Workload Characterization and Prediction in the Cloud: A Multiple Time Series Approach

Arijit Khan, Xifeng Yan
University of California, Santa Barbara
Santa Barbara, CA 93106
{arijitkhan, xyan}@cs.ucsb.edu

Shu Tao, Nikos Anerousis
IBM T. J. Watson Research Center
Hawthorne, NY 10532
{shutao, nikos}@us.ibm.com

Abstract—**Cloud computing promises high scalability, flexibility and cost-effectiveness to satisfy emerging computing requirements. To efficiently provision computing resources in the cloud, system administrators need the capabilities of characterizing and predicting workload on the Virtual Machines (VMs). In this paper, we use data traces obtained from a real data center to develop such capabilities. First, we search for repeatable workload patterns by exploring cross-VM workload correlations resulted from the dependencies among applications running on different VMs. Treating workload data samples as time series, we develop a co-clustering technique to identify groups of VMs that frequently exhibit correlated workload patterns, and also the time periods in which these VM groups are active. Then, we introduce a method based on Hidden Markov Modeling (HMM) to characterize the temporal correlations in the discovered VM clusters and to predict variations of workload patterns. The experimental results show that our method can not only help better understand group-level workload characteristics, but also make more accurate predictions on workload changes in a cloud.**

I. Introduction

Cloud computing [1] promises high scalability, flexibility and cost-effectiveness to satisfy emerging computing requirements. To realize these promises, cloud providers need to be able to quickly plan and provision computing resources, so that the capacity of the supporting infrastructure can closely match the needs of new applications. To this end, it calls for mechanisms to characterize and predict workload continuously.

Our goal is to develop such a mechanism, using data traces obtained from an in-production, *private cloud* [1] environment. The private cloud studied in this paper supports hundreds of enterprise customers, each runs 100 to 5000 Virtual Machines (VMs). Most VMs are provisioned to run certain business applications. For instance, a retail chain customer may have a set of VMs running 3-tiered web applications to support online orders. The workloads on these VMs are often driven by repeatable business behavior and hence, exhibit temporal and spatial correlations. Temporally, business behavior may show time-of-day effects. Spatially, when many users send online orders in the above example, the workload on front-end Web servers will increase, which would lead to workload increases on the middleware and database servers. As a result, workload variations on these servers may exhibit "clustered" behavior. This observation, together with the fact that *complete application configurations on customers' VMs are often unavailable to the cloud provider*, motivated this study.

Our approach is to first develop a model to capture groups of VMs that behave in frequent and repeatable patterns. Such behavior patterns make some workload variations predictable. Based on this model, we then design a technique that predicts the workload of individual VMs. For simplicity of exposition, we limit our discussions to CPU utilization in the paper, but our approach is applicable to other resource demands in the cloud, such as memory, I/O, network, etc.

The problem of workload characterization and prediction has been widely studied in the past [2], [3], [4], [5], [6]. Among them, a set of works focus on creating mathematical models that can be used to represent typical workload for Web servers [7], [8], [9], high performance computing platforms [3] or networks [4]. More recently, some work has been done toward characterizing the workload in a cloud environment [10], [11], [12], [13]. However, these studies are concerned with statistically understanding and reproducing computing tasks (e.g., MapReduce tasks) scheduled on a cloud. The work on capacity management and VM placement typically employs some workload modeling and prediction techniques [14], [15], [16], [17], [18]. These methods depend on the statistics of individual workload time series to predict future resource demand. *Different from the existing works, we apply a multiple time series approach, so that workload is analyzed at the group level, rather than at the individual VM level.* As we shall see, this helps to achieve a deeper understanding of workload characteristics and greater prediction accuracy.

We make the following unique contributions in this paper. First, we propose a new means of characterizing correlated workload patterns across VMs resulted from the dependencies of applications running on them. Treating workload data samples as multiple time series, we introduce a co-clustering algorithm to identify VM groups and the time periods in which certain workload patterns appear in a group. Second, we use Hidden Markov Model (HMM) to explore the temporal correlations in workload pattern changes. This allows us to predict individual VM's workload based on the groups found in the previous step. Our study is based on real measurement data collected from a production cloud environment, hence provides insights for system administrators to understand the typical cloud workload patterns and to better manage resources.

II. Background

Our study is based on workload traces collected from a production cloud environment that supports a number of enterprise customers. A customer runs business applications on VMs provisioned to it. Each application has one or more components, each of which runs on one or multiple VMs. From the service provider's perspective, application configurations on these VMs are not all known. A software agent is deployed on every VM to periodically collect performance data, including CPU, memory, disk and I/O utilization, etc., and reports the data back to a central server, which then stores all performance data into a database. We obtained the CPU utilization traces from this database for our study.

The agents collect CPU workload data at a minimum time granularity of 15 minutes. We obtained a total of 21 days of traces from 3019 VMs that belong to 3 different customers for analysis. The trace for each VM comprises a time series. Due to space constraint, we only show the results with 1212 VMs that belong to customer A. The corresponding results for customer B and C are available in the technical report [19].

Fig. 1 shows the sample time series for 3 VMs collected in two weeks. We see that despite some "noise", all VMs exhibit certain patterns in their workload e.g., the diurnal workload variations are obvious across all VMs shown in the figure. In the bottom two figures, we can see workload pattern shifts during certain period of time. It is also noticeable that there exist correlated behaviors across different servers, either due to time-of-day effect, or due to application-level workload correlations. Hence, it is possible to mine the correlated workload patterns, and use them to predict workload variations.

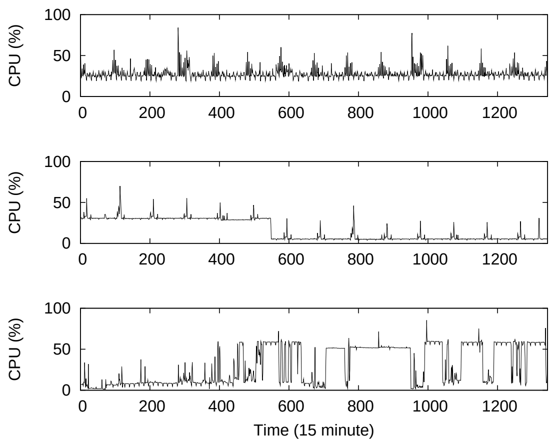


Fig. 1. Sample CPU workload time series on three VMs

III. Workload Characterization

Our first goal is to identify common workload patterns across multiple VMs. To achieve that, we discretize CPU utilization time series into several levels. In our data, the CPU utilization of each VM is represented as a continuous time series, with values between 0 and 100 (in percentages). We discretize each time series into five workload levels (e.g., 1 to 5), by fiting the data with a Gaussian Hidden Markov Model (GHMM) [19], with each hidden Markov state corresponding to a workload level. Next, we search for groups of VMs that frequently show certain combinations of workload levels.

A. *Group-based Workload Analysis*

With the discretized time series data, we analyze its variations over time and in particular, search for common patterns that can be potentially useful for workload prediction. Such patterns can be both *temporal* and *spatial*. Temporally, VMs that consistently have certain workload levels suggest that we can predict their future workload levels based on historical observations. However, as we show later in this paper, predictions based on data monitored from individual VM often leads to inaccurate results. This is because individually, the workload measures are more noisy and random, hence less predictable. We find that when VMs are configured to run some applications collaboratively, their workloads tend to vary in a correlated fashion. As a result, such spatial correlations can be used to filter out the measurement noise at the individual VM level, therefore to improve prediction accuracy. This motivates us to develop a model that captures common workload patterns along both temporal and spatial dimensions. In our study, we adopt the concept of co-clustering [20] for this purpose.

B. *VM Grouping by Co-clustering*

Using discretized workload time series, we identify groups of VMs that show recurring workload patterns. The input data can be represented by an $N \times M$ matrix $\mathbb{A}$, where N is the number of time intervals and M is the number of VMs. A co-cluster is a sub-matrix of the original matrix where all rows of this sub-matrix have similar values. For example, in Fig. 2, we can potentially identify two VM groups or co-clusters: one consisting of VMs s_1, s_2 and s_3, and the other consisting of VMs s_2, s_3, and s_4. The first group showed workload pattern $\{s_1 = 3, s_2 = 1, s_3 = 4\}$ in time intervals t_1, t_2 and t_N, while the second group showed workload pattern $\{s_2 = 1, s_3 = 4, s_4 = 2\}$ in t_2 and t_3. Note that a VM can potentially belong to multiple groups. In the above example, workload patterns in a group are identical in a number of time intervals. In practice, however, this condition can be relaxed to accommodate noisy measurement data. In other words, we search for groups of VMs that show *similar* behavior over time. We next describe a co-clustering method to identify such patterns.

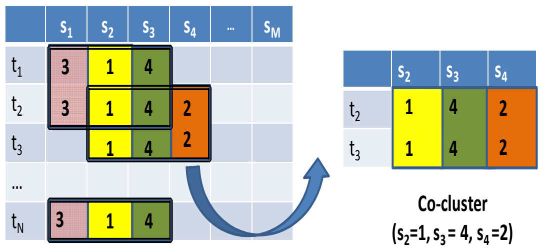


Fig. 2. An example of co-clustering

1) Co-clustering Method: Let us consider a matrix $\mathbb{A}$ with N rows and M columns as discussed above. A co-cluster is a submatrix $\mathbb{C} = (I, J)$ of A, with I being a subset of rows (time intervals) and J a subset of columns (VMs). The *rank* of a co-cluster is defined by $|I| \times |J|$. We denote the element at i-th row and j-th column of a co-cluster $\mathbb{C} = (I, J)$ as a_{ij}, and the most frequent element at the j-th column as a_j. Let d denote the total number of discretized levels. We define the *consistency* of co-cluster $\mathbb{C}$ as

$$const(\mathbb{C}) = \min_{j \in J} \frac{\sum_{i \in I}(1 - \frac{|a_{ij} - a_j|}{d})}{|I|}, \quad (1)$$

which measures the minimum variation in the columns of a co-cluster. If $const(\mathbb{C}) = 1$, then all rows in $\mathbb{C}$ are identical. The closer is $const(\mathbb{C})$ to 0, the less similar are the rows in $\mathbb{C}$. Formally, we define the co-clustering problem as follows.

Problem Statement 1: Given an $N \times M$ matrix A, find all its submatrices $\mathbb{C} = (I, J)$, so that $const(\mathbb{C}) \geq \theta$, where $0 < \theta \leq 1$ is a predefined consistency threshold.

We develop a technique to identify all highly-ranked (i.e., of larger size) co-clusters by extending the method proposed in [20]. Our algorithm contains three major steps. First, it selects a pair of time intervals in which the most number of VMs have same workload levels. Next, using this interval pair and the selected VMs as seed, it expands the sub-matrix both temporally and spatially, subject to the constraints on the size and consistency measure of the resulting submatrix. These two steps are iterated until no more qualifying submatrices can be found. Finally, the algorithm examines all resulting submatrices, removes highly overlapping ones, and the resulting submatrices are the co-clusters we look for. The details of this algorithm are explained as follows.

(a) Seed Selection: Given matrix $\mathbb{A} = \{a_{i,j}\}$, a pair of time intervals (rows) (i_1, i_2) can be considered as a seed, if they satisfy one of the following two conditions: i) neither row i_1 nor row i_2 has been included for any VM in previously identified co-clusters, or ii) both row i_1 and row i_2 have been included in previously identified co-clusters, but not in the same co-cluster, and there exists at least one VM s that has not been included in any previously identified co-clusters and satisfies $s(i_1) = s(i_2)$ (i.e., the workload levels of VM s are the same in time interval i_1 and i_2). With this seed, the algorithm builds an initial co-cluster $\mathbb{C} = (I, J)$ as $I = \{i_1, i_2\}$, $J = \{s : s(i_1) = s(i_2)\}$, and proceeds to the next step, co-cluster expansion.

The algorithm also maintains a set of all qualifying time interval pairs in the descending order of $|\{s : s(i_1) = s(i_2)\}|$, i.e., the number VMs having unchanged workload in those time intervals. The algorithm iterates through the seed set and removes disqualified seeds after each iteration, based on the additional co-clusters identified, until the set is empty.

(b) Co-cluster Expansion: After the seed selection step, the consistency of the initial co-cluster $\mathbb{C} = (I, J)$ is 1. The algorithm seeks to expand this initial co-cluster by including both more time intervals (i.e., row-wise expansion) and more VMs (i.e., column-wise expansion).

It first performs row-wise expansion as follows. It iteratively adds a new row $k \notin I$ to I, so that the highest consistency of the resulting co-cluster $\mathbb{C}$ can be maintained. The expansion continues until there is no more qualifying rows to be added

Next, the algorithm performs column-wise expansion in a similar way. It iteratively adds new column $l \notin J$ to J, while maintaining the consistency of the resulting co-cluster to be at least θ, until further expansion is no longer possible.

We note that row-wise expansion is performed before column-wise expansion, since a co-cluster that exists for more time intervals, rather than one that spans fewer time intervals but contains more VMs, is better for prediction.

After each round of expansion, the algorithm goes back to the seed selection step and repeats the above two procedures.

(c) Overlap Removal: After the iterations of step (a) and (b), the resulting co-clusters may overlap with each other, which make the co-cluster model unnecessarily complex. To simplify the model, we post-process co-clusters by removing those smaller ones that significantly overlap with other larger ones. We define overlap between two co-clusters $\mathbb{C}_1 = (I_1, J_1)$ and $\mathbb{C}_2 = (I_2, J_2)$ as

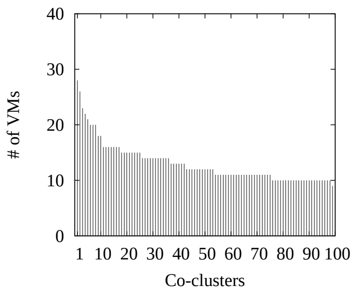
$$overlap(\mathbb{C}_1, \mathbb{C}_2) = \frac{|\{(i,j) : i \in I_1 \bigcap I_2, j \in J_1 \bigcap J_2\}|}{min\{|I_1| \times |J_1|, |I_2| \times |J_2|\}}. \quad (2)$$

If $overlap(\mathbb{C}_1, \mathbb{C}_2)$ is higher than 90%, we remove the smaller co-cluster, $\mathbb{C}_r$, $r = \arg\min_{r \in \{1,2\}} |I_r| \times |J_r|$.

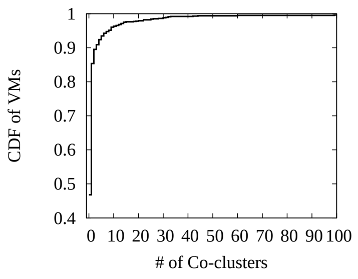
2) Validation of Co-clusters: We validate our co-clustering algorithm using the CPU utilization data of 1212 VMs collected from a single customer in the studied cloud environment. The data spans 400 15-minute intervals. The consistency threshold θ is set as 90%. Since the co-clusters with very small size are of less interest, we eliminate any co-cluster that contains no more than 10 VMs or spans no longer than 20 15-minute time intervals. Since we are more interested in VMs under high CPU utilization in practice, we tune our co-clustering method to consider only the three highest workload levels on each VM (i.e., level 3, 4 and 5). With that, our algorithm find 98 co-clusters in this data set.

We first look at the coverage of the identified co-clusters. Fig. 3(a) shows the number of VMs covered by each of the 98 co-clusters. We see that the largest number of VMs contained in a co-cluster is 28, and on average, a co-cluster contains 15 VMs. Fig. 3(c) shows the time span of each co-cluster. The largest number of time intervals spanned by a co-cluster is 302, whereas 90% of the co-clusters span at least 100 intervals or 25 hours within the 400 intervals. We also show the distribution of the VMs in different numbers of co-clusters in Fig. 3(b). The x-axis is the number of co-clusters spanned by a VM and the y-axis represents the cumulative distribution function (CDF) of this number. It can be observed that 46% of VMs did not have high workload for a significant amount of time, hence are not included in any co-cluster. Furthermore, 40% the VMs belong to only one co-cluster, which indicates that the overlap between co-clusters in terms of VMs is quite limited.

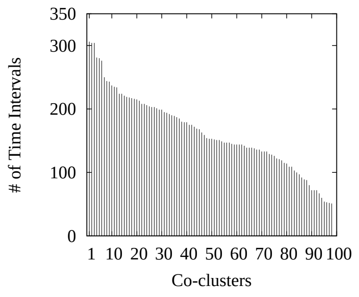
To visually study the identified co-clusters, we plot in Fig. 4(a) the discretized workload levels for all VMs during the entire 400 intervals, using a color-scale map. In Fig. 4(b), we show the corresponding map for the VMs and time intervals captured by the 98 co-clusters. It can be observed that Fig. 4(b) contains most of the higher workload levels of Fig. 4(a). This confirms that our co-clustering method is effective in capturing



(a) # of VMs spanned

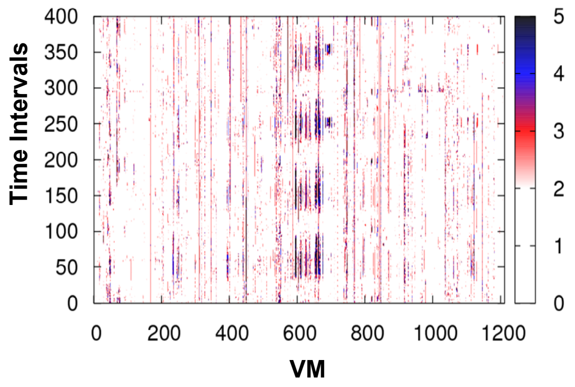


(b) VM distribution in co-clusters

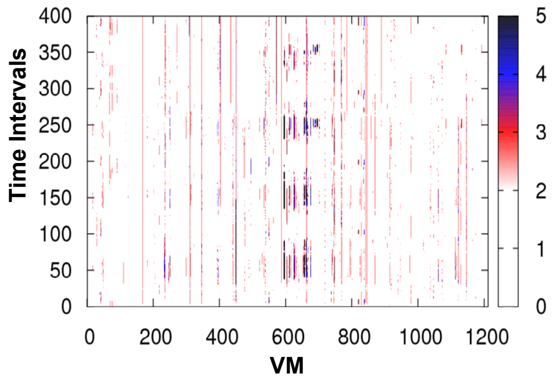


(c) # of time intervals spanned

Fig. 3. VMs and high-workload time intervals spanned by co-clusters



(a) Original workload



(b) Workload captured by co-clusters

Fig. 4. VM with high workloads: overall and those captured by co-clusters

the most prominent workload patterns, while identifying the groups of VMs with correlated behavior.

IV. Workload Prediction

Through co-clustering, we identified groups of VMs that have certain workload patterns. In this section, we explore the temporal correlations in these co-clusters, and develop a Hidden Markov Model that utilizes such temporal correlations to predict the existence of different co-clusters and the workload levels of individual VMs in them.

A. Predictable Co-clusters

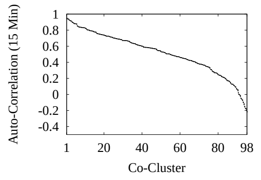
By analyzing the autocorrelation of individual workload time series, we find that its temporal correlation is the strongest at the minimum time granularity of 15 minutes [19]. Therefore, we use a 15-minute prediction interval in our analysis. We also note that not all co-clusters discovered have the same predictability. This is because a co-cluster consists of a number of VMs, of which the predictability of workload varies. As a result, we need to determine, among all co-clusters discovered, the ones that are more predictable.

Intuitively, a co-cluster is predictable because either (1) its workload pattern is consistent over time, or (2) it shows correlated (but delayed) behavior with other co-cluster(s).

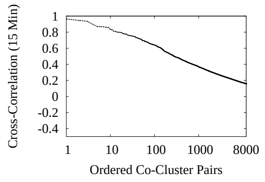
The first scenario can be identified by measuring autocorrelation function of a time series that represents the existence of a co-cluster at each observed time interval, $C = \{c_1, c_2, \ldots, c_n\}$, where $c_i (i = 1, 2, ..., n)$ is an indicator variable representing whether this co-cluster exists at time interval i. We measure its autocorrelation function as

$$R_C = \sum_{t=1}^{n-1} [(c_t - \bar{c})(c_{t+1} - \bar{c})]/(n\sigma_c^2). \quad (3)$$

Here $\bar{c}$ and σ_c represent the mean and standard deviation of time series C, respectively. Note that, here we consider time lag 1, which represents one prediction interval of 15 minutes. In Fig. 5(a), we plot R_C for the 98 co-clusters discovered from the 1212 VMs previously. We can see that a significant portion of them have fairly large autocorrelation.



(a) autocorrelation



(b) Cross correlation

Fig. 5. Determination of predictable co-clusters

The second scenario can be identified by measuring the following time-lagged cross-correlation function between two time series. Let $C_1 = \{c_{1,1}, c_{1,2}, \ldots, c_{1,n}\}$ and $C_2 = \{c_{2,1}, c_{2,2}, \ldots, c_{2,n}\}$ be the two indicator time series representing the existence of co-clusters C_1 and C_2 during time interval 1 to n, the time-lagged cross-correlation between the ordered pair (C_1, C_2) is

$$R_{C_1,C_2} = \sum_{t=1}^{n-1}[(c_{1,t} - \bar{c_1})(c_{2,t+1} - \bar{c_2})]/(n\sigma_1\sigma_2), \quad (4)$$

where $\bar{c_1}, \bar{c_2}$ and σ_1, σ_2 are the means and standard deviations of C_1, C_2, respectively. Note that R_{C_1,C_2} is not the same as R_{C_2,C_1}, as R_{C_1,C_2} measures the likelihood of seeing co-cluster C_2 in the next time interval following the appearance of C_1 in the current interval. Fig. 5(b) shows the cross-correlation for all ordered pairs of the 98 co-clusters. We notice there are also a significant number of co-cluster pairs that have strong cross correlation, hence can be used in prediction.

Combining the two scenarios, we define *predictable co-cluster* as follows. We select an empirical threshold of $\gamma = 0.4$, and consider a co-cluster C as predictable, if its autocorrelation $R_C > \gamma$, or if there exists at least one other co-cluster C', so that the time-lagged cross correlation $R_{C',C} > \gamma$ [1]. In other words, a predictable co-cluster shows relatively strong temporal correlation either by itself or with some other co-clusters observed prior to its appearance. We denote the second scenario by $C' \rightarrow C$. We call C' a *predicting co-cluster*, which may or may not be a predictable co-cluster itself.

From the 98 co-clusters discovered previously, we find 65 are predictable. Among the 65 predictable co-clusters, 51 are self-predictable, and 23 are deemed predictable due to cross-correlation with other co-clusters, while 9 are both. In addition, there are 12 co-clusters that are purely serving as predicting co-clusters but are not predictable themselves. We next build a prediction model using these 77 co-clusters.

B. Prediction Model

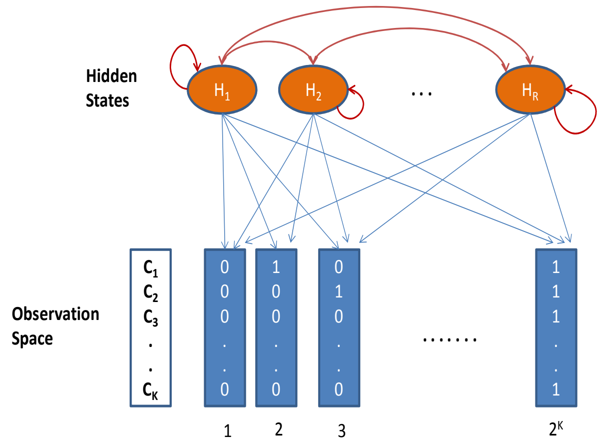


Fig. 6. Model for predicting the appearence of a set of co-clusters

[1]The selection of γ represents a trade-off between prediction accuracy and coverage of co-clusters. Limited by space, we omit this discussion here. Detailed discussions can be found in [19].

In order to predict workload changes for those predictable co-clusters, we divide them into *prediction groups*. A prediction group is a self-contained set of co-clusters that are either self-predictable or predictive of other co-clusters in the same set. In other words, each prediction group only consists of one or multiple predictable co-clusters and their corresponding predicting co-clusters. A prediction group represents a number of VMs, of which the workload is predictable through historical observations within this group. Separate prediction models will be developed for different prediction groups.

We model the workload variations across all co-clusters in a prediction group as a continuous-time Markov process. Each Markov state represents a specific type of application behavior, hence corresponds to the appearance of some co-clusters with certain probabilities. Using a Hidden Markov Model (HMM) [21], we define the following parameters (see Fig. 6).

- R states, denoted as $H = \{H_1, H_1, ..., H_R\}$, that represent R different application behaviors in a prediction group; the state at time t is q_t.
- 2^k distinct observations per state. Assume there are k co-clusters in the prediction group, each can either appear or not in an observation interval, our observation space has 2^k possible outcomes: $O = \{o_0, o_1, ..., o_{2^k}\}$.
- The state transition probabilities $A = \{a_{ij}\}$, where
$$a_{ij} = P\{q_{t+1} = H_j | q_t = H_i\}, H_i, H_j \in H.$$
- The observation probabilities in state j, $B = \{b_k\}$, where
$$b_j(k) = P\{o_k \text{ at } t | q_t = H_j\}, H_j \in H, o_k \in O.$$

All parameters H, O, A and B can be estimated using the expectation-maximization method (e.g., Baum-Welch algorithm [21]). With an initial training data, a set of values for these parameters can be determined off-line. As new data samples are added to the training data, we update the parameters incrementally, so that the updates can be performed fast enough for the model to be used online [21], [19].

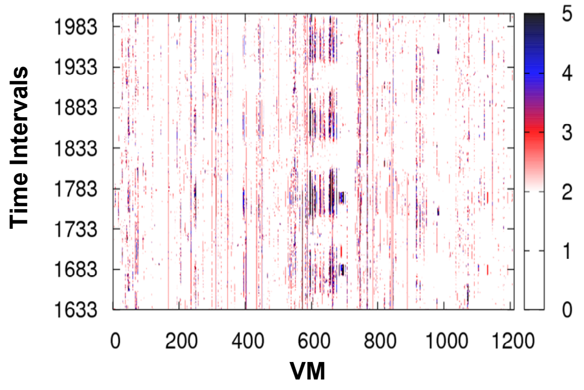
Once we estimated the state transition and observation probabilities, given the current observation o_t at time t, we calculate the probability of observing o_{t+1} at time $t+1$ as:

$$P(o_{t+1}|o_t) = \quad (5)$$
$$\sum_{H_t}\{P(H_t|o_t)\sum_{H_{t+1}} P(H_{t+1}|H_t)P(o_{t+1}|H_{t+1})\}$$

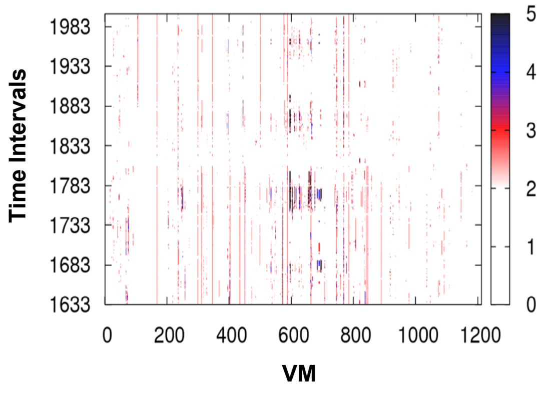
In Eq.(6), the first term, $P(H_t|O_t)$, can be determined using the Viterbi's algorithm [22], while the second and third terms are the state transition probabilities and observation probabilities that have been estimated. With this probability estimation, we then predict the most likely observation at time $t+1$ as:

$$o'_{t+1} = \arg\max_{o_i} P(o_i|o_t), o_i \in O. \quad (6)$$

Note that, the next observation represents a possible combination of which co-clusters in a prediction group would appear. Therefore, if a co-cluster is predicted to appear in the next time interval, we can predict the workload levels of all associated VMs accordingly.

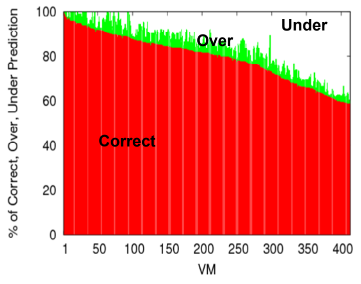


(a) Original workload

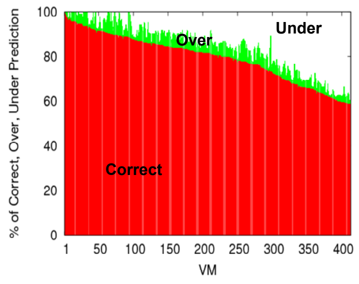


(b) Predicted workload

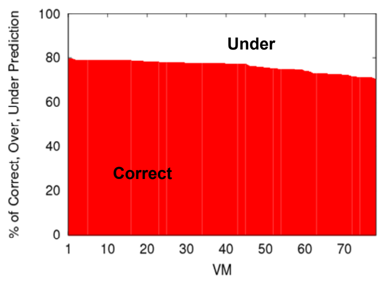
Fig. 7. Predicted vs. actual workload for 1212 VMs in 4 days



(a) Workload level 3



(b) Workload level 4



(c) Workload level 5

Fig. 8. Percentages of correct, over and under predictions on workload levels

V. Experimental Results

We conduct a series of experiments to verify the proposed workload prediction method. Our experiments are based on 21 days of CPU utilization time series collected from one enterprise customer. We first use 17 days (i.e., time interval 1 to 1632) of historical data as the initial training set to bootstrap the co-clustering algorithm described in Section III and to derive the HMM-based predictor introduced in Section IV. Using this predictor, we make predictions of workload level changes for all VMs in the predictable co-clusters, for every 15-minute interval in the next 4 days (i.e., time intervals 1633-2016). After each round of prediction, we add the data from the current interval into the training set, and incrementally update the predictor, which will then be used in the next round.

A. Effectiveness of Prediction Method

We first evaluate the overall effectiveness of our method. We apply the co-clustering algorithm and the prediction model to 1212 VMs from customer A. As shown in Table I, our method identified 98 co-clusters from them during this time period. This represents 48% of all VM-intervals observed. Out of these 98 co-clusters, 65 were considered predictable using the filtering mechanism described in Section IV-A. These 65 predictable co-clusters contains 434 VMs, or 36% of all VMs monitored. During the 384 time intervals (or 4 days) we made prediction, there are a total of $55,243$ instances in which some VM had high workload level (i.e., 3, 4 or 5). Our prediction model can be applied to predict $32,216$ or 59% of these VM-intervals.

TABLE I
STATISTICS OF PREDICTION FOR CUSTOMER A

	Count	Percentage
Co-clusters	98	48% of VM-Intervals
Predictable Co-clusters	65	41% of VM-Intervals
Predicted VMs	434	36% of VMs
Predicted VM-Interval's	32,216	59% of high workload

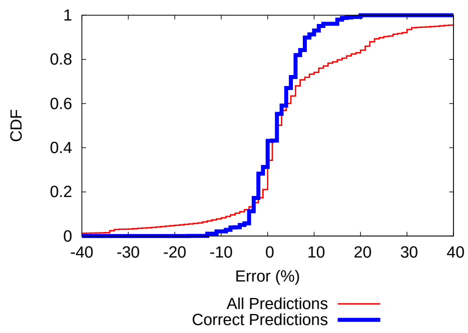


Fig. 9. CDF of prediction error with respect to actual CPU utilization

To better assess the overall effectiveness, we plot in Fig. 7(a) and 7(b), respectively, the actual and predicted workloads for each VM during the 4-day period in our experiment. We can see that most of the high workload instances were correctly predicted by our method.

To evaluate prediction accuracy, we consider the following three scenarios: If a VM's workload level is correctly predicted by our model, we consider it a *correct prediction*; if the predicted workload level is lower than the actual, we consider it an *under-prediction*; otherwise, we considered it an *over-prediction*. In Fig. 8, we plot percentages of correct, under- and over-prediction for each VM in the predictable co-clusters over all 384 prediction intervals, while the actual workload level is 3, 4 or 5, respectively. It can be observed that, the overall prediction accuracy is ranging from 60% to 95% for VMs at workload level 3, 80% to 85% for VMs at workload level 4. and 75% to 80% for VMs at workload level 5. Note that for VMs at workload level 5, there are no cases of over-prediction for obvious reasons.

The predictions generated by our model are discretized workload levels, each of which represents a normal distribution of CPU utilization percentage [19]. One can either explore the statistics of the predicted distributions or simply predict CPU utilization as the mean of the predicted distribution. For the latter, we compare the predicted and original CPU utilizations in Fig. 9. The x-axis represents the difference (in absolute percentage) between the predicted and actual utilizations. The CDF of this prediction error is shown along the y-axis for all predicted VM-intervals, as well as for those VM-intervals with correctly predicted workload levels. It can be observed that, 66% of the all predicted VM-intervals and 91% of the correctly predicted VM-intervals lie within $\pm10\%$ of the predicted (mean) values. This indicates that our prediction has a fairly narrow error margin, with the number of discretized workload levels only set to 5.

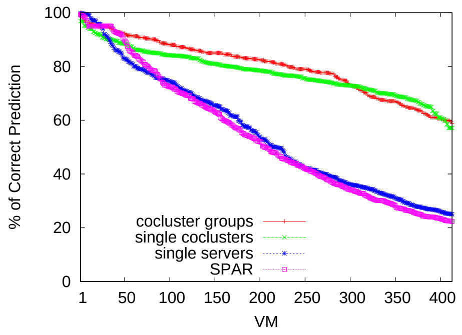


Fig. 10. Single and multiple time-series based predictions

B. Comparison with Single Time-Series Prediction

A key benefit of the proposed method is that it is based on multiple time series and explores the cross-correlation among them through the discovery of co-clusters and prediction groups. To verify this benefit, we compare the prediction accuracy of our technique with two other benchmark methods and one state-of-the art algorithm proposed in [18]. The first benchmark is single time series (VM) prediction. In this method, we treat each VM as a "co-cluster" in our algorithm, and build HMM for each predictable co-cluster/VM. The second benchmark is single co-cluster prediction. In this method, we discover the co-cluster as in the proposed method, but do not form prediction groups. That is, we build HMM for each self-predictable co-cluter, without considering other co-clusters that might be a predicting co-cluster for it. We also compare our prediction technique against the SPAR model [18], which is *single time-series based prediction method* considering both periodic patterns and local adjustments. Fig. 10 shows the percentage of correct predictions for these four methods for the same set of VMs. Compared to the single-VM prediction approach and the SPAR model, our co-clustering-based method can improve the prediction accuracy from 55% to 73%. By exploiting the cross-correlation between co-clusters, we can further boost the accuracy to 79%.

C. Feasibility of Online Prediction

A typical application of our workload prediction method is to use it to forecast workload for the "hot spots" in a cloud. To make such application feasible, we need to ensure that the model training procedure can be executed fast enough, so that predictions can be made before actual workload changes happen. In Table II, we show the average running time of our co-clustering and HMM learning algorithms. It is worth noting that the initial steps of parameter learning from the historical training data require most of the time. In particular, it requires multiple iterations for the Baum-Welch algorithm to converge, when we derive the HMM parameters. However, co-clustering and initial learning steps can be performed off-line. After the initial steps, the identified co-clusters do not need to be updated in each round of prediction, and the HMM for prediction can be incrementally updated when new data are added to the training data set. On average, the online parameter update and prediction can be finished within 96 seconds for all 35 predictable co-cluster groups (2.7 seconds per group). Thus, it is quite feasible to apply our method for online prediction of workload changes given the workload monitoring data is collected every 15 minutes.

TABLE II
RUNNING TIME OF PREDICTION ALGORITHM

Step	Running Time (seconds)
Co-clustering	604
Initial HMM learning	22,730
Incremental HMM learning	2.7 per group

VI. Related Work

Server workload characterization and prediction have been studied extensively in the past. However, most existing works have a somewhat different focus or use a different approach. For instance, theoretical models [9], [3] have been developed to generate representative workload traces. Focusing on specific application environment, Arlitt *et al.* [7] studied the workload characteristics on Web servers, while Cherkasova *et al.* [8] conducted a study on media servers. More recently, Gmach *et al.* studied the workload for data center applications [12], and Mishra *et al.* analyzed workload in a cloud environment [11]. Different from these works, which focus on workload modeling at individual server level, our study focuses on understanding the correlated workload patterns within groups of VMs that result from application dependencies.

Also related to this study are the works on capacity management and virtual server placement [23], [24], [25], [26], [27], [28], which typically employ some workload modeling and prediction techniques. For example, Bobroff *et al.* [15] used regression models to forecast workload variations, in order to dynamically place virtual machines. Verma *et al.* [16] proposed to consolidate servers using correlation or peak cluster based placement. A trace-based workload forecasting method was used in [17] for capacity management. These approaches reply on the statistics (e.g., percentiles, peaks, etc.) of individual server or application's workload to predict future capacity demand. In contrast, our approach explores cross-VM workload correlations, therefore has better prediction accuracy.

Compared to the existing works, our study provides an application-agnostic method for cloud provider to better manage its resources. For instance, cloud provider often faces the challenges of resource consolidations [27], [29]. Our workload characterization method can help understand the scope and magnitude of workload variations, hence provides guidance on how consolidation should be performed. Other applications include elastic resource provisioning: a provider can use our model to predict how workload patterns will change, and dynamically scale the resources to meet the demand [26], [30].

VII. Conclusion

In this paper, we introduced a new method of characterizing and predicting workload in a cloud environment, when complete application configurations on customers' VMs are unavailable to the cloud providers. Our method discovers and leverages repeatable workload patterns within groups of VMs that belong to a cloud customer. We developed a co-clustering technique for identifying such VM groups and the common workload patterns. Based on the co-clusters discovered, we further designed an HMM-based method to capture the temporal correlations and to predict the changes of workload pattern. Compared to the traditional method of predicting workload changes at individual sever level, this method showed significantly higher prediction accuracy. In future work, it will be interesting to combine the periodic daily patterns observed for individual servers with our current prediction model.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] M. Calzarossa and G. Serazzi, "Workload characterization: A survey," *Proc. of the IEEE*, vol. 81, no. 8, pp. 1136–1150, 1993.

[3] W. Cirne and F. Berman, "A comprehensive model of the supercomputer workload," in *Proc. of IEEE Int'l Workshop on Workload Characterization*, 2001.

[4] M. Crovella, "Performance evaluation with heavy tailed distributions," *Lecture Notes in Computer Science*, vol. 2221, pp. 1–10, 2001.

[5] P. Barford and M. Crovella, "Generating representative web workloads for network and server performance evaluation," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 1, pp. 151–160, 1998.

[6] A. B. Downey and D. G. Feitelson, "The elusive goal of workload characterization," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 4, pp. 14–29, 1999.

[7] M. Arlitt and C. Williamson, "Web server workload characterization: The search for invariants," in *Proc. of SIGMETRICS*, May 1996.

[8] L. Cherkasova and M. Gupta, "Characterizing locality, evolution, and life span of accesses in enterprise media server workloads," in *NOSSDAV*, 2002.

[9] B. Urganokar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi, "An analytical model for multi-tier internet services and its applications," in *SIGMETRICS*, 2005.

[10] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, "Towards understanding cloud performance tradeoffs using statistical workload analysis and replay," EECS, UC Berkeley, Tech. Rep., 2010.

[11] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads: Insights from google computer clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 34–41, 2010.

[12] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *IEEE Int'l Symp. on Workload Characterization*, 2007.

[13] A. Ganapathi, Y. Chen, A. Fox, R. Katz, and D. Patterson, "Statistics-driven workload modeling for the cloud," in *Int'l Workshop on Self managing Database Systems*, 2010.

[14] S. Govindan, J. Choi, B. Urgaonkar, A. Sivasubramaniam, and A. Baldini, "Statistical profiling-based techniques for effective power provisioning in data centers," in *EuroSys*, 2009.

[15] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *IFIP/IEEE IM*, 2007.

[16] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *USENIX ATC*, 2009.

[17] J. Rolia, L. Cherkasova, M. Arlitt, and A. Andrzejak, "A capacity management service for resource pools," in *Proc. of ACM Workshop on Software and Performance*, 2005.

[18] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-aware server provisioning and load dispatching for connection-intensive internet services," in *NSDI*, 2008, pp. 337–350.

[19] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," Tech. Rep., 2011. [Online]. Available: http://www.cs.ucsb.edu/∼arijitkhan/Papers/multiple_timeseries_prediction.pdf

[20] G. Li, Q. Ma, H. Tang, A. H. Paterson, and Y. Xu, "Qubic: a qualitative biclustering algorithm for analyses of gene expression data." *Nucleic acids research*, vol. 37, no. 15, 2009.

[21] R. Paroli and L. Spezia, "Parameter estimation of gaussian hidden markov models when missing observations occur," *Metron - Int'l Journal of Statistics*, pp. 163–179, 2002.

[22] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *IEEE*, 1989, pp. 257–286.

[23] B. Urgaonkar, P. Shenoy, and T. Roscoe, "Resource overbooking and application profiling in shared hosting platforms," in *OSDI*, 2002.

[24] A. Chandra, W. Gong, and P. Shenoy, "Dynamic resource allocation for shared data centers using online measurements," in *IWQoS*, 2004.

[25] W. Zheng, R. Bianchini, G. J. Janakiranman, J. R. Santos, and Y. Turner, "Justrunit: Experiment-based management of virtualized data centers," in *Usenix ATC*, 2009.

[26] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *IEEE CNSM*, 2010.

[27] Z. Gong and X. Gu, "Pac: Pattern-driven application consolidation for efficient cloud computing," in *ACM CCS*, 2010.

[28] J. Dejun, G. Pierre, and C.-H. Chi, "Resource provisioning of web applications in heterogeneous clouds," in *WebApps*, 2011.

[29] U. Sharma, P. Shenoy, S. Sahu, and A. Shaikh, "Kingfisher: Cost-aware elasticity in the cloud," in *IEEE Infocom Mini-Conference*, 2011.

[30] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal, and T. Wood, "Agile dynamic provisioning of multi-tier internet applications," *ACM Trans. Autonomous and Adaptive Systems*, vol. 3, no. 1, 2008.