2011 31st International Conference on Distributed Computing Systems Workshops

Exploiting Resource Usage Patterns for Better Utilization Prediction

Jian Tan, Parijat Dube, Xiaoqiao Meng, Li Zhang
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
{*tanji,pdube,xmeng,zhangli*}@us.ibm.com

***Abstract*—Understanding the resource utilization in computing clouds is critical for efficient resource planning and better operational performance. In this paper, we propose two ways, from microscopic and macroscopic perspectives, to predict the resource consumption for data centers by statistically characterizing resource usage patterns. The first approach focuses on the usage prediction for a specific node. Compared to the basic method of calibrating AR models for CPU usages separately, we find that using both CPU and memory usage data can improve the forecasting performance. The second approach is based on Principal Component Analysis (PCA) to identify resource usage patterns across different nodes. Using the identified patterns, we can reduce the number of parameters for predicting the resource usage on multiple nodes. In addition, using the principal components obtained from PCA, we propose an optimization framework to optimally consolidate VMs into a number of physical servers and in the meanwhile reduce the resource usage variability. The evaluation of the proposed approaches is based on traces collected from a production cloud environment.**

I. Introduction

In modern compute cloud, consolidating Virtual Machines (VMs) is the key technique to improve the overall resource usage and thus to achieve economy of scale. Most VM consolidation schemes [1][2][3][4][5] need to estimate the future demand for each VM. Such an estimate is usually made from a characterization of the resource usage patterns based on historical workload traces. In the VM consolidation phase, the estimated demand becomes the basis for determining VM placement.

Identifying the resource usage patterns across a data center can help resource prediction and VM consolidation. In this paper, we propose two ways, from microscopic and macroscopic perspectives, to predict the resource consumption for data centers. The first approach focuses on the usage prediction for a specific node. Compared to the basic method of calibrating CPU usages by AR models separately, we find that using both CPU and memory usage data can improve the forecasting performance.

The second approach is based on the observation that the workload running on a large number of nodes is likely to be caused by a smaller number of users or jobs. For example, the CPU, I/O, memory usage on the slave nodes for a map/reduce Hadoop program may be the resultant of the map tasks running in parallel; A work flow spanning multiple stages may involve computation on different nodes in each stage. In all these scenarios, the compound effect is that the resource usage can have spatial and temporal correlations over multiple nodes. These independent jobs and work flows, which can not be observed directly from the collected data, if identified properly, can help to reduce the number of parameters needed for fitting the forecasting model, and improve the robustness of the prediction result. In view of these observations, we propose a PCA based approach to forecast CPU and memory usages using the data obtained from a commercial data center. We show that our PCA based approach can dramatically reduce the number of model parameters and the time complexity for computation, while predicting the CPU usages on a large number of nodes with errors comparable to the approach of treating the nodes separately.

In addition, the identified principal components from PCA can be used to better consolidate virtual machines on a good number of physical servers. Along this line, most existing work formulates it as a bin packing problem, which assumes that the required CPU, memory and network for each virtual machine remains constant over the whole operation period. As a result, the consolidation process basically is to place the given virtual machines into the smallest number of physical servers, where each virtual machine is characterized by a constant vector denoting its resource usage profile, and the physical servers can be viewed as multi-dimensional containers. However, these approaches ignore the fact that the workload fluctuates in a stochastic fashion and, due to the dependency along work flows that may span over multiple nodes, the resource usages on different nodes inherit temporal and spatial correlations. We propose to place VMs using variance reduction, based on the intuition that placing two virtual machines with negatively correlated CPU usages can reduce the usage variability on the physical server. This technique can provide a smoother resource usage demand on the physical servers of a data center.

Capacity planning in compute cloud has been a hot topic in the literature [6][7][8]. A primary approach in these work is first to characterize resource usage, then

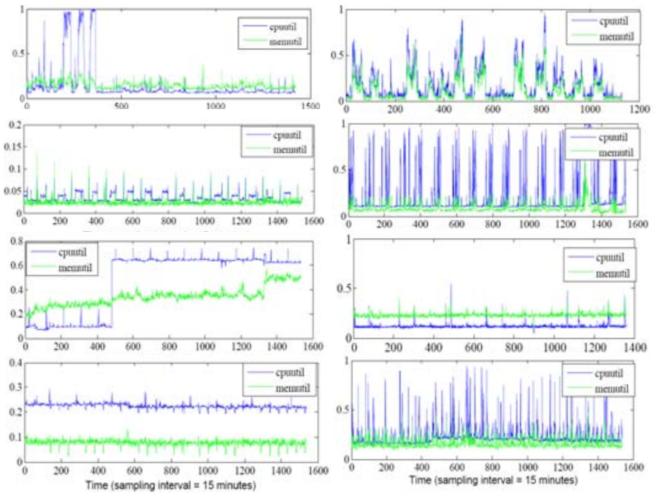

Fig. 1. CPU and memory utilization patterns across different VMs.

to determine VM consolidation by predicting the future resource usage based on the characterization. [6] applied standard time series techniques to extract the periodicity and trend from CPU utilization; based on the prediction, it addressed the VM placement by consolidating VMs with complementary workload patterns. [7] proposed a three-stage method to identify workload demand patterns, then applied the prediction to VM placement. [8] did not explicitly forecast workload. However, it still assumed historical workload pattern would continue to hold. Compared to our paper, all these work make prediction on per node basis; they do not study the dimension of cross-VM patterns.

II. Model description and data preprocessing

In this paper, we study a large trace collected from real-world clouds hosted by IBM Global Services. These clouds reside in different countries and are used to host workloads from a large number of enterprise customers. Virtual machines run an agent program which periodically collects the server performance data and sends to a centralized data warehouse. Our study is based on a subset data in a three-month period of early 2010. This subset data involves 1323 physical servers and 15883 virtual machines hosted on them. The performance data on each virtual machine includes the utilization ratio of CPU, memory and storage, disk I/O rate, and network traffic rate. In this paper, we limit our study to CPU and memory utilization ratio because they are by far the most important resource concerns.

A. Resource Usage Patterns and Motivation

Figure 1 shows the CPU and memory utilization data for different VMs. The data is sampled every 15 minutes and is collected over several weeks. Observe that these VMs have different levels of memory and CPU utilization. Some have higher memory utilization compared to CPU while

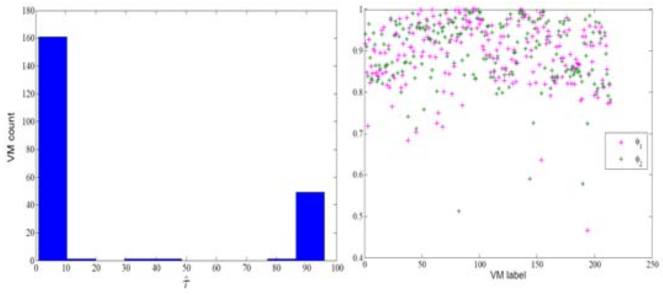

Fig. 2. $\hat{\tau}$, $\phi_1(\hat{\tau})$, $\phi_2(\hat{\tau})$ for 214 VMs.

others show opposite behavior. Some exhibit periodicity in time while others follow trends. Some have regular load patterns while others change loads at different time scales. In addition, there is a high correlation between CPU and memory utilization within a VM, and the resource usages across some nodes are highly correlated. These observations motivate us to exploit the dependencies among CPU and memory usage within VMs as well as the correlations across VMs for improving resource usage prediction and hence efficiency of VM provisioning.

B. Data Preprocessing

Consider a data center of N nodes, where $k \geq 1$ different resource usages are measured on each node. In this paper, we only consider CPU and memory ($k = 2$). In addition, we assume that the time t is slotted and finite, $t = 1, 2, 3, \cdots, M$(for example, we measure the resource utilization at fixed intervals). Let $\mathbf{Y}_j(t) = (Y_j^1(t), Y_j^2(t), ..., Y_j^k(t))$ be the resource usage on node j, $0 \leq j \leq N-1$ at time t. Let $\{\mathbf{X}_j(t)\}$ be the stationary time series obtained after preprocessing the observed nonstationary time series $\{\mathbf{Y}_j(t)\}$.

The CPU and memory utilization time series for different VMs exhibit periodicity ranging from several minutes/hours to a day. Thus to remove this periodicity we apply ARAR algorithm [9] and obtain a stationary time series. Since for each VM the CPU and memory utilization data is collected every fifteen minutes, for each $\tau = 1, 2, \cdots, 100$, we find the value $\hat{\Phi}(\tau)$ of $\Phi(\tau)$ that minimizes

$$ERR(\Phi(\tau), \tau) = \frac{\sum_{t=\tau+1}^{M}\left((Y_1(t) - \phi_1 Y_1(t-\tau))^2 + (Y_2(t) - \phi_2 Y_2(t-\tau))^2\right)}{\sum_{t=\tau+1}^{M}(Y_1(t)^2 + Y_2(t)^2)},$$

and choose the lag $\hat{\tau}$ to be the value of τ that minimizes ERR$(\hat{\Phi}(\tau), \tau)$. Then, let

$$\mathbf{X}_j(t) = \mathbf{Y}_j(t) - \hat{\Phi}(\hat{\tau})\mathbf{Y}_j(t - \hat{\tau}). \quad (1)$$

If the sample mean of $\mathbf{X}_j(t)$ is not equal to zero, it can be subtracted from $\mathbf{X}_j(t)$ to get a zero mean stationary time series $\mathbf{X}_j^\circ(t)$.

Fig. 2 shows the frequency distribution of $\hat{\tau}$ and corresponding values of ϕ_1 and ϕ_2 for 214 nodes from our data set. The $\hat{\tau}$ values are concentrated mainly at some low values (1,2,4) and 96. The value $\hat{\tau} = 4$ characterize nodes with hourly periodicity while $\hat{\tau} = 96$ is for nodes with daily periodicity. The high values of ϕ_i, $i = 1, 2$ for almost all VMs establishes very regular and repetitive patterns for resource usage for different VMs. In next section we develop AR models for time series $\{\mathbf{X}_j(t)\}$; the estimates for original observations $\{\mathbf{Y}_j(t)\}$ can then be determined from $\{\mathbf{X}_j(t)\}$ using (1).

III. Autoregressive Models for a VM

Let $\mathbf{X}_j^\circ(t) = (X_j^1(t), X_j^2(t))$ denote the stationary multivariate time series obtain by preprocessing $\mathbf{Y}_j(t)$, with $X_j^1(t)$ associated with the CPU usage and $X_j^2(t)$ the memory usage. We next apply standard AR models to characterize $\mathbf{X}_j^\circ(t)$. We apply two different approaches. The first approach treats each series separately and fits a single-dimensional AR(1) model to it. The second approach exploits intra-VM dependencies between CPU and memory usage to fit a multi-dimensional AR model, denoted by AR-M(1). It relies on both CPU and memory time series of a VM for prediction, using

$$\begin{aligned} X_j^1(t) &= b_{j,11}X_j^1(t-1) + b_{j,12}X_j^2(t-1) + \beta_j^1\epsilon_j^1(t) \\ X_j^2(t) &= b_{j,21}X_j^1(t-1) + b_{j,22}X_j^2(t-1) + \beta_j^2\epsilon_j^2(t), \end{aligned}$$

where $\epsilon_j^1(t), \epsilon_j^2(t)$ are white noises ($WN(0,\sigma^2)$). We apply Yule-Walker method [9] to estimate the coefficients of the model. After obtaining $\hat{\mathbf{X}}_j(t)$ as estimate of $\mathbf{X}_j^\circ(t)$ we can compute $\hat{\mathbf{Y}}_j(t)$ as estimate of $\mathbf{Y}_j(t)$. To compare goodness of fit we calculate Root Mean Square Error (RMSE) between the predicted and observed data for both CPU and memory utilization.

$$\text{RMSE}_j^i = \sqrt{\frac{\sum_{t=1}^{M}(Y_j^i(t) - \hat{Y}_j^i(t))^2}{M}}. \tag{2}$$

We next define two measures for normalized Root Mean Square Error. RMSE normalized by sample mean of observed data will be called Mean Normalized RMSE (MNRMSE) and RMSE normalized by sample standard deviation of observed data will be called Range Normalized RMSE (RNRMSE).

$$\text{MNRMSE}_{j,1}^i = \frac{\text{RMSE}_j^i}{\text{mean}(Y_j^i)}, \tag{3}$$

$$\text{RNRMSE}_{j,2}^i = \frac{\text{RMSE}_j^i}{\text{stddev}(Y_j^i)}. \tag{4}$$

To test different approaches we build prediction models and quantify different error measures as defined above. We applied AR(1) and AR-M(1) models to 214 VMs. Figure 3 compares the empirical cumulative distribution function (ECDF) of RMSE for these 214 VMs for CPU and memory utilization. Observe that the ECDF of AR(1) is always dominated by the ECDF of AR-M(1). Similar ordering among ECDFs is also observed for MNRMSE and

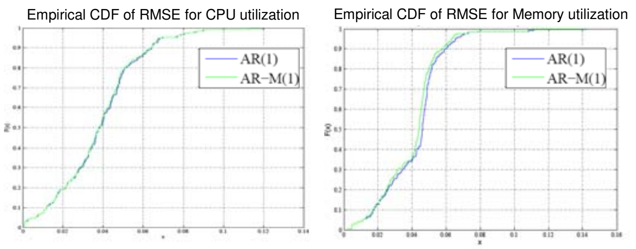

Fig. 3. Empirical CDF of RMSE for CPU and memory utilization.

Metric	CPU Utilization			Memory Utilization		
	AR(1)	AR(2)	AR-M(1)	AR(1)	AR(2)	AR-M(1)
RMSE	0.033	0.032	0.032	0.035	0.034	0.033
MNRMSE	0.418	0.411	0.413	0.409	0.402	0.393
RNRMSE	0.777	0.764	0.771	0.787	0.773	0.762

TABLE I
Average value of error metrics with AR(1), AR(2) and AR-M(1).

RNRMSE metrics. This establishes that AR-M(1) models are statistically better than AR(1) models and hence better predictors of resource usage. We increased the data set to about 500 VMs and again observed similar orderings between various ECDFs. Similar generalizations between AR(p), AR-M(p), $p > 1$ can probably be established. Table I shows the average values of RMSE, MNRMSE and RNRMSE for the three prediction models for experiments with 500 VMs. We also compared AR(2) models with AR(1) and AR-M(1). Table II shows the corresponding relative decrease in these error metrics with AR(2) and AR-M(1) compared to AR(1). We note that for CPU utilization prediction AR(2) is a better choice than AR-M(1) whereas for memory utilization prediction AR-M(1) has about two times more error reduction than AR(2).

Metric	CPU Utilization		Memory Utilization	
	AR(2)	AR-M(1)	AR(2)	AR-M(1)
RMSE	1.68%	1.01%	1.78%	3.84%
MNRMSE	1.64%	1.07%	1.64%	3.88%
RNRMSE	1.67%	0.87%	1.77%	3.23%

TABLE II
Relative decrease in error metrics with AR(2) and AR-M(1) models compared to AR(1).

IV. PCA based prediction across nodes

In the preceding section, we have investigated the problem of predicting the resource usage on one specific node. However, in many situations, we need to forecast the usage utilization for large scale distributed computing systems, e.g., computing cloud and data centers. This type of prediction involves possibly a large number of nodes. In this section, we focus on characterizing CPU usage on nodes across a data center.

Let $\mathbf{Z}(t)$ be a $N \times 1$ vector that represents the measurements of CPU usage at time t, $\mathbf{Z}(t) = \left(Y_0^1(t), Y_1^1(t), \ldots, Y_{N-1}^1(t)\right)^T$. In principle, one can use a multivariate AR model with a $N \times N$ matrix to forecast this time series. However, when the data center contains a large number of nodes (N is very large), we may have the problem of over fitting the model. We propose to use principal component analysis to identify the resource usage patterns across the whole data center, and then conduct AR model prediction on a subset of principal components that account for most of the variability in the measurements. After forecasting resource usages for the small number of principal components, we map the predictions along these identified principal components (projected on some selected eigenvectors) back to the resource usage observation space as the prediction.

Next, we will describe the details of our approach. After measuring the system from $t = 1$ to $t = M$, we know the average of CPU usage on node j

$$\bar{Y}_j \triangleq \frac{1}{M}\sum_{t=1}^{M} Y_j^1(t).$$

Denote $\bar{Y} = \left(\bar{Y}_0, \bar{Y}_2, \cdots, \bar{Y}_{N-1}\right)^T$, and subtract the average from the real usage to obtain $Z^\circ(t) = Z(t) - \bar{Y}$. Since estimating the covariance matrix is not the focus of the paper, we choose the simplest way. Using $\{Z^\circ(t)\}_{1\le t\le M}$, we can construct a $N \times M$ matrix

$$\mathbf{Z}^\circ = \left(Z^\circ(1)\ \ Z^\circ(2)\ \ \cdots\ \ Z^\circ(M)\right),$$

and the covariance matrix $\mathbf{C}_X$ is estimated by

$$\mathbf{C}_X = \frac{1}{M-1}\mathbf{Z}^\circ \mathbf{Z}^{\circ T}. \tag{5}$$

Note that this way cannot guarantee a positive semi-definite matrix estimation. For better approaches see, e.g., [10]. Denote the eigenvalues of the matrix $\mathbf{C}_X$ by $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_N$ and the corresponding normalized eigenvectors by $\mathbb{P} = \{P_1, P_2, \cdots, P_N\}$.

Now, for a fixed $k < N$, we choose a subset $\mathbb{P}^* = \{P_1, P_2, \cdots, P_k\} \subset \mathbb{P}$, which represents the most important components. We will use these principal components to approximate $\mathbf{Z}(t)$ by constructing $\tilde{Z}(t)$ in the following form

$$\tilde{Z}(t) = \bar{Y} + \sum_{i=1}^{k} V_i(t) P_i, \tag{6}$$

where

$$V_i(t) = Z^\circ(t)^T \cdot P_i. \tag{7}$$

Using the preceding procedure we next transform the observed N number of time series $\{Y_j^1(t)\}_{0\le j\le N-1}$ into $k < N$ number of new time series $\{V_i(t)\}_{1\le i\le k}$. For the reduced number of new series, if we can provide good prediction for $\{V_i(t)\}$, then using (6) we can obtain the prediction for the resource usage of interest. We formally describe the whole process in the following algorithm.

Algorithm 1 Predict CPU usage across multiple nodes using PCA

```
Pick $0 < k < N$
1. Obtain the eigenvectors $\mathbb{P}^*$ using (5)
2. Compute the time series $\{V_i(t)\}$ according to (7) for the training data
3. Preprocess $\{V_i(t)\}$ to obtain $V_i'(t) = V_i(t) - \hat{\Phi}(\hat{\tau})V_i(t-\hat{\tau})$ using a similar approach as in (1)
4. Calibrate the AR(2) model for $1 \le i \le k$,
       $V'_i(t) = \alpha_1 V'_i(t-1) + \alpha_2 V_i'(t-2) + \epsilon(t)$
   with $\epsilon(t) \sim WN(0, \sigma_v^2)$
5. Compute $\hat{V}_i(t) = \alpha_1 V_i'(t-1) + \alpha_2 V_i'(t-2) + \hat{\Phi}(\hat{\tau})V_i(t-\hat{\tau})$ (noting that $V_i(t-\hat{\tau})$ is already known at time $t$)
6. Return the prediction $\hat{Z}(t) = \bar{Y} + \sum_{i=1}^{k} \hat{V}_i(t) P_i$
```

Next we verify our algorithm by using the measurements collected from a domain of a data center that contains 1418 virtual machine instances. We only plot the largest 500 eigenvalues of the CPU usages covariance matrix in Figure 4. As clearly shown in this figure, the eigenvalues decay very fast, and thus we can just choose the first 20 largest eigenvalues as well as the corresponding eigenvectors to characterize the CPU usages across this domain. By doing so, we can reduce the original 1418 time series to only 20 time series.

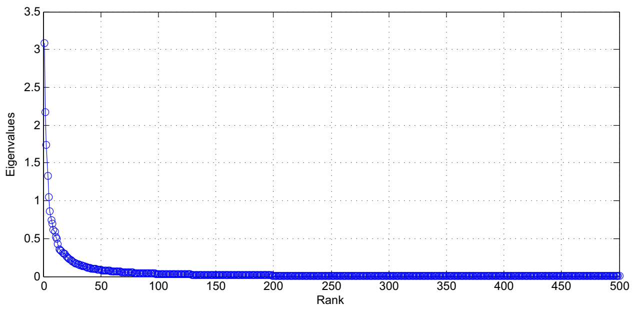

Fig. 4. Eigenvalues of the CPU usage covariance matrix

In order to ease the presentation and visualize the results, we pick 16 nodes from the collected measurements for analysis. In Figure 5 we plot the inner product of the observed CPU usage on these 16 nodes and the eigenvectors corresponding to the three largest eigenvalues.

To compare these newly constructed time series along the important principal components with the real observations, we also plot the CPU usage observed for 900×15 minutes on four nodes in Figure 6.

For forecasting the CPU usage for these 16 nodes, we choose the largest 4 eigenvalues and the corresponding eigenvectors as the principal components. In our forecasting task, we use the measurements collected at time t to predict the CPU usage on a node after 15 minutes. We plot the real measurement as well as the 15-minute prediction

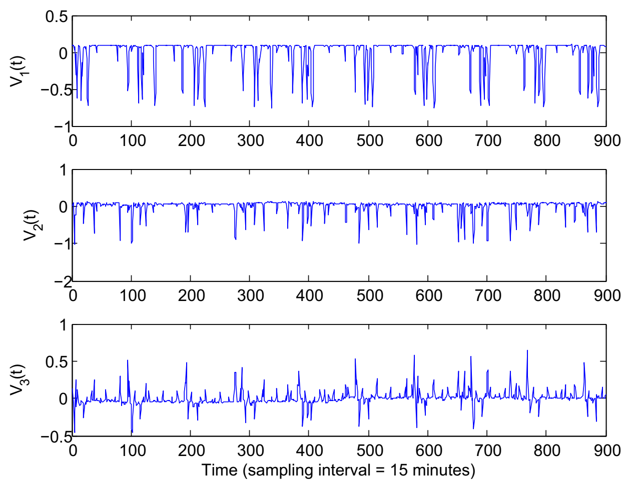

Fig. 5. The three most important principal components

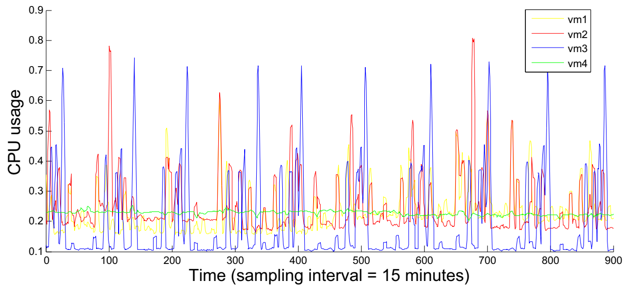

Fig. 6. CPU usage on four nodes

in Figure 7. In this figure the solid line (blue) corresponds to the real measurements and the circles (red) represents the predicted CPU usages on a node. As shown from the figure, even though we only forecast 4 time series, the prediction of CPU usages on these 16 nodes is still good. To quantify the prediction error, we introduce the following metric

$$\mathrm{ERR} = \sqrt{\sum_{t=1}^{M}\sum_{j=0}^{N-1}(Y_j^1(t) - \hat{Y}_j^1(t))^2 \Big/ \left(\sum_{t=1}^{M}\sum_{j=0}^{N-1} Y_j^1(t)^2\right)},$$

which measures the relative distance between the real CPU usage and predicted one. For the data set we used in this case study, the prediction error is equal to 31.3%.

To compare this error with the one that comes from prediction on a single virtual machine, we also run AR(2) model on the node VM 1 to fit the CPU usage time series. The prediction error on this node is equal to 32.0%, which is approximately equal to the error of using PCA based approach. From these two results, we see that, by identifying the typical resource usage patterns using PCA, one can dramatically reduce the number of model parameters and the time complexity for computation, while

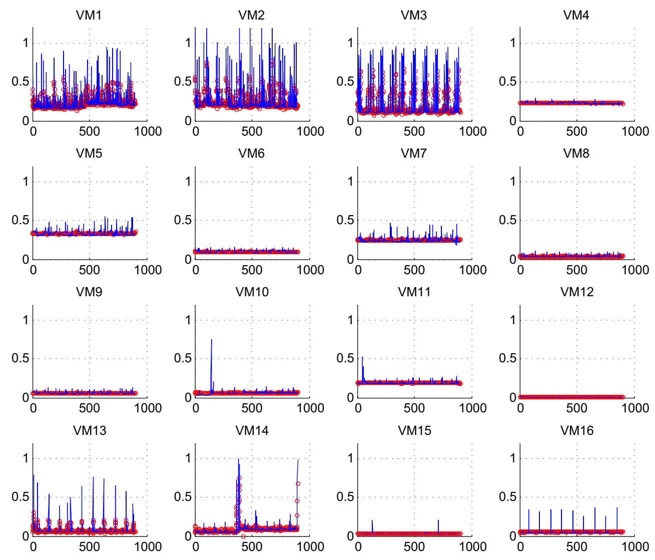

Fig. 7. CPU usage prediction on 16 nodes using 4 principal components

predicting the CPU usages on a large number of nodes with errors comparable to the approach of treating the nodes separately. The RNRMSE values from (4), for these 16 nodes are 0.7309, 0.7768, 0.6374, 0.9922, 0.9813, 0.9693, 0.9911, 0.9870, 0.9731, 0.9980, 0.9898, 0.9981, 0.9987, 0.5585, 0.9993 and 0.9992, respectively.

V. Engineering Ramifications

Using the resource usage patterns we identified in (6) from PCA, we can utilize the spatial correlations across the cluster of virtual machines to better consolidate virtual machines into physical servers. More specifically, most existing work formulates it as a bin packing problem, which assumes that the required CPU, memory and network for each virtual machine remains constant over the whole operation period. As a consequence, the consolidation process basically is to place the given virtual machines into the smallest number of physical servers, where each virtual machine is characterized by a constant vector to denote its resource usage profile, and the physical servers can be viewed as multi-dimensional containers.

However, this approach relies on the assumption that the resource usage profile can be characterized well by the mean values. In practice, the workload on each virtual machine fluctuates in a stochastic fashion. Due to the dependency along work flows that may span over multiple nodes, the resource usages on different nodes inherit the temporal and spatial correlations. Our approach is based on the intuition that placing two virtual machines with negatively correlated CPU usages can reduce the usage variability on the physical server. In Figure 8, we plot the eigenvector corresponding to the largest eigenvalue of the CPU covariance matrix obtained from the measurements used in Fig. 7. As we can see from this figure, the number of positive and negative numbers in this eigenvector are comparable.

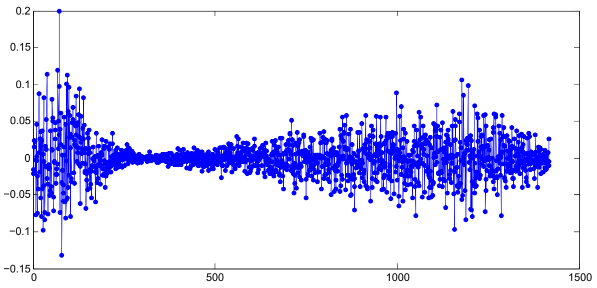

Fig. 8. Eigenvector $(p_{1,1}, p_{1,2}, \cdots, p_{1,1418})$ corresponding to the largest eigenvalue in Figure 4

To simplify the presentation, we set $k = 2$ in (6) and only focus on CPU usages. Thus, we obtain the following explicit expression

$$\hat{Z}(t) = \bar{Y} + V_1(t)\begin{pmatrix} p_{1,1} \\ p_{1,2} \\ \vdots \\ p_{1,N} \end{pmatrix} + V_2(t)\begin{pmatrix} p_{2,1} \\ p_{2,2} \\ \vdots \\ p_{2,N} \end{pmatrix}. \quad (8)$$

Now, we describe the details of our approach. First, we categorize the collected mean values (vector $\bar{Y}$) of the CPU usages into a fixed number C of classes using any clustering algorithm (e.g., K-mean clustering). Then, we compute the average CPU usages across all the nodes within each class, and assign this number to each node of the class. Using this newly obtained CPU profiles, we run any bin packing algorithm (e.g., see [11]) to consolidate all these virtual machines into a fixed number S of servers. This step computes an initial placement of the virtual machines on the physical servers. Note that all nodes within the same class are assigned with the same average CPU usage.

More specifically, after the preceding computation, we obtain a vector $(n_j^1, n_j^2, \cdots, n_j^C)$ for server j, $1 \le j \le S$, which represents that the initial placement of the virtual machines is to place n_j^i number of virtual machines from class i on server j. In addition, we also obtain two sets $\mathbb{S}_i^1$ and $\mathbb{S}_i^2$ for class i. These two sets are obtained from the two vectors P_1 and P_2 shown in (8). Specifically, if a virtual machine numbered x belongs to class i, then we add $p_{1,x}$ to $\mathbb{S}_i^1$ and $p_{2,x}$ to $\mathbb{S}_i^2$.

In the next step, we will refine the placement by utilizing the eigenvectors obtained from the covariance matrix to reduce the resource usage variances. To this end, we formulate the following problem for virtual machines in class i, $1 \le i \le C$.

Problem 1. *Given* $\mathbb{S}_i^1 = \{p_1^{1,i}, p_2^{1,i}, \cdots, p_l^{1,i}\}$, $\mathbb{S}_i^2 = \{p_1^{2,i}, p_2^{2,i}, \cdots, p_l^{2,i}\}$ *and* $(n_1^i, n_2^i, \cdots, n_S^i)$ *with* $S^i(x) \triangleq \sum_{j=1}^{x} n_j^i$, $x \ge 1$ *and* $S^i(0) \equiv 0$, *find a permutation* $(\sigma_1, \sigma_2, \cdots, \sigma_l)$ *of* $(1, 2, \cdots, l)$ *that minimizes*

$$\sum_{\tau=1}^{S} \lambda_1 \left(\sum_{j=S^i(\tau-1)}^{S^i(\tau)} p_{\sigma_j}^{1,i} \right)^2 + \sum_{\tau=1}^{S} \lambda_2 \left(\sum_{j=S^i(\tau-1)}^{S^i(\tau)} p_{\sigma_j}^{2,i} \right)^2,$$

where λ_1, λ_2 *are the two largest eigenvalues and* $l = \sum_{j=1}^{S} n_j^i$.

This optimization is related to the classic minimum sum-of-squares problem. Denote the optimal solution by $(\sigma_1^*, \sigma_2^*, \cdots, \sigma_l^*)$. Then, we place the virtual machines $\{\sigma^*_{S^i_{j-1}}, \cdots, \sigma^*_{S^i_j}\}$ on server j. Notice that the vector $(n_1^i, n_2^i, \cdots, n_S^i)$ is fixed when solving this optimization. Therefore, the number of possible permutations is equal to $l!/\prod_{j=1}^{l} n_j^i!$. After running this algorithm for each class i of virtual machines, we can place all of them into the corresponding physical servers.

VI. Conclusion

Identifying resource usage patterns across a data center can help resource prediction and VM consolidation. In this work, we propose a PCA based approach for prediction using the measurements obtained from a commercial data center. We show that this approach can dramatically reduce the number of parameters and the time complexity for computation. In addition, these identified principal components can also improve VM consolidation. Different from most of the existing works that formulate consolidation as a bin packing problem under the assumption that the resource usages remain constant, we characterize the resource usage dynamics by principal components, and propose to place VMs using variance reduction. This approach is based on the intuition that placing two virtual machines with negatively correlated usages can reduce the variability on the physical server.

References

[1] VMware Inc., "VMware Capacity Planner, http://www.vmware.com/products/capacity-planner/."
[2] IBM WebSphere CloudBurst, "http://www-01.ibm.com/software/webservers/cloudburst/."
[3] Novell PlateSpin Recon, "http://www.novell.com/products/recon/."
[4] Lanamark Suite, "http://www.lanamark.com/."
[5] S. Mehta and A. Neogi, "Recon: a tool to recommend dynamic server consolidation in multi-cluster data centers," in *NOMS*, 2008.
[6] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis, "Efficient resource provisioning in compute clouds via vm multiplexing," in *IEEE ICAC*, Washington DC, USA, 2010.
[7] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Capacity management and demand prediction for next generation data centers," 2007.
[8] A. N. S. Mehta, "Recon: A tool to recommend dynamic server consolidation in multi-cluster data centers," in *Network Operations and Management Symposium*, 2008.
[9] P. Brockwell and R. Davis, *Introduction to Time Series and Forecasting*. Springer, 1996.
[10] D. N. Politis, "Higher-order accurate, positive semi-definite estimation of large-sample covariance and spectral density matrices," *Econometric Theory*, 2010.
[11] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *Proceedings of the International Conference for the Computer Measurement Group (CMG)*, 2007.