PRACTISE: Robust Prediction of Data Center Time Series

Ji Xue*, Feng Yan*, Robert Birke†, Lydia Y. Chen†, Thomas Scherer†, and Evgenia Smirni*
* College of William and Mary
Williamsburg, VA, USA
{xuejimic,fyan,esmirni}@cs.wm.edu
†IBM Research Zurich Lab
Zurich, Switzerland
{bir,yic,tsc}@zurich.ibm.com

***Abstract*—We analyze workload traces from production data centers and focus on their VM usage patterns of CPU, memory, disk, and network bandwidth. Burstiness is a clear characteristic of many of these time series: the existence of peak loads within clear periodic patterns but also patterns that DONOT have clear periodicity. We present PRACTISE a neural network-based framework that can successfully and quite accurately predict future loads, peak loads, and their timing. Extensive experimentation using traces from IBM datacenters illustrates PRACTISE's superiority when compared with ARIMA and baseline neural network models, with average prediction errors that are three times less. Its robustness is also illustrated with respect to the prediction window that can be short-term (i.e., hours) or long-term (i.e., a week).**

I. Introduction

Effective workload characterization and prediction hold the answers to the conundrum of efficient resource allocation in distributed and scaled out systems. Being able to *accurately* predict the upcoming workload within the next time frame (which can be defined as the next 10 minutes, half hour, hours, or even a week) allows the system to make proactive decisions, rather than reactive ones. Proactive decisions can be used with superior performance in storage systems by timely warming up the cache with the working set [1], [2], especially in systems where traditional internal work (e.g., garbage collection, snapshots, upgrades) is interleaved with the user workload during opportune times. Proactive scheduling of data analytics work can result in personalized advertising, sentiment analysis, or product recommendation to be available timely, i.e., before the user leaves the site [3], [4], [5]. Virtual machine (VM) consolidation and migration is yet another example where accurate prediction of the physical machine utilizations can guide more effective system usage [6], [7], [8]. In all of the above cases, prediction of the intensity of peak loads and their timing becomes key to the effective launching of proactive management.

In this paper, we focus on data center workloads within a private cloud operated by IBM and used by major corporations for all their IT needs. Prior work on workload characterization at IBM datacenters [9] focused on statistical analysis of the usage of specific components of the virtual and physical machines, e.g., CPU and IO [9], [10]. This statistical analysis focused on averages, percentiles, and trends trying to understand how the workload evolves across a two-year period but largely ignored the time series of the various metrics. In this paper, we make a concentrated effort to focus on the time series and develop methodologies for accurate time series prediction of various workload metrics and especially peaks and their timing.

Classic time series models such as ARIMA [11] can be used for online prediction. Such models first need to be trained using past observations and can predict the upcoming workload. Alternatively, neural networks can be used in the same manner and provide a black box approach to predict future workload, especially to predict events that have been observed in the past. Superior to the classic time series models that use a linear function basis, neural networks model the input using non-linear functions, which improves their ability to handle more complex observations. Features gathered from the observations are not equally informative; some are correlated or relevant, while others are noise. Therefore, the key to effective neural network prediction is the discovery of the appropriate features. Neural network training is then conducted based on these.

In this paper, we develop a robust framework for prediction of data center time series (PRACTISE) and illustrate the flexibility of such a black box approach by showing remarkable accuracy in usage prediction of data center workloads in the wild. We focus on four components: CPU, memory, disk, and network. We focus on an actual production workload and particularly on 56 physical machines that host 775 virtual machines during a time period of 60 days. Based on observations of the workload pattern and its periodicity, we extract the features that identify the time periods in which the repetitive patterns occur. We also develop a bagging [12] and an online updating module to improve the stability, accuracy, and speed of our predictions. We provide detailed comparisons with ARIMA and and show that the proposed black box approach offers a significant improvement in predicting resource usage, by reducing average errors by three times. PRACTISE is lightweight and performs training and prediction much faster than default neural network methods, by an order of magnitude. This property allows it to be used online. **TBD – need quantitative info for peak loads and timing.**

This paper is organized as follows. Section II presents an

overview of the workload. Section III presents the machine learning model. Section IV presents extensive experimental evaluation. Section V discusses the potential use scenarios. Section VI discusses related work. We conclude in Section VII.

II. VMs Workloads in a Private Cloud

The target systems of this study are IBM private data centers, which are geographically distributed across all continents. These systems are used by various industries, including banking, pharmaceutical, IT, consulting, and retail, and are based on various UNIX-like operating systems, i.e., AIX, HP-UX, Linux, and Solaris. Those systems are highly virtualized, meaning that multiple virtual machines (VMs) are consolidated on a single physical box. Both VMs and boxes are very heterogenous in terms of resource configuration. The average virtulization level per box is ten to one [9]. We have collected resource utilization statistics from several thousands of VMs and boxes since February 2013. The finest observation granularity is 15 minutes[1]. The analysis here is based on two month data from March 2013 to April 2013.

We focus on usage of four types of resources: CPU, memory, disk, and network, on both VMs and boxes. Using the base observation window of 15 minutes, we collect the following statistics:

- CPU utilization: the percentage of time the CPU is active over the observation window.
- Memory utilization: the percentage of memory capacity used.
- Disk space usage: the percentage of allocated disk space used.
- Network bandwidth usage: the total network traffic rate measured in mega bits per second (Mbps)

The collected trace data is retrieved via `vmstat`, `iostat`, and supervisor specific monitoring tools.

The VM workloads within the IBM private cloud exhibit clear periodic patterns over time [9], see Figure 1. The figure focuses on two different VMs and illustrates the CPU utilization within successive time windows of 15 minutes across all 60 days. The upper plot shows a regular periodic pattern with a period of 7 days, while the bottom one shows a more complex pattern with a clear trend change over time. We also observe that similar periodic patterns exist in different resources, i.e., memory, disk, and network. To the interest of space, we do not present these results here. The periodicity of the pattern suggests that there are good opportunities for workload prediction that could drive better scheduling and resource management strategies.

To capture and quantify such periodical patterns, we perform statistical analysis of the workloads by computing the autocorrelation of the time series of CPU utilization. Autocorrelation is a mathematical representation of the degree of similarity in a time series and a lagged version of itself over successive time intervals. As such, it is ideal for discovering

[1]Collection of data is done by another corporatism company, thus 15 minutes is the finest available granularity.

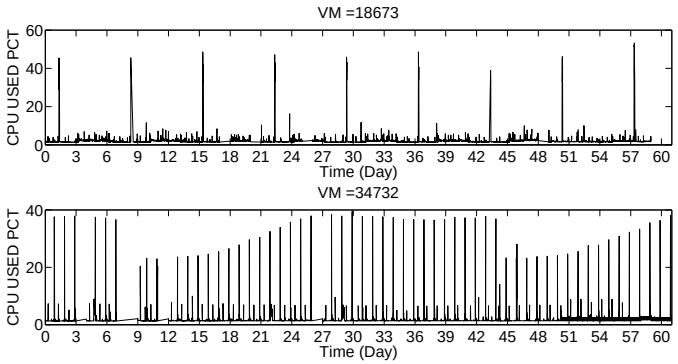

Fig. 1: CPU utilization over time for two different VMs.

repeating patterns by quantifying the relationship between different points of a time series as a function of the time lag [13]. The value of autocorrelation 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 correlation. Zero values suggest no periodicity, i.e., the successive points in the time series are independent. Negative values show that the two points that are lag elements apart are diametrically different. We show the autocorrelation of CPU resource usage for two VMs in Figure 2. It is clear that the autocorrelation becomes high at certain lag values and that the lag values[2] can be different for different VMs. In the following section, we demonstrate how to utilize autocorrelation to select the appropriate features in order to train a neural network that can model the workloads accurately.

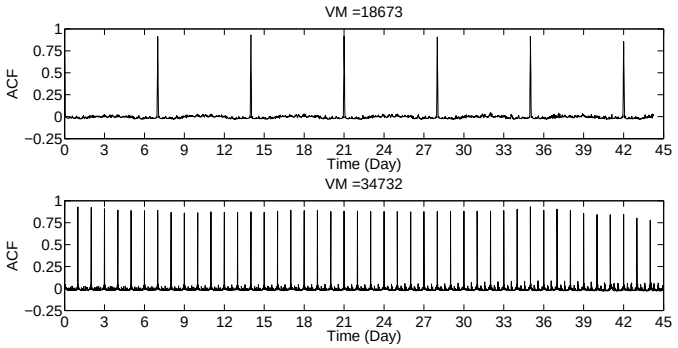

Fig. 2: Autocorrelation of CPU utilization for two different VMs.

III. Methodology

A time series prediction model uses past observations to forecast future values. There are different ways to build the time series prediction model. Traditional time series e.g., the ARMA/ARIMA [11] and Holt-Winters exponential smoothing [14] are based on a linear basis function, and as a result they are not effective in predicting complex behaviors. In addition, these models are backward looking only methods, which makes it difficult to capture any new patterns that have not appeared before. Furthermore, the underlying approximation function is complex and usually lacks intuitive

[2]Note that a lag of 1 corresponds to two successive 15 minutes intervals.

explanations. For all of the above reasons, it is difficult to improve the prediction accuracy of such types of models. On the other hand, neural networks are capable of modeling input as non-linear functions, which offers great potential in handling complex time series [15]. We start from the standard universal neural network toolbox provided by MATLAB [16] and then introduce PRACTISE by selecting more appropriate features, using bagging and online updating to improve its accuracy and stability. An overview of PRACTISE is shown in Figure 3. The workload is input and feed to the autocorrelation-based feature selection module. The selected features are input to the neural network training component. Then the bagging module process the aggregated results from neural networking model. The online updating model monitors the prediction error and trigger a retraining if large errors are detected. Next, we introduce each component in details.

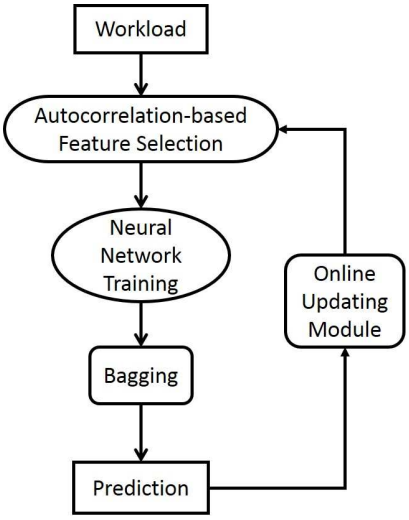

Fig. 3: Overview of PRACTISE.

A. Universal Neural Network Model

Artificial neural networks are inspired by biological neural networks [17] and are composed of many interconnected *neurons*. The weights associated with the neurons are used to approximate non-linear functions of the inputs and are tuned during a training process. Discovering appropriate features is the key to build an accurate neural network model. The universal neural network toolbox provided by MATLAB uses a generalized algorithm for feature selection. To train a neural network, the input data set is usually divided into three subsets [18]: training, validation, and test. The neural network uses the training set to tune tits weights and utilizes the validation set to determine the convergence point and prevent overfitting. The test set is used for evaluation of the training accuracy.

To understand the prediction accuracy of the standard neural network toolbox provided by MATLAB, we conducted extensive experiments. Figure 4 illustrates the default MATLAB prediction (tagged BaselineNN) of the utilization of the two VMs shown in Figure 1. We have trained and validated the neural network using the first 13 days in the data, and we show here the results for days 14 to 24. The figure clearly shows the neural network's pitfalls as prediction accuracy is often poor. Using the standard toolbox, the underlying feature selection algorithm is not tuned to optimize the information provided by the repeating patterns. Therefore, we are motivated to explore a better feature selection algorithm for selecting the appropriate features for the data center workloads that we have in hand.

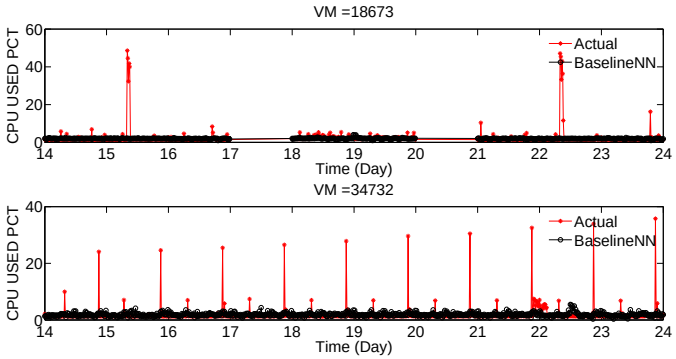

Fig. 4: CPU workload prediction by the neural network toolbox provided by MATLAB for two different VMs. The two gaps in the first plot are due to the VM being switched off.

B. Autocorrelation-based Features

Intuitively, appropriate features should reliably capture periodic behavior, changing trends, and repeating patterns. To identify the appropriate features, we resort to the correlogram in Figure 2 because autocorrelation can provide quantitative and qualitative information on the above factors. Figure 2 shows that there can be several lags with high positive autocorrelation values. This indicates that there exist several good candidate features that represent short-term to long-term correlation patterns. To automate the process, we use a local maximum function to identify the peak points in autocorrelations and use the respective lag values as features for neural network training. In this way, different correlation ranges from short-term to long-term can all be captured, which improves the effectiveness of the neural network model. The remaining steps are the same as with the universal neural network toolbox provided by MATLAB.

C. Bagging

The training features are not the sole factor in prediction accuracy of a neural network model; the quality of the trained model also depends on other factors. The training data sets are another crucial factor [12]. As discussed earlier, the training set is split into training, validation, and test subsets. Different ways of splitting may result in different samples being used at different stages and therefore result in different trained models. In order to minimize the artificial effects caused by a certain splitting rule, we split the data set randomly several times (e.g., 20 times) and each split trains a different model. In other words, we train a group of neutral network models by using the same data set but with different splits. Each model has its own prediction result. The prediction results from different models together become a distribution of prediction results. To compute the final prediction results from the distribution of prediction results, we first use the 3-sigma rule [19] (e.g., 99% confidence interval) and z-score [20] (e.g., within [-0.85,0.85])

VM ID	Training Time (sec)		Prediction Time (sec)	
	BaselineNN	PRACTISE	BaselineNN	PRACTISE
18673	300	30	257	10
34732	480	50	360	15

TABLE I: Training time using 14 days' data and prediction length of 1 day.

to filter out outliers and then compute the average of the remaining results as the final prediction. Such bagging may not always guarantee that optimal prediction is achieved, but it consistently improves the prediction accuracy compared to using only a single trained model.

D. Online Updating Module

In a real cloud environment, there can be sudden or permanent workload changes caused by unexpected events. As neural network models rely on past information to forecast the future, workload characterization changes may not be timely reflected in the prediction results. To ensure an agile response to workload characterization changes, we add an online updating module. The update is trigger-based on monitoring the prediction errors periodically. If the errors suddenly surge, a workload change is suspected and the neural network model is retrained. We emphasize that the computational cost of the neural network training and prediction is not significant thanks to the simple yet efficient feature selection process. Thus, it allows us to retrain the model online quickly at low cost. We demonstrate two examples in Table I to show how long PRACTISE takes for training and prediction compared to BaselineNN on a machine with 2.8 GHz Intel Core i7 CPU, 16 GB memory and 750 GB SSD. From the table, it is clear that both the training time and prediction time of PRACTISE are very low and that PRACTISE is more than one order of magnitude faster than BaselineNN. This different in times may appear modest (e.g., half minutes vs 5 minutes). If prediction has to be done simultaneously for thousands of VMs and multiple resources, the one-order of magnitude difference becomes significant. The training and prediction time change linearly with the amount of training data and prediction length.

IV. Experimental Evaluation

Methods We describe the methods used in evaluation below:

- **ARIMA**: the standard ARIMA algorithm, used as baseline comparison.
- **BaselineNN**: the default setting of the neural networking toolbox provided by MATLAB, used as a second baseline comparison.
- **PRACTISE**: the workloads prediction framework proposed in this paper.

Evaluation Strategies We use the first 14 days as training data and use the following days as the data for evaluation of the goodness of prediction. With the online updating module, PRACTISE automatically triggers a retraining if the monitored error is outside the confidence interval determined by the 3 sigma rule. We also evaluate different prediction lengths ahead to show the robustness of PRACTISE. We present the results for prediction length of 1 day ahead. We also present two more cases, one with 2 hours ahead (short window and one with 1 week ahead (long window). We evaluate PRACTISE against ARIMA and BaselineNN and in two prediction scenarios:

- State predictions: several scheduling or management frameworks [1], [21] do not require quantified prediction. Instead, workloads are classified into states, e.g., peak or non-peak states, and only qualitative predictions are needed. In such scenario, the quality of the prediction is measured by whether the future state can be predicted correctly. In our experiments, we count and compare the number of wrong state prediction events for different prediction methods.
- Quantified predictions: there is also a number of scheduling and management frameworks that require quantified prediction [22], [23], especially for systems that need to meet certain service level objectives. In this case, the quality of the prediction is measured by the prediction error, defined as $error = \frac{|prediction - actual|}{actual}$, where *prediction* is the predicted result and *actual* is the actual measurement.
- Prediction timing: often, it is critical to be able to not only predict a peak state, but also *when* this state occurs. We quantify the timing of the predictions across all VMs in a cumulative way, i.e., we count for how many 15-minute intervals the timing of the prediction of the peak state is delayed.

A. Prediction of Workload States

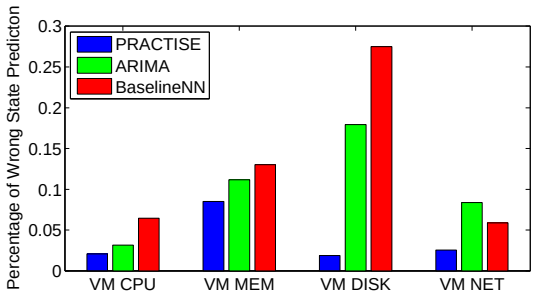

Fig. 5: The states prediction accuracy for different prediction methods. Prediction length is one day ahead.

The workloads are classified into two states: peak state with relatively high resource usage and non-peak state with relatively low resource usage. The threshold between peak and non-peak states is determined via K-means clustering. To quantify accuracy, we compare the number of wrong state predictions in Figure 5. Here we consider predictions across all VMs and plot the percentage of wrong state predictions for the time series of VM CPU utilization, memory utilization, disk space usage, and network bandwidth usage. PRACTISE consistently outperforms ARIMA and BaselineNN, with peak state prediction errors at about 2-3% for CPU, disk, and network, and 8% for memory. ARIMA and BaselineNN do surprisingly poorly in disk prediction.

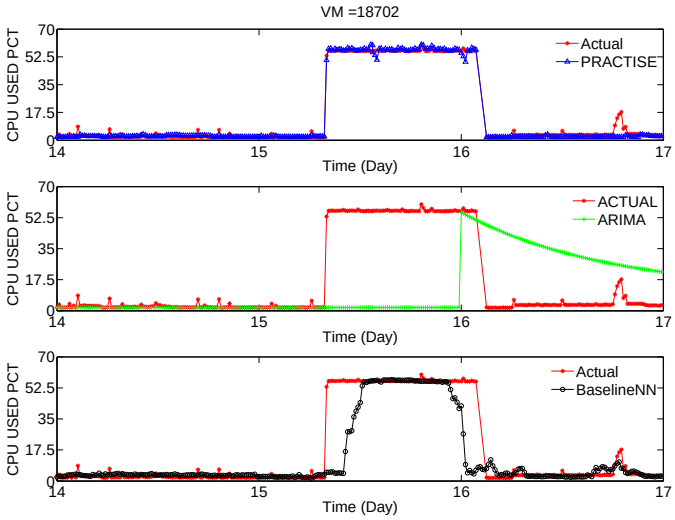

Fig. 6: Quantified prediction for VM CPU utilization. Each point corresponds to the finest workload granularity that is available, i.e., 15 minutes.

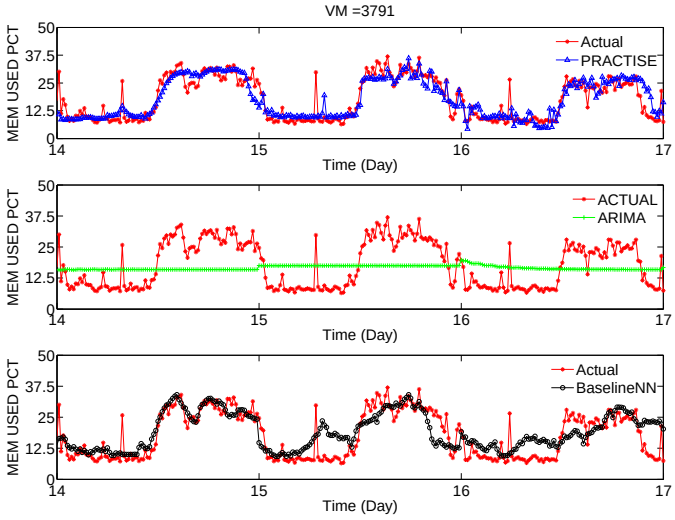

Fig. 7: Quantified prediction for VM memory utilization. Each point corresponds to the finest workload granularity that is available, i.e., 15 minutes.

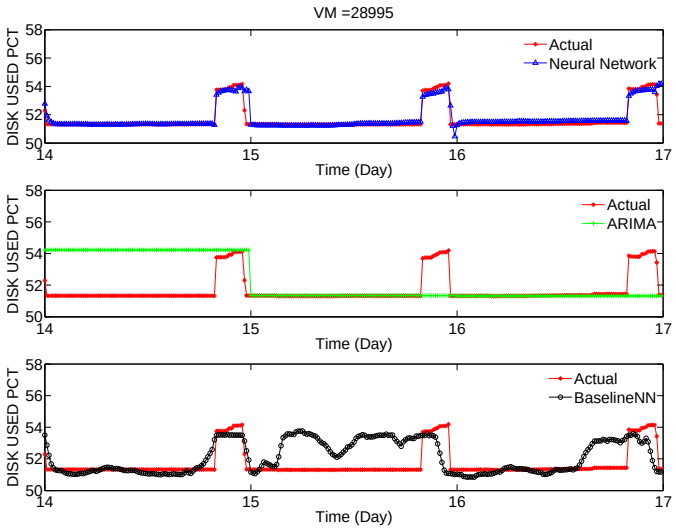

Fig. 8: Quantified prediction for VM disk space usage. Each point corresponds to the finest workload granularity that is available, i.e., 15 minutes.

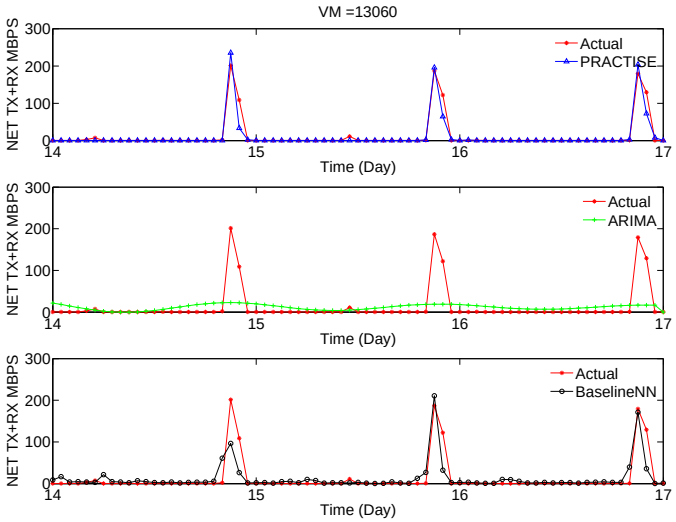

Fig. 9: Quantified prediction for VM network bandwidth usage. Each point corresponds to the finest workload granularity that is available, i.e., 15 minutes.

B. *Quantified Prediction*

Now we evaluate the accuracy of quantified prediction. We illustrate how PRACTISE accurately predicts the workload by showing overtime plots for the actual workload and predicted results. The results for VM CPU utilization, VM memory utilization, VM disk space usage, and VM network bandwidth usage are shown in Figure 6, Figure 7, Figure 8, and Figure 9 respectively[3]. It is clear that the prediction error of PRACTISE is consistent lower than both ARIMA and BaselineNN, especially for the sudden surge of the workloads, thanks to the more appropriate features and bagging used in PRACTISE. Note that predicting sudden workload surges is very important as it can drive scheduling/management to allocate timely the right amount of resources to prevent performance pitfalls.

While the results presented before show cases of consistent periodicity across time, in Figure 10 we show a more challenging case where the trends of the periodical pattern change. The results show that PRACTISE can effectively capture this thanks to the online updating component. ARIMA and BaselineNN can only predict events that have been observed before, and are therefore unable to capture such trend changes.

To quantify the errors of prediction, we show the CDFs of prediction errors for one specific VM (first column of Figure 11) for CPU, memory, disk, and network. We also show the mean and 90th percentile in the legend. The results illustrate clearly that PRACTISE is consistently superior in

[3]We zoom in the results in a 4 days' period for clear presentation.

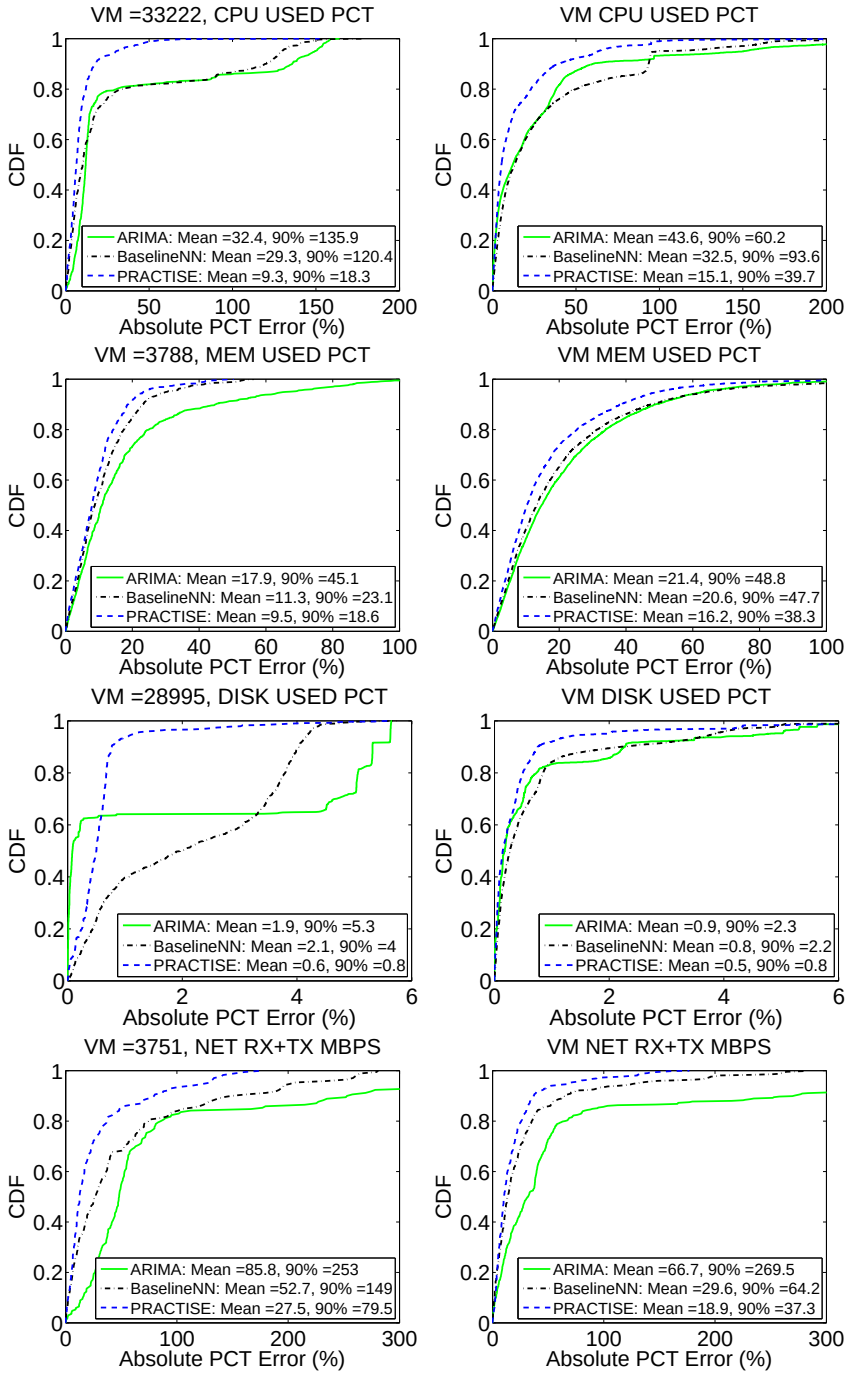

Fig. 11: Prediction error comparison for different prediction methods. The graphs are for VM CPU utilization (row 1), VM memory utilization (row 2), VM disk space usage (row 3), VM network bandwidth prediction (row 4). The first column is for a selected VM and the second column is accumulated results over all VMs. Perdition length is 1 day ahead.

accuracy comparing to ARIMA and BaselineNN, PRACTISE achieves up to 3 times better prediction accuracy than ARIMA and BaselineNN in terms of average prediction errors. The second column of Figure 11 illustrates the same information but cumulative across *all* 775 VMs. The superiority of PRACTISE is also clear.

TBD: new results and discussion about timing to be added here.

C. Robustness

To demonstrate the importance of bagging and online updating module, we also compare the prediction errors when bagging is activated and when the online updating module is activated, see Figure 12. The results indicate that with these two components, the prediction accuracy can be significantly improved, which verifies that these two enhancements are non-trivial and very useful.

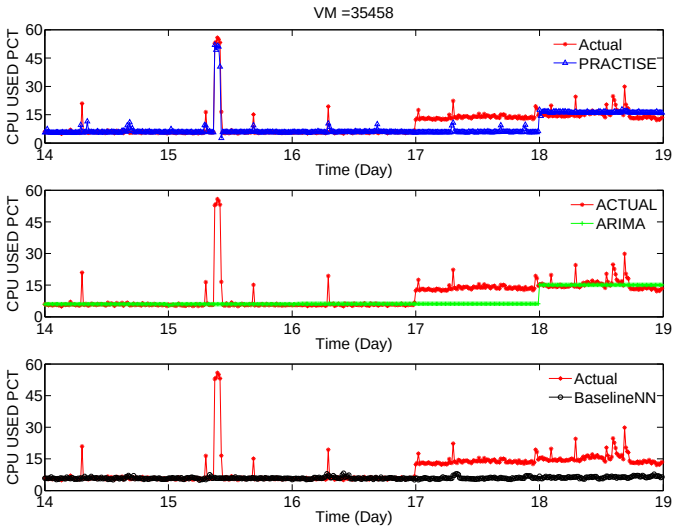

Fig. 10: Quantified prediction for CPU utilization, the trends changes after day 17. Each point corresponds to the finest workload granularity that is available, i.e., 15 minutes.

Finally, we evaluate PRACTISE for different prediction lengths. We demonstrate the prediction length of 2 hours and 1 week in Figure 13. The results clearly illustrate that PRACTISE consistently outperforms ARIMA and BaselineNN in different prediction lengths. In addition, the change in the size of the prediction window does not affect the robustness of PRACTISE.

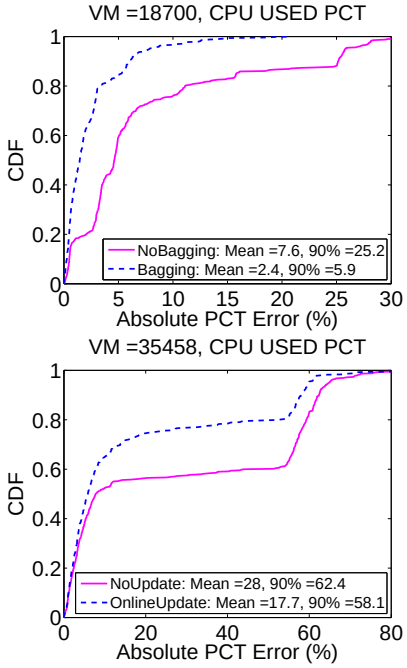

Fig. 12: Prediction error comparison of VM CPU utilization for with and without bagging (top plot), and for with and without online updating module (bottom plot).

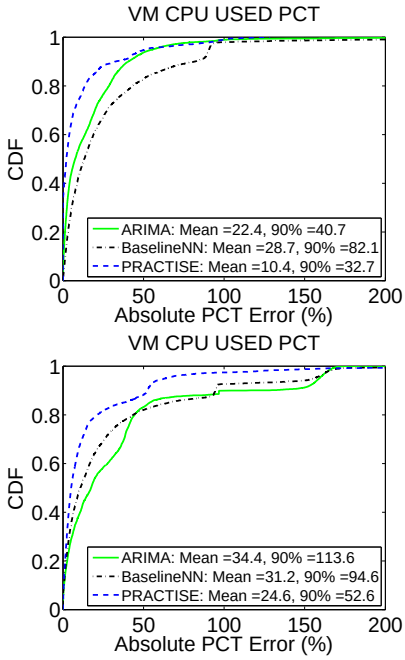

Fig. 13: Prediction error comparison of VM CPU utilization for prediction length of 2 hours (top plot) and 1 week (bottom plot). The results are accumulated across all VMs.

D. Challenging Cases

PRACTISE relies on the autocorrelation values as features for neural networking training. Here we evaluate a challenging case with poor autocorrelation structure, see Figure 14. The figure illustrates the autocorrelation plot of the memory utilization of a VM. The autocorrelation function switches between positive and negative values, which makes feature selection very challenging. The CDF of prediction errors is presented in Figure 15. The results suggest that even for this case, PRACTISE still achieves relatively good prediction accuracy and clearly outperforms ARIMA and BaselineNN. The robustness in prediction is due to the fact that PRACTISE does not solely rely on the autocorrelation features but also on bagging and online updating which contribute to more sophisticated predictions. We conclude that PRACTISE is effective, efficient, and robust.

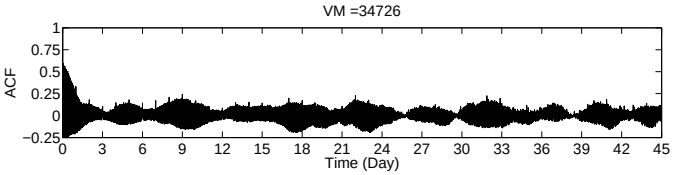

Fig. 14: Autocorrelation of memory utilization for a VM. A challenging case.

V. Discussion

In a dynamic environment, where resource usage is bursty as in the overwhelming cases of VM and PM resource usage in datacenters, being able to accurately *predict* the magnitude but

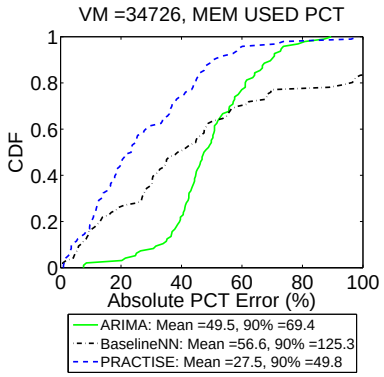

Fig. 15: Prediction error comparison of VM memory utilization for different prediction methods for VM 34726, see Figure 14.

also timing of “peak" loads can provide important leverage for successful VM consolidation, migration, and load balancing. In our prior characterization work of the IBM datacenter data, we have observed that the majority of VM migrations occur during midnight [9], this phenomenon was attributed to standard practice but the workload conditions that triggered migration were not provided by the traces.

In this paper, we provided the first important step that is required for VM consolidation and/or load balancing: prediction. This prediction is robust: even for cases where the workload burstiness does not have a clear repetitive pattern, we still manage to achieve remarkable accuracy, outperform ARIMA and plain vanilla neural network models, and more importantly for different future time windows that can range from 1 hour to a week. There are many important implications of this work:

- **Dynamic VM consolidation that is driven by resource demands**: PRACTISEcan provide VM usage of various resources (e.g., CPU, memory, disk) but the same prediction can be provided for PMs where the various VMs may be consolidated. Indeed, for PMs, the traces provide the number of VMs per PM, as well as resource usage information. PRACTISE can be used to predict this information and drive different consolidation strategies[4].
- **VM consolidation driven by prediction of multiple resource availability**: Here, we are able to explore the availability of both CPU and memory on PMs and how this would change by consolidation strategies. Most importantly, having predictions on multiple resources, one can focus on the most scarce resource and drive its consolidation policies accordingly. Even more specifically, since there is significant burstiness in both memory and CPU usage, a strong prediction model can really help in optimizing usage of both resources.
- **Different intervals**: Since VM migration is not instantaneous, the ability to predict within different time intervals can be used as a tool to more effective consolidation.

[4]PRACTISE shows excellent prediction for the PMs in this data set. Such results are not shown due to lack of space.

- **A bird’s eye view of datacenter resource usage**: beyond VM consolidation, predicting how load balancing evolves across time can leverage better energy and power consumption policies.

TBD: general comments on SLOs and how all of the above can really help

VI. Related Work

ARMA/ARIMA [11] have been widely used for time series prediction in several systems areas. Tran and Reed use ARIMA to improve block prefetching for scientific applications [24]. They use ARIMA to predict the temporal access pattern and Markov models to identify spatial access patterns and manage to identify what and how much to prefetch. Their predictor is implemented on the Linux file system. In [5] ARIMA is used for effective user traffic prediction for capacity planning. The authors focus on cost-efficient database replication that is driven by the anticipated user traffic within the linkedin social network. ARIMA models have also been used in sensor networks to reduce the frequency of sampling and to improve on energy efficiency by transmitting only deviations from the ARIMA-predicted values [25]. Anomaly detection is yet another area where ARIMA models have been used [26].

Machine learning techniques are used to overcome the limitation of the linear basis function of ARIMA models and are used for effective characterization of TCP/IP [27] and web server views [28]. Neural networks are used for performance prediction of the total order broadcast, which is a key building block for fault-tolerant replicated systems [29]. Ensembles of time neural network models have been used to project disk utilization trends in a cloud setting [30]. Neural networks and hidden Markov models are used for automatic IO pattern classification and are evaluated with both sequential and parallel benchmarks [21]. Probabilistic models that define workload states via Markov Modulated Poisson Processes have been used in [1] to interleave workloads with different performance objectives.

The effectiveness of the proposed neural network approach that we advocate in this paper is based on statistical analysis of the workload so that the most relevant features are selected for the training data set. Training the model with careful feature selection significantly improves its accuracy and stability but also increases the speed of training and prediction, making it appropriate to use for online performance prediction and capacity planning. Meanwhile, due to the appropriate feature selection, PRACTISE can provide short-term (e.g., 15-min) to long-term (e.g., one day/one week ahead) predictions and achieve better accuracy than other prediction methods such as ARIMA and BaselineNN. This superiority facilitates robust long-term capacity planning and resource allocation.

VII. Conclusion and Future Work

In this paper, we develop PRACTISE, an enhanced neural network based framework for predicting usage of various resources at datacenters. PRACTISE uses autocorrelation-based feature selection, boostrap aggregation, and online updating. We extensively evaluate PRACTISE on predicting CPU, memory, disk and network usage on a set of 775 VMs over a period of 2 months and compare its prediction effectiveness

to ARIMA and basic neural network models. We are able to achieve 3 times better prediction accuracy. Thanks to the excellent prediction accuracy of PRACTISE, we are able to capture the "peak" loads in terms of intensities and timing, in contrast to the classic time series models. Consequently, PRACTISE will enable us to explore VM consolidation and migration policies that are tailored to cater to peak demands in a cost-effective way.

REFERENCES

[1] J. Xue, F. Yan, A. Riska, and E. Smirni, "Storage workload isolation via tier warming: How models can help," in *Proceedings of the 11th ICAC*, 2014, pp. 1–11.

[2] Y. Zhang, G. Soundararajan, M. W. Storer, L. N. Bairavasundaram, S. Subbiah, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Warming up storage-level caches with Bonfire," in *FAST*, 2013, pp. 59–72.

[3] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, "Starfish: a self-tuning system for big data analytics." in *CIDR*, vol. 11, 2011, pp. 261–272.

[4] J. Cohen, B. Dolan, M. Dunlap, J. M. Hellerstein, and C. Welton, "Mad skills: new analysis practices for big data," *Proceedings of the VLDB Endowment*, vol. 2, no. 2, pp. 1481–1492, 2009.

[5] Z. Zhuang, H. Ramachandra, C. Tran, S. Subramaniam, C. Botev, C. Xiong, and B. Sridharan, "Capacity planning and headroom analysis for taming database replication latency: experiences with linkedin internet traffic," in *Proceedings of the 6th ACM/SPEC ICPE*, 2015, pp. 39–50.

[6] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *NSDI*. USENIX Association, 2005, pp. 273–286.

[7] M. Nelson, B.-H. Lim, G. Hutchins *et al.*, "Fast transparent migration for virtual machines." in *USENIX ATC*, 2005, pp. 391–394.

[8] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and gray-box strategies for virtual machine migration." in *NSDI*, vol. 7, 2007, pp. 17–17.

[9] R. Birke, A. Podzimek, L. Y. Chen, and E. Smirni, "State-of-the-practice in data center virtualization: Toward a better understanding of VM usage," in *DSN*, 2013, pp. 1–12.

[10] R. Birke, M. Björkqvist, L. Y. Chen, E. Smirni, and T. Engbersen, "(big)data in a virtualized world: volume, velocity, and variety in cloud datacenters," in *FAST*, 2014, pp. 177–189.

[11] B. George, *Time Series Analysis: Forecasting & Control, 3rd ed*. Pearson Education India, 1994.

[12] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

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

[14] P. Goodwin, "The holt-winters approach to exponential smoothing: 50 years old and going strong," *Foresight*, pp. 30–34, 2010.

[15] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of Intelligent and Robotic Systems*, vol. 31, no. 1-3, pp. 91–103, 2001.

[16] H. Demuth, M. Beale, and M. Hagan, "Neural network toolbox^{TM} 6," *User's Guide*, 2008.

[17] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*, 1st ed. Cambridge, MA, USA: MIT Press, 1995.

[18] T. Hill, M. O'Connor, and W. Remus, "Neural network models for time series forecasts," *Management Science*, vol. 42, no. 7, pp. 1082–1092, 1996.

[19] G. Upton and I. Cook, *A Dictionary of Statistics 3e*. Oxford university press, 2014.

[20] M. L. Marx and R. J. Larsen, *Introduction to mathematical statistics and its applications*. Pearson/Prentice Hall, 2006.

[21] T. M. Madhyastha and D. A. Reed, "Learning to classify parallel input/output access patterns," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 8, pp. 802–813, 2002. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPDS.2002.1028437

[22] X. Zhu, D. Young, B. J. Watson, Z. Wang, J. Rolia, S. Singhal, B. McKee, C. Hyser, D. Gmach, R. Gardner *et al.*, "1000 islands: Integrated capacity and workload management for the next generation data center," in *Autonomic Computing, 2008. ICAC'08. International Conference on*. IEEE, 2008, pp. 172–181.

[23] P. Xiong, C. Pu, X. Zhu, and R. Griffith, "vperfguard: an automated model-driven framework for application performance diagnosis in consolidated cloud environments," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*. ACM, 2013, pp. 271–282.

[24] N. Tran and D. A. Reed, "Automatic ARIMA time series modeling for adaptive I/O prefetching," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 4, pp. 362–377, 2004. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TPDS.2004.1271185

[25] M. Li, D. Ganesan, and P. J. Shenoy, "PRESTO: feedback-driven data management in sensor networks," *IEEE/ACM Trans. Netw.*, vol. 17, no. 4, pp. 1256–1269, 2009. [Online]. Available: http://doi.acm.org/10.1145/1618562.1618581

[26] B. Zhu and S. Sastry, "Revisit dynamic ARIMA based anomaly detection," in *PASSAT/SocialCom 2011, Privacy, Security, Risk and Trust (PASSAT), 2011 IEEE Third International Conference on and 2011 IEEE Third International Conference on Social Computing (SocialCom), Boston, MA, USA, 9-11 Oct., 2011*, 2011, pp. 1263–1268. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/PASSAT/SocialCom.2011.84

[27] P. Cortez, M. Rio, M. Rocha, and P. Sousa, "Multi-scale internet traffic forecasting using neural networks and time series methods," *Expert Systems*, vol. 29, no. 2, pp. 143–155, 2012.

[28] J. Li and A. W. Moore, "Forecasting web page views: methods and observations," *Journal of Machine Learning Research*, vol. 9, no. 10, pp. 2217–2250, 2008.

[29] M. Couceiro, P. Romano, and L. Rodrigues, "A machine learning approach to performance prediction of total order broadcast protocols," in *4th IEEE SASO*, 2010, pp. 184–193.

[30] M. Stokely, A. Mehrabian, C. Albrecht, F. Labelle, and A. Merchant, "Projecting disk usage based on historical trends in a cloud environment," in *Proceedings of the 3rd workshop on ScienceCloud*, 2012, pp. 63–70.