbol{P_i} = \{0, 4, 6, 8, 9, 10\}$. Due to the decreasing order of $\boldsymbol{D_i'}$, $\boldsymbol{P_i}$ has an increasing order, i.e., $P_{i,v+1} \geq P_{i,v}$. The total number of tickets for a box can thus be written as $\sum_i \sum_v Y_{i,v} P_{i,v}$ and the resource constraint of the total capacity as $\sum_i \sum_v Y_{i,v} D'_{i,v} \leq C$.

In summary, we reach a multi-choice knapsack problem, where items (in the original knapsack problem) are divided into subgroups and exactly one item needs to be selected from each group. Putting our problem into the context of multi-choice problem, we have M groups of VM demands and we need to choose exactly one demand from each group as their capacity. The decision variables are $Y_{i,v}$ denoting that a particular demand is chosen as the size for VM i, where $i \in [1, M]$ and that the number of tickets, $P_{i,v}$, can be seen as "weights". The transformed ticket reduction problem is:

$$(\mathbb{R}') \min \quad \sum_i \sum_v Y_{i,v} P_{i,v} \tag{10}$$

$$s.t. \quad \sum_i \sum_v Y_{i,v} D'_{i,v} \leq C \tag{11}$$

$$\sum_v Y_{i,v} = 1 \tag{12}$$

$$Y_{i,v} \in \{0, 1\} \tag{13}$$

The formulation of problem $\mathbb{R}'$ enables the introduction of a tunable parameter, ε, which decides the discretization of demand values. We illustrate this point using the running example of original series $\boldsymbol{D_i}$ and its reduced series $\boldsymbol{D_i'}$. The original formulation $\mathbb{R}$ has 11 integer variables (including the 0), whereas the transformed problem $\mathbb{R}'$ has only 6 integer variables. One can even further decrease the number of binary variables in $\boldsymbol{P_i}$ by discretizing the demand values, such as rounding off the first digit. For example using $\boldsymbol{D_i'} = \{60, 40, 30, 0\}$, where 23 and 25 are rounded up to 30. Another point worth mentioning is that we need to update the number of corresponding tickets too, i.e., $\boldsymbol{P_i} = \{0, 4, 6, 10\}$. Rounding up demands makes the capacity planning algorithm more aggressive in allocating resources. Consequently, we formally introduce a discretization factor, ε, which further reduces the complexity and provides a safety margin for resource allocation. We note that ε is only applied on the predicted series. In summary, the initial step computes $\boldsymbol{D_i'}$ from $\boldsymbol{D_i}$ using ε, and calculates their corresponding tickets, $\boldsymbol{P_i}$ for all co-located VMs i.

To solve the MCKP problem, we resort to the so-called minimal algorithm [21]. We illustrate the general idea in the context of our capacity planning problem. The algorithm chooses capacity candidates for each VM and shuffles around the capacity across VMs, comparing to the available capacity and marginal ticket reductions. For all VMs, it chooses capacity candidates that can incur a minimum number of tickets, i.e., starts from the maximum values in $\boldsymbol{D_i'}$. When there is no sufficient capacity to achieve such allocations for all VMs, the priority is given to the VM having the lowest marginal ticket reduction values (MTRV). MTRV represents the additional ticket increment when reducing one unit of capacity provisioning. Its formal definition is:

$$MTRV = \frac{P_{i,o} - P_{i,o-1}}{D'_{i,o-1} - D'_{i,o}}, \tag{14}$$

where o denotes the index of candidates in $\boldsymbol{D_i'}$. The VM with the lowest MTRV is always chosen to reduce the capacity provision from its current candidate value to the next one in $\boldsymbol{D_i'}$. Note that as $\boldsymbol{D_i'}$ is in decreasing order, the next candidate immediately implies a capacity reduction. Once the candidate

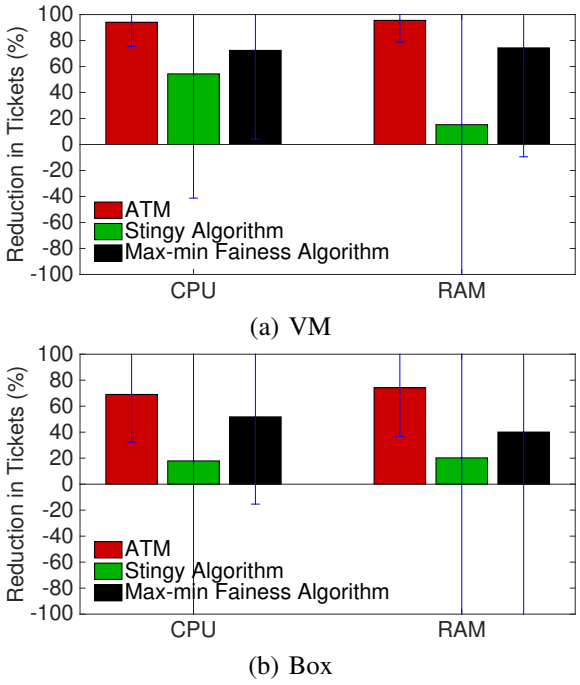
list is updated, the same process continues until the sum of all candidates is less or equal to the available capacity.

For a practical implementation, in addition to the constraint of total available capacity, it is also imperative to consider the lower and upper bounds of capacity. In order to avoid spillovers of unfinished demands from previous ticketing windows, we impose a lower bound on the VM capacity size, such that its peak usage before resizing is satisfied. Moreover, as any VM is not able to use more resources than the available resource amount of the underlying physical box, we introduce the allocation upper bound based on the box resource capacity. We can easily incorporate such lower and upper bounds into our capacity planning algorithm by limiting the values in $\boldsymbol{D_i'}$ for each VM i.

B. Results on Usage Ticket Reduction

Prior to moving on to the evaluation of the full-fledged ATM, i.e., the combination of spatial-temporal prediction and capacity planning, we first show how effective the proposed capacity planning algorithm is against existing resource allocation heuristics. For a fair comparison, the demand inputs are based on the original dataset described in Section II, instead of prediction. We implement the max-min fairness algorithm [18] and a "stingy" algorithm which only allocates the capacity according to the lower bound, i.e., the maximum demand regardless of the ticket threshold, often used in practice. In contrast, the max-min algorithm considers the fairness of resource allocation among co-located VMs or single-client boxes. For example, the max-min algorithm starts to allocate resources to all VMs based on the demand of the smallest VM, considering its ticket threshold, and continues onto VMs in the increasing order of their demands until all capacity is exhausted. Similarly, for single-client boxes, the max-min algorithm always first determines the appropriate physical capacity for the smallest box, and then continues to other boxes in the same client.

Here, we evaluate the trace data of April 3, 2015 across all 6K boxes within around 300 clients and set the threshold to trigger usage tickets to 60%, i.e., in every ticketing window the monitoring system checks if the average usage of CPU or RAM exceeds the 60% of the allocated capacity for both VMs and boxes. Figure 11 summarizes the mean ticket reduction and its standard deviation, when applying the proposed ATM capacity planning, max-min fairness, and stingy algorithms. As expected, the stingy algorithm is completely unaware of the ticket threshold. On average it achieves a reduction of 54% and 15% on VM usage tickets for CPU and RAM respectively, see Figure 11(a). Similar observations hold for the stingy algorithm on box usage tickets, with a reduction of only 20% for both CPU and RAM, shown in Figure 11(b). Max-min fairness reduces VM usage tickets by around 70% for both CPU and RAM, and box usage tickets by around 50% and 40% for CPU and RAM, respectively. This is still roughly 30% worse than the proposed ATM capacity planning results. Due to the nature of favoring small machines, large machines can be severely punished under max-min fairness resulting in no ticket reduction, this explains the high standard deviation under max-min fairness for both VMs and boxes.



(a) VM

(b) Box

Fig. 11: Ticket reduction of CPU and RAM for (a) VMs and (b) boxes: comparing ATM, max-min fairness, and stingy algorithms.

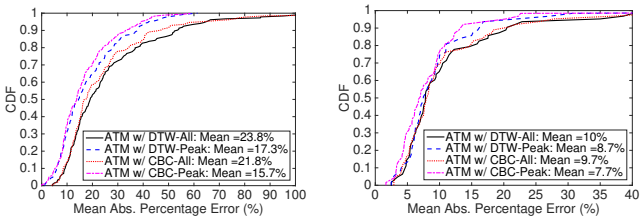
The proposed capacity planning algorithm does exceptionally well. It achieves 95% VM usage ticket reductions and 70% box usage ticket reductions for both CPU and RAM, a remarkable improvement for both performance and cost. This is also attributed to the fact that the systems of the original traces are equipped with abundant resources, i.e., typically data centers are lowly utilized [6]. By simply shuffling resources across co-located VMs and efficient capacity planning on single-client boxes, we are able to achieve significant performance gain. Moreover, we also eliminate the overhead of inspecting and resolving a large number of usage tickets, a process that is known to be expensive.

V. Evaluation

We extensively evaluate ATM on a large number of data center production traces. For the first part of evaluation, we focus on illustrating the effectiveness and versatility of ATM in time series prediction using spatial-temporal prediction models. In the second part of evaluation, we focus on presenting the effectiveness of ATM in ticket reduction to improve system dependability and to reduce the high cost associated with ticket resolution.

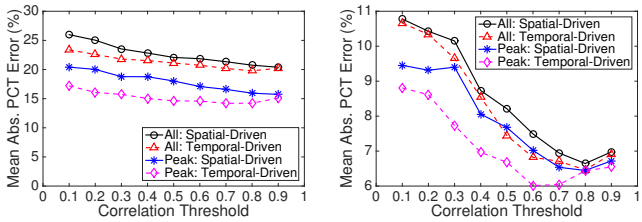
A. Analysis on Spatial-Temporal Models

We engage training of the signature series for 5 days and then predict the following day. For spatial models, we consider both DTW and CBC. The temporal models used for the signature series are feed-forward neural networks [8], trained using back-propagation via MATLAB [22]. ATM performs the prediction of 16000 usage series, each of which has 96



(a) VM (b) Box

Fig. 12: CDF of prediction accuracy between DTW and CBC methods for (a) VM and (b) box usage series.



(a) VM (b) Box

Fig. 13: Comparison of prediction errors between *spatial-* and *temporal-driven* signature selection methods with CBC for (a) VM and (b) box usage series.

ticketing windows, with each window being 15 minutes long. To check the generalization of our spatial-temporal models, the tested usage series include VMs and boxes for both CPU and RAM. We note that results presented in this section differ from Section III, where only the proposed spatial models are evaluated, excluding the temporal prediction models. Here, we have the full effect of both prediction models.

1) Spatial Models - DTW v.s. CBC: As discussed in Section III-C, *temporal-driven* signature selection method is supposed to dominate the prediction accuracy. Consequently, in this part of comparing spatial models, we leverage a *temporal-driven* method to select signature series from each cluster. To fairly compare DTW and CBC, we set $\rho_{Th} = 0.2$ for CBC in order to have similar usage series reduction between DTW and CBC.

In Figures 12(a) and 12(b), we present the CDFs of the prediction accuracy of ATM in terms of APE with different spatial models, i.e., DTW and CBC, for VM and box usage series respectively. The average prediction errors of VM usage are 23.8% and 21.8%, for DTW and CBC respectively, see Figure 12(a). Given that the peak usages trigger tickets, it is also important to check the prediction errors of peaks, i.e., usage higher than 60%, also shown in Figure 12 The figure shows that the average peak errors across all VM usage series are 17.3% and 15.7% for DTW and CBC, respectively. This shows that neural networks can capture well the temporal dynamics of the signature series. Similar observations are seen in box usage series prediction, see Figure 12(b). Note that our proposed CBC beats DTW with higher accuracy in overall and peak VM usage series prediction, as well as box usage series. We further note that compared with VM usage series prediction, we achieve even better prediction for box usage series, see Figures 12(a) and 12(b). This is an outcome of the strong temporal dependency within the box usage series, as discussed in Section II-C.

2) Signature Selection - Spatial- v.s. Temporal-Driven: To evaluate the efficiency of two different signature selection methods, we consider their effectiveness when used with CBC. Consistent with the focus of this section, we also report on overall prediction errors after neural networks prediction is sdone. In Figures 13(a) and 13(b), we compare the mean prediction accuracy with *spatial-* and *temporal-driven* signature selection methods across different correlation thresholds, in terms of all and peak usages, for VMs and boxes. It is clearly shown that in all cases, the *temporal-driven* signature selection method outperforms the *spatial-driven* one. The lower the correlation threshold, the bigger the prediction difference between them. It is clear that taking advantage of both spatial and temporal dependencies achieves higher efficiency in both prediction accuracy and computational cost.

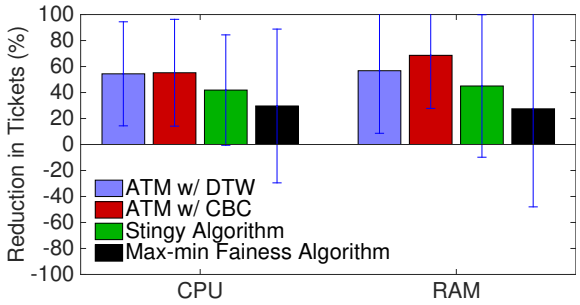
B. ATM on Ticket Reduction

In this section, we show how different configurations of ATM can proactively reduce the number of tickets. After obtaining the predicted VM series, ATM triggers the capacity planning algorithm for every box to determine the near optimal CPU and RAM capacity for all co-located VMs. Similarly for each client, ATM plugs in the predicted box series to perform the capacity planning algorithm across single-client boxes for both CPU and RAM. We stress that this analysis is post-hoc, i.e., we can not change the size of the actual VMs or physical boxes in the trace, we focus only on ticket reduction via ATM. In the remaining of this section, we assume that usage tickets related to CPU and RAM are automatically issued when utilization is greater than 60%.

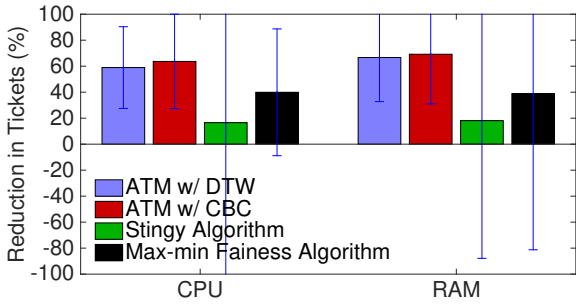
Figure 14 compares the results of average ticket reduction using two different versions of ATM (i.e., DTW and CBC) against the max-min fairness, and stingy policies, see Section IV. Each bar illustrates the mean and standard deviation of ticket reduction across tested boxes (in Figure 14(a)) and clients (in Figure 14(b)) divided into CPU and RAM tickets. The key observations are the following. Both versions of ATM are able to achieve a higher ticket reduction, around 60% and 70% for CPU and RAM, respectively, compared to the other two heuristics. We also like to point out that the standard deviation is high for stingy and max-min fairness algorithms, indicating instability of these two heuristics. However, ATM suffers from much lower variation of ticket reduction, which verifies the robustness of our proposed ATM. In summary, ATM achieves efficient and stable ticket reduction for both VM and box usage series.

VI. Related Work

Ticketing systems are widely used to improve on system dependability, e.g., slow responsiveness, failure [5], software bugs [23], [24] and system misconfigurations [25]. Prior art in ticketing systems centers on two directions: derive system management for software concurrency [23], database systems [4], and distributed data-intensive systems [26] but



(a) VM



(b) Box

Fig. 14: Comparing ticket reduction of CPU and RAM for (a) VMs and (b) boxes: comparing ATM, max-min fairness, and stingy algorithms.

also to develop automatic detection systems for different types of tickets, bugs [24] and software misconfigurations by leveraging the rich correlation between configuration entries [25]. Machine learning has been used for automating ticket resolution recommendation [27], [28], [9]. To the best of our knowledge, there are no proactive methodologies for preventing ticket issuing, with the exception of models for database reconfiguration [29]. The proposed ATM policy fills this gap by not only deriving management insights for usage ticket patterns, but also by developing novel prediction and ticket avoidance strategies via capacity planning.

Time series prediction and analysis have been viewed as an excellent way to develop proactive system management policies [30], [31]. Temporal models such as ARIMA models [10] have been widely used to predict time series with strong seasonality. Sophisticated neural network models show a strong promise in capturing highly irregular time series at a cost of long training overheads [32]. Time series clustering aims to explore spatial dependency, either through their original series, e.g., DTW [14], or extracted features [13], e.g., moments. ATM combines spatial with temporal models to contain the cost of neural network training and scales well for very large numbers of time series.

Virtualization technology has become the industry standard offering great opportunities to multiplex physical resources over a large number of VMs. There are two ways to change the efficiency of resource multiplex ratios: by sizing the virtual resource capacities [33] and by dynamically consolidating VMs [34]. While dynamically changing the degree of VM consolidation is shown effective to take advantage of the time variability of the workload [35], the overhead of migrating VMs can greatly reduce its performance benefits. On the contrary, sizing resource of co-located VMs incurs less system overhead [33]. A central question of multiplexing resources is how to strike a good tradeoff of fairness and performance for workloads, e.g., latency [36] and throughput [37]. Fairness driven policies, e.g., max-min fairness, proportional fairness, and bottleneck resource fairness [38], have been proposed for various systems components, including storage systems [37] and networks [39]. The capacity planning algorithm proposed in ATM differs from related work by its objective to reduce the number of usage tickets. While max-min fairness also reduces the number of tickets, it cannot achieve this as effectively as ATM since ticket reduction is a side-effect rather than a main focus.

Compared to [1], here we apply the proposed spatial-temporal prediction model and capacity planning algorithm to a new data set, namely box usage series from IBM private data centers. Evaluation results again verify the efficiency of ATM. Moreover, we propose a new, improved signature selection method for the spatial-temporal prediction model, by reducing computation overhead while maintaining accurate prediction. The improved version of ATM presented here provides a customized interface for users to determine the trade-off between prediction accuracy and prediction cost. In summary, this paper not only more comprehensively evaluates ATM [1], but also further improves its accuracy and effectiveness.

VII. Concluding Remarks

We presented ATM, a methodology to achieve efficient time series prediction and capacity planning so as to reduce VM and box usage tickets that are issued in production data centers. We have shown the effectiveness of ATM in predicting usage series in production data centers by exploiting spatial-temporal usage patterns across/within co-located VMs and single-client boxes, and by using detailed prediction of a small subset of the usage series, allowing the methodology to scale well. This prediction drives the development of a capacity planning policy that is shown effective on a production trace. In future work we intend to use ATM's prediction abilities to drive online dynamic workload management.

Acknowledgment

The research presented in this paper has been supported by NSF grants CCF-1218758 and CCF-1649087, EU commission FP7 GENiC project (Grant Agreement No.608826), and the Swiss National Science Foundation (projects 200021_141002 and 407540_167266).

References

[1] J. Xue, R. Birke *et al.*, "Managing data center tickets: Prediction and active sizing," in *DSN*, 2016.
[2] Y. Liang, Y. Zhang *et al.*, "Bluegene/l failure analysis and prediction models," in *DSN*, 2006.
[3] I. Giurgiu, J. Bogojeska *et al.*, "Analysis of labor efforts and their impact factors to solve server incidents in datacenters," in *CCGrid*, 2014.
[4] I. Giurgiu, A.-D. Almasi *et al.*, "Do you know how to configure your enterprise relational database to reduce incidents?" in *IM*, 2015.
[5] R. Birke, I. Giurgiu *et al.*, "Failure analysis of virtual and physical machines: patterns, causes and characteristics," in *DSN*, 2014.

[6] R. Birke, A. Podzimek *et al.*, "State-of-the-practice in data center virtualization: toward a better understanding of VM usage," in *DSN*, 2013.

[7] R. Birke, M. Bjoerkqvist *et al.*, "(Big) data in a virtualized world: volume, velocity, and variety in cloud datacenters," in *FAST*, 2014.

[8] J. Xue, F. Yan *et al.*, "PRACTISE: robust prediction of data center time series," in *CNSM*, 2015.

[9] M. M. Botezatu, J. Bogojeska *et al.*, "Multi-view incident ticket clustering for optimal ticket dispatching," in *SIGKDD*, 2015.

[10] C. Chatfield, *The analysis of time series: an introduction*. CRC press, 2013.

[11] M. Vlachos, P. Yu *et al.*, "On periodicity detection and structural periodic similarity," in *SDM*, 2005.

[12] L. M. Leemis and S. K. Park, *Discrete-event simulation: a first course*. Pearson Prentice Hall Upper Saddle River, NJ, 2006.

[13] B. D. Fulcher and N. S. Jones, "Highly comparative feature-based time-series classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 12, 2014.

[14] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series." in *KDD workshop*, vol. 10, no. 16, 1994.

[15] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*. Springer, 2005.

[16] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, 1987.

[17] M. Kutner, C. Nachtsheim *et al.*, *Applied Linear Regression Models*. McGraw-Hill Education, 2004.

[18] L. Tassiulas and S. Sarkar, "Maxmin fair scheduling in wireless networks," in *INFOCOM*, vol. 2, 2002.

[19] A. Ghodsi, M. Zaharia *et al.*, "Dominant resource fairness: fair allocation of multiple resource types," in *NSDI*, 2011.

[20] CPLEX Optimizer. [Online]. Available: http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html

[21] D. Pisinger, "A minimal algorithm for the multiple-choice knapsack problem," *European Journal of Operational Research*, vol. 83, no. 2, 1995.

[22] H. Demuth and M. Beale, "Matlab neural network toolbox users guide version 6. the mathworks inc," 2009.

[23] S. Lu, S. Park *et al.*, "Learning from mistakes: a comprehensive study on real world concurrency bug characteristics," in *ASPLOS*, 2008.

[24] A. Nistor, P.-C. Chang *et al.*, "Caramel: detecting and fixing performance problems that have non-intrusive fixes," in *ICSE*, 2015.

[25] J. Zhang, L. Renganarayana *et al.*, "Encore: exploiting system environment and correlation information for misconfiguration detection," *ACM SIGPLAN Notices*, vol. 49, no. 4, 2014.

[26] D. Yuan, Y. Luo *et al.*, "Simple testing can prevent most critical failures: an analysis of production failures in distributed data-intensive systems," in *OSDI*, 2014.

[27] W. Zhou, L. Tang *et al.*, "Resolution recommendation for event tickets in service management," in *IM*, 2015.

[28] Q. Shao, Y. Chen *et al.*, "Easyticket: a ticket routing recommendation engine for enterprise problem resolution," *VLDB*, vol. 1, no. 2, 2008.

[29] I. Giurgiu, M. Botezatu *et al.*, "Comprehensible models for reconfiguring enterprise relational databases to avoid incidents," in *CIKM*, 2015.

[30] N. Tran and D. A. Reed, "Automatic ARIMA time series modeling for adaptive I/O prefetching," *IEEE Transactions on Parallel Distributed Systems*, vol. 15, no. 4, 2004.

[31] Z. Zhuang, H. Ramachandra *et al.*, "Capacity planning and headroom analysis for taming database replication latency: experiences with linkedin internet traffic," in *ICPE*, 2015.

[32] R. Livni, S. Shalev-Shwartz *et al.*, "On the computational efficiency of training neural networks," in *NIPS*, 2014.

[33] S. Spinner, N. Herbst *et al.*, "Proactive memory scaling of virtualized applications," in *CLOUD*, 2015.

[34] M. Wang, X. Meng *et al.*, "Consolidating virtual machines with dynamic bandwidth demand in data centers," in *INFOCOM*, 2011.

[35] C. Delimitrou and C. Kozyrakis, "Quasar: resource-efficient and QoS-aware cluster management," in *ASPLOS*, 2014.

[36] A. Gulati, A. Merchant *et al.*, "pClock: an arrival curve based approach for QoS guarantees in shared storage systems," in *SIGMETRICS*, 2007.

[37] H. Wang and P. Varman, "Balancing fairness and efficiency in tiered storage systems with bottleneck-aware allocation," in *FAST*, 2014.

[38] T. Bonald and J. Roberts, "Multi-resource fairness: objectives, algorithms and performance," in *SIGMETRICS*, 2015.

[39] A. Sridharan and B. Krishnamachari, "Maximizing network utilization with max–min fairness in wireless sensor networks," *Wireless Networks*, vol. 15, no. 5, 2009.

Ji Xue received his Ph.D. degree in Computer Science from College of William and Mary in 2017 under the supervision of Prof. Evgenia Smirni. He is currently a software engineer in Google. His research interests broadly lie in system reliability analysis, performance optimization and modeling, cloud computing, data center, machine learning, and data mining, especially in time series.

Robert Birke received the Ph.D. degree from the Politecnico di Torino in 2009 with the telecommunications group under the supervision of Prof. F. Neri. He is currently a Post-Doctoral Fellow with the Cloud Server Technologies Group, IBM Research Zurich Laboratory. He has co-authored over 40 scientific papers. His main research interests are high performance computing, cloud computing, and datacenter networks with special focus on performance, quality of service, and virtualization.

Lydia Y. Chen is a research staff member at the IBM Zurich Research Lab, Zurich, Switzerland. She received a Ph.D. in Operations Research and Industrial Engineering from the Pennsylvania State University in Dec 2006. Her research interests include performance modeling in multi-core systems and power-workload management in data centers. She has served on several technical program committees in various performance and network conferences, including DSN, INFOCOM, Globecom, ICC, ICNC, and ICCCN, and has served as an organizer of the Data Center Performance (DCPerf) workshop in 2011-2013.

Evgenia Smirni is the Sidney P. Chockley professor of computer science at the College of William and Mary, Williamsburg, VA. She holds a diploma degree in computer science and informatics from the University of Patras, Greece (1987) and a Ph.D. in computer science from Vanderbilt University (1995). Her research interests include queuing networks, stochastic modeling, resource allocation, storage systems, cloud computing, workload characterization, and modeling of distributed systems and applications. She is an ACM Distinguished scientist, and a member of the IEEE and the Technical Chamber of Greece.