*Journal of Applied Science and Engineering, Vol. 19, No. 4, pp. 497–504 (2016)* *DOI: 10.6180/jase.2016.19.4.13*

# A Big Data Processing-oriented Prediction Method of Cloud Computing Service Request

Shenghui Zhao[1,2], Haibao Chen[1,2]*, Ruibin Zhao[1,2], Yuyan Zhao[1,2] and Guilin Chen[1,2]

[1]*School of Computer and Information Engineering, Chuzhou University, Chuzhou 239000, P.R. China*

[2]*Anhui Center for Collaborative Innovation in Geographical Information Integration and Application, Chuzhou 239000, P.R. China*

## Abstract

In order to guarantee the cloud service quality, the service should be able to dynamically predict the change of data processing request. Existing prediction methods in cloud are mostly focused on the amount of computing resource required by service. In fact, in cloud computing environment for big data processing, it is not enough to simply predict the computing resource, because when the created virtual machine is far from the data, it will need a certain time to transfer data to the virtual machine for processing. To solve this problem, in this paper, we propose a data-centered prediction method using Bayes classifier, which can make prediction for data type or location based on the data resources needed by the service request. We carry out experiments with Google cluster trace, and the experimental results show that our method performs better than the existing methods. For example, our method improves the load prediction accuracy by 45–60% compared to other state-of-the-art methods based on final state-based method, simple moving average method, linear weighted moving average method, exponential moving average method, and prior probability-based method.

***Key Words*:** Big Data, Bayes Classifier, Data-centered Prediction Method, Google Cluster Trace

## 1. Introduction

Cloud computing has almost infinite resources and a service-centered business model, thus it has become the inevitable choice of big data processing [1]. In a big data processing-oriented cloud computing system, it is generally to create and deploy cloud services based on the user's task and its requirements for resources. In the case of data and computation intensive scientific applications, large amount of requests of data processing may surge in a short time. In order to guarantee the service quality, cloud service should be able to dynamically perceive the change of data processing request. On the one hand, by monitoring the service request information (e.g., the number and speed of the service requests), we can judge whether it occurs a big data processing request or not. On the other hand, if the big data processing request is generated, the size of the data and the demand for the resources need to be predicted.The Internet of Things (IoT) is the collection of billions of end devices, so it can generate big data which be computed in cloud computing at present.

Existing service prediction methods are mostly focused on computing resource quantities needed by service, such as [2]. In fact, in the cloud computing environment, especially for big data processing cloud computing environment (such as remote sensing cloud [3]), it is not enough to simply predict the computing resource needed by service request, because if the created virtual machine is far from the data, it will need a certain time to transfer data to the virtual machine for processing. Therefore, in this scenario, it also needs to predict the big data resources to be processed by the requested service (such as predicting the location of big data), and based on the

*Corresponding author. E-mail: chb@chzu.edu.cn

principle of proximity, to create virtual machines on the node where the data resides at or near, thus saving time spent by data transmission and accelerating the progress of data processing.

To solve this problem, in this paper, we propose a data-centered prediction method, which can make prediction based on the data resources needed by the service request (for example, to predict data type or location).

In big data processing oriented cloud system, the benefits of successfully predicting data resource types (position) of cloud service requests, and the amount of computing resources are: (1) cloud system can use idle VMs (virtual machines) as much as possible to avoid idle VMs restart; and (2) cloud system can start VMs in advance, so as to achieve the purpose of rapid response to users' requests.

The contributions of this paper are two-fold:

- We propose a service requests forecasting method based on Bayesian model, which is used to predict the location of data resources for the next service request, so as to speed up data processing.
- We evaluate the service requests forecasting method with the load trace of Google data centers.

This paper is organized as follows. Section 2 introduces and analyzes the load trace of Google data center. Section 3 presents a forecasting method based on Bayesian model, which is evaluated in section 4. The related work is discussed in section 5. Section 6 concludes this paper.

## 2. Analysis of Load Trace of Google Data Center

The research of prediction method here is based on the load trace of Google data center. Google [4] tracks the job information of the production system (about 12000 machines) in a month in 2011 with minute-level granularity, including more than 670000 jobs and more than 4 million task events. Users submit jobs to a batch scheduler, where each job contains a task set and the resource constraints (e.g., CPU and memory) set. A batch scheduler orderly distributes these tasks to physical machines. The job events table of Google cluster trace includes 8 fields: timestamp, missing info, job ID, event type, the user name, scheduling class, job name, and logical job name. Among them, all of the jobs have a scheduling class, which is represented by a single number and roughly represents the delay sensitivity of the job. For example, number 3 represents a more delay sensitive task, while the 0 represents a non-productive task.

In task resource usage table of Google cluster trace, there contains 18 fields: start and end time, job ID, task index, machine ID, CPU usage-mean (mean and maximum in 1s window), memory usage, assigned memory, unmapped page cache memory usage, page cache memory usage and maximum memory usage, disk I/O time-mean, local disk space used (mean), CPU rate – max, disk I/O time – max, cycles per instruction (CPI), memory accesses per instruction (MAI), sampling rate, aggregation type. Among them, the job ID field indicates the ID of the current task. The average of the CPU, memory, and disk used by the current task during sampling are represented by CPU usage, memory usage, and local disk space used (mean) fields, respectively.

In Figure 1, it can be seen that data of CPU, memory and disk above are basically consistent with normal distribution.

## 3. Bayesian Model-based Prediction Method

The goal of this paper is to predict the location of data resource required by cloud service request. Taking

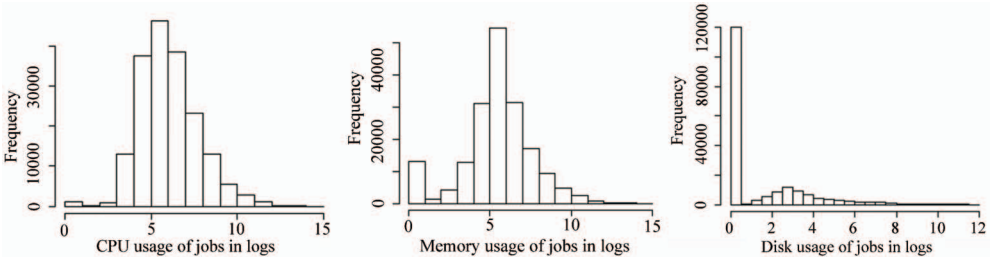


**Figure 1.** Histogram of job usage for resources.

into account the lack of publicly available related cloud service history in academic and industrial circles for this test, an indirect data set (the Google data center trace) has been adopted for study.

The basic idea of this paper is to use the Bayes classifier to generate a posteriori probability distribution of the job type according to the prior probability distribution and the amount of job demand for resources.

### 3.1 Bayes Classifier

Bayes classifier [3−5] is a classic supervised learning classifier in data mining [6]. Bayes classification includes five major steps:

(1) Determine the target state set (represented by $W = (\omega_1, \omega_2, \ldots, \omega_m)^T$, where m is the number of states), and the event vector with h independent features (represented by $\chi = (x_1, x_2, \ldots, x_h)^T$);
(2) Based on the training set (sample), compute the prior probability distribution for target states $P(\omega_i)$;
(3) For each state $\omega_i$, compute joint probability distribution $p(\chi|\omega_i)$;
(4) Calculate posterior probability according to formula (1);
(5) Make decisions based on risk function $\lambda(\widehat{\omega_l}, \dot{\omega}_l)$, where $\widehat{\omega_l}$ and $\dot{\omega}_l$ respectively represents the state of predicted values and real values.

$$P(\omega_i \mid x_j) = \frac{p(x_j \mid \omega_i)P(\omega_i)}{\sum_{k=1}^{m} p(x_j \mid \omega_i)P(\omega_k)} \tag{1}$$

Based on different risk functions, Bayes classifier will use different ways to make decision. This paper uses the Naive Bayes classifier [4], the risk function used is shown in formula (2). And the predicted state value $(\widehat{\omega_l})$ is determined by formula (3).

$$\lambda(\widehat{\omega_l}, \dot{\omega}_l) = \begin{cases} 0, & \widehat{\omega_l} - \dot{\omega}_l = 0 \\ 1, & \widehat{\omega_l} - \dot{\omega}_l \neq 0 \end{cases} \tag{2}$$

$$\widehat{\omega_l} = \arg\max\, p(\omega_i \mid x_j) \tag{3}$$

From the above analysis, the target state vector and the event feature vector are the key to the accurate prediction.

### 3.2 Target State and Event Feature

In the data set of Google data center trace, we take each job record as a cloud service request, and the type of job as the type of the data resource required by cloud service request. Without loss of generality, this paper at first assumes that the data center is based on the type of data to make the placement decision. Therefore, by predicting of the type of the data resource required by cloud service request, we can obtain the corresponding data storage location. On this basis, the research work of this paper is converted to how to predict the type of job using Google data center trace.

According to the job classification of Google data center, the target states are job category. The status vector consists of 4 elements, namely, category 1, category 0, category 2 and category 3. By analyzing the job record of Google data center, 9 features are extracted as the event in the Bayesian model, and each feature can partly reflect the attribute of the job. The 9 features are described below.

**The number of Task (*NumofTasks*):** It is used to indicate the number of tasks a job contains, namely, $|Job|$.

**The makespan of Job (*Makespan*):** It describes the time between the start and the end of the job running. In the job records of Google data center, there are only the start and end time of each task. Therefore, based on the task resource usage table in job traces of Google, we use formula (4) to calculate the maekspan of jobs (namely, the difference between the latest finish time and the earliest start time in all tasks for a job).

$$\begin{aligned} Makespan = {} & Max\{task_i^{EndTime}, i \in [1, |Job|]\} \\ & - Min\{task_i^{BeginTime}, i \in [1, |Job|]\} \end{aligned} \tag{4}$$

**Standard deviation of running time for all tasks (*StDevTime*):** It is calculated by formula (5), where AveTime is the average of the completion time for all tasks of the current job.

$$StDevTime = \sqrt{\frac{1}{|Job|}\sum_{i=1}^{|Job|}(Task_i^{EndTime} - Task_i^{BeginTime} - AveTime)^2} \tag{5}$$

**The amount of used CPU resources (*UsedCPU*):** It is used to indicate the amount of CPU resources that a

job uses when it completes, and can be got by formula (6), i.e., it is the sum of CPU resources for all tasks of the job.

$$UsedCPU = \sum_{1}^{|Job|} task_i^{CPU} \tag{6}$$

**Standard deviation of the amount of CPU resources for all tasks (*StDevCPU*):** It is calculated by formula (7), where AveCPU is the average of CPUs used by all tasks of the current job.

$$StDevCPU = \sqrt{\frac{1}{|Job|}\sum_{i=1}^{|Job|}(Task_i^{CPU} - AveCPU)^2} \tag{7}$$

**The amount memory resources used (*UsedMem*):** It is used to indicate the amount of memory resources that a job uses when it completes. And we use formula (8) for calculation, namely, it is the sum of the memory resources used by all tasks of the job.

$$UsedMem = \sum_{1}^{|Job|} task_i^{Mem} \tag{8}$$

**Standard deviation of the amount of memory resources for all tasks (*StDevMem*):** It is calculated by formula (9), where AveMem is the average value of the amount of memory used by all tasks of the current job.

$$StDevMem = \sqrt{\frac{1}{|Job|}\sum_{i=1}^{|Job|}(Task_i^{Mem} - AveMem)^2} \tag{9}$$

**The amount disk resources used (*UsedDisk*):** It is used to indicate the amount of disk resources that a job uses when it completes. And we use formula (10) to calculate, namely, it is the sum of the memory resources used by all tasks of the job.

$$UsedDisk = \sum_{1}^{|Job|} task_i^{Disk} \tag{10}$$

**Standard deviation of the amount of disk resources for all tasks (*StDevDisk*):** It is calculated by formula (11), where AveDisk is the average value of the disk used by all tasks of the current job.

$$StDevDisk = \sqrt{\frac{1}{|Job|}\sum_{i=1}^{|Job|}(Task_i^{Disk} - AveDisk)^2} \tag{11}$$

Therefore, a job can be described by the characteristics in Table 1.

### 3.3 Discretization of Feature Vector

Considering the range of the feature vector is too large and continuous, we have carried on discrete processing to the value of the feature vector. The basic method is to handle the data in logs. If we directly use discretization processing by rounding at this moment, then the data state number obtained will become too few (as shown in Figure 1, only up to 16 states (i.e., 0 to 15)). In order to increase the number of states, we multiple it by 1000 (an empirical value), then discretize it by rounding the data.

### 3.4 Bayes-based Type Prediction Algorithm

After discretization of the above feature vectors, a Bayes-based type prediction algorithm is given as follows. The input of the algorithm is job history records, which is the Google data center trace used in this paper. The output of the algorithm is the type of job predicted by it.

**Table 1.** Job features and description

| Features | Description |
|---|---|
| Makespan | The time needed for the job to complete |
| NumofTasks | The number of tasks in the job |
| StDevTime | Standard deviation of the running time of all tasks in a job |
| StDevCPU | Standard deviation of the amount of CPU resources used by all tasks in a job |
| StDevMem | Standard deviation of the amount of memory resources used by all tasks in the job |
| StDevDisk | Standard deviation of the amount of disk resources used by all tasks in the job |
| UsedCPU | Total CPU resources required for job to complete |
| UsedMem | Total memory resources required for job to complete |
| UsedDisk | The total disk resources required for the job to complete |
| Class | The category of the job |

**Algorithm 1.** Bayes-based type prediction algorithm
**Input**: Job history records (Google data center trace)
**Output**: The type of job

1: Get the number of tasks for each job as the first feature of the job, $x_1$;
2: Get the 2nd to 9th feature of the job (namely, $x_2 - x_9$) by formula (4) (5) (6) (7) (8) (9) (10) (11);
3: Based on the training set, calculate the prior probability distribution of each job type $P(\omega_i)$;
4: For each type of job $\omega_i$, compute the joint probability distribution $p(\chi|\omega_i)$, where $\chi = (x_1, x_2, \ldots, x_9)^T$;
5: Calculate the posterior probability according to formula (1);
6: Based on the risk function $\lambda(\widehat{\omega_l}, \dot{\omega}_l)$ in formula (2), make a decision about the job type, where $\widehat{\omega_l}, \dot{\omega}_l$ respectively represents the predicted value and the real value;
7: Return the job type.

## 4. Performance Evaluation

### 4.1 Algorithms in Experimental Comparison

In addition to the prediction method in this paper, we also implement five other classic prediction methods, as shown below. Among them, the collection of jobs within the event window is denoted by $J = (J_1, J_2, \ldots, J_d)^T$.

**Final state-based method:** The type of the last job recorded in the event window is used to predict the type of the next job.

**Simple moving average method (SMA):** The average value of all the records in the event window after rounding (for example, the ceiling of a number) is used to predict the type of the next job.

**Linear weighted moving average method (LWMA):** the linear weighted average value of all job records in the event window after rounding (for example, the ceiling of a number) is used to predict the type of the next job. In particular, the linear weighted average value is obtained by formula (12).

$$WMV(J) = \frac{\sum_{i=1}^{d}(d-i+1)J_i}{\sum_{i=1}^{d} i} = \frac{2}{d(d+1)}\sum_{i=1}^{d}(d-i+1)J_i \tag{12}$$

**Exponential moving average method (EMA):** This method uses formula (13) to predict values (the prediction value at time $t$ is denoted by $S(t)$), where $J_1$ is the last job, $\alpha$ value is to optimize the accuracy and adjusted by experience.

$$S(t) = \alpha \cdot J_1 + (1 - \alpha) \cdot S(t-1) \tag{13}$$

**Prior probability-based method (PP):** This method uses a job class with a maximum prior probability as the predicted value for the next job type.

### 4.2 Experiment

In order to evaluate the accuracy of the prediction, we use a metric: success rate, i.e., the ratio of the number of correct predictions to the total number of predictions. From the above analysis, the track records of Google data center of 10 days include 183859 job information entries totally. In order to compare the performance of different methods, we design three experiments: the first one makes comparison of different methods by fixing the size of training set and test set; the second one evaluates the success rate of different methods by changing the size of training set and test set; and the last one compares the time spent by different methods with different training sets and test sets.

In the first experiment, we divide the data set into two parts: the training data set (the first 182859) and the test data set (the last 1000). The training data set is used to train the model, for example, to calculate the prior probability $P(\omega_i)$ and conditional probability $p(x_j|\omega_i)$ from formula (1). Test data set is used to verify the performance of different methods.

Experimental results are shown in Figure 2. From the graph, we can see that the Bayes-based method has the highest prediction success rate (94.69%), while the

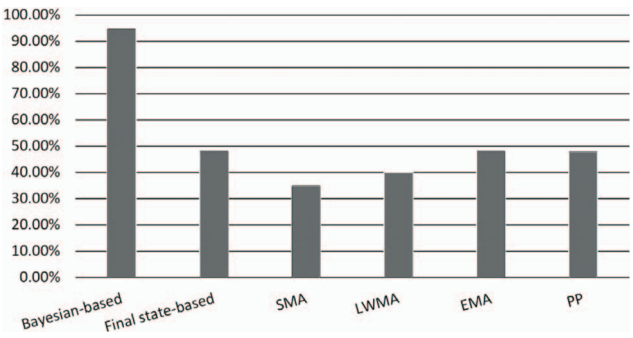


**Figure 2.** Prediction success rates of different methods.

other methods are between 49% and 35%. Specifically, the success rate of final state-based prediction method is 48.24%, and the success rate of simple moving average method is about 35%. The linear weighted moving average method has the highest success rate (40%) when the event window is 2. The exponential moving average method has the highest success rate (48.24%) when $\alpha$ = 0.9. The success rate of maximum a priori probability-based method is 48%.

In the second experiment, we evaluated the success rate of different methods with different training sets and testing sets. The training sets and the test sets are shown in Table 2.

Experimental results are shown in Figure 3.

As can be seen from the figure, our proposed method (Bayes-based method) has the highest success rate when the training set and the test set change (the success rate of the proposed method is between 94.69% and 92.69%), followed by EMA method (between 39.56% and 47.83%), final state-based method (between 39.56% and 47.8%), PP method (between 35.85% and 44.24%), LWMA method (between 27.78% and 36.73%), and SMA method (27.27% and 36.01%).

**Table 2.** The numbers of jobs in training sets and test sets

| ID | The number of jobs in training set | The number of jobs in testing set |
|---|---|---|
| T1000 | 182839 | 1000 |
| T5000 | 178839 | 5000 |
| T10000 | 173859 | 10000 |
| T15000 | 168839 | 15000 |
| T20000 | 163859 | 20000 |
| T30000 | 153859 | 30000 |

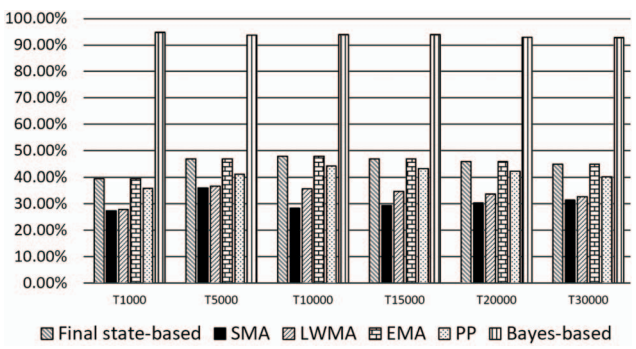


**Figure 3.** Experimental results with different training sets and test sets.

In the third experiment, we compare the time spent by different methods with different training sets and test sets. The descriptions of training and test sets are shown in Table 2.

The experimental results are shown in Figures 4 and 5. From these two figures, it can be seen that our proposed method spends the longest time, followed by PP method, LWMA method, SMA method, EMA method, and final state-based method. The reason behind the results is that our proposed method needs to train the data set with the most of the time. And the time increases with the sizes of training sets.

To sum up, our proposed method has the highest prediction accuracy. However, compared with other methods, the time spent by our method is the most.In future work, we will focus on the parallelization of our method to reduce the time it spends on training data set.

## 5. Related Work

There are some work that have described the features of load in the system, they are mainly concerned about

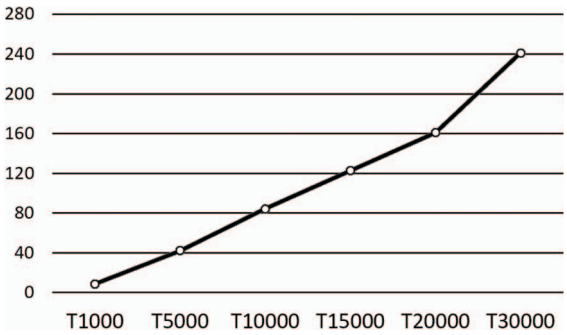


**Figure 4.** Time spent by our proposed method (unit: second).

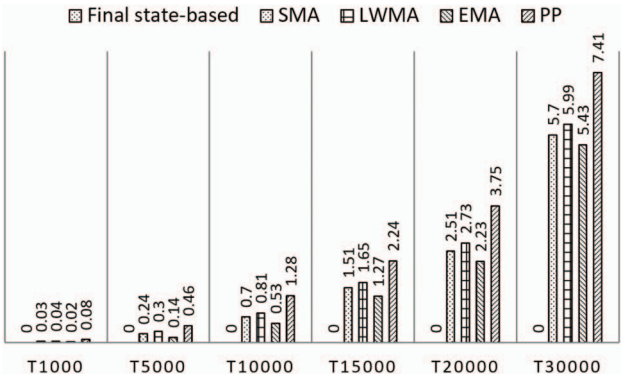


**Figure 5.** Time spent by other methods (unit: second).

the placement constraints [7] or usage morphology [8] of job. For example, Sharma from the perspective of [7], made a detailed study of the impact of job placement constraints on performance based on resource utilization, while Reiss in [8] proposed a model that can characterize the job usage morphology of the Google computation clusters. Khan in [9] designed a model to capture CPU load autocorrelation in cloud data center by means of Hidden Markov model (HMM). Barnes introduced a regression-based prediction method for paralleling application workload with error within 6.2% and 17.3% in [10]. Roy et al. in [11] proposed a model prediction algorithm, which uses two order autoregressive moving average method for the prediction of workload. Saripalli et al. proposed a two-step prediction method and hot spot detection model are proposed in [12]. Gong et al. adopted mode matching and state driven method to predict the workload in [13], where it first uses the signal processing technique to identify the repeat patterns, and if no feature is found, it will use statistical state driven method, and use discrete time Markov chains to predict the recent needs. Carrington et al. in [14], for example, predict load values based on convolution. Khazaei et al. in [15] use queuing systems for performance analysis in cloud. Yang et al. in [16] proposed a new method based on PSR and EA-GMDH for host load prediction in cloud computing system. Gu et al. in [17] presented a pattern discovery model for host load data. Gmach et al. in [18] use Fourier transform-based off-line method to extract the patterns of the long-term cyclic workloads. In addition, there are other work used offline or online analysis method in [19,20] for the requirement of application resource on the basis of the benchmark program or the actual application of workload. In comparison, we focus on the prediction of the location of the data required by the next service in the cloud computing environment.

## 6. Conclusions and Future Work

In the cloud computing system for big data processing, the benefits of the successful prediction of the type of data resource (location) and the amount of computing resource are as follows: (1) considering the principle of data proximity, it can use the idle virtual machines to avoid restart after shutting down; (2) in consideration on the principle of data proximity, it can start virtual machine in advance, to eliminate the start-up time the service waits for virtual machine, so as to achieve rapid response to user requests. In this paper, we present nine relatively independent feature vectors of user jobs. And we propose a service prediction method based on Bayesian model to predict the location of the data resource needed by the next service request. We use Google's data center traces within 10 days to evaluate the proposed method. The experimental results show that the prediction success rate of the proposed method is about 93%, significantly outperforms the existing related work (including simple moving average method, linear weighted moving average method, exponential moving average method and prior probability method, etc.).

Based on the work of this paper, we will focus on the deployment and start of virtual machine nearby the location of data needed by service.

## Acknowledgements

This work is supported by the Anhui Natural Science Foundation of China under Grant (No. 1408085MF126 and No. 1608085QF147) and Key Project of Support Program for Excellent Youth Scholars in Colleges and Universities of Anhui Province (No. gxyqZD2016332). It is also supported by University Natural Science Foundation of Anhui Province, China (KJ2014A184) and the Scientific Research Staring Foundation of Chuzhou Univeristy (No. 2014qd016).

## References

[1] Shen, Z., Subbiah, S., Gu, X., et al., "Cloudscale: Elastic Resource Scaling for Multi-tenant Cloud Systems," *Proc. of the 2nd ACM Symposium on Cloud Computing*, Cascais, Portugal, pp. 5:1–5:14 (2011).

[2] Information on http://www.rsclouds.com/index.php/rscloud/

[3] Ritov, Y., Bickel, P. J., Gamst, A. C., et al., "The Bayesian Analysis of Complex, High-Dimensional Models: Can It Be CODA?" *Statistical Science*, Vol. 29, No. 4, pp. 619–639 (2014). doi: 10.1214/14-STS483

[4] Di, S., Kondo, D. and Cirne, W., "Google Host Load Prediction Based on Bayesian Model with Optimized Feature Combination," *Journal of Parallel and Dis-*

*tributed Computing*, Vol. 74, No. 1, pp. 1820–1832 (2014). doi: 10.1016/j.jpdc.2013.10.001

[5] Jules, O., Hafid, A. and Serhani, M. A., "Bayesian Network and Probabilistic Ontology Driven Trust Model for SLA Management of Cloud Services," *Proc. of 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, Luxembourg, pp. 77–83 (2014). doi: 10.1109/CloudNet.2014.6968972

[6] Wu, X., Zhu, X., Wu, G. Q., et al., "Data Mining with Big Data," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 1, pp. 97–107 (2014).

[7] Sharma, B., Chudnovsky, V., Hellerstein, J. L., Rifaat, R. and Das, C. R., "Modeling and Synthesizing Task Placement Constraints in Google Compute Clusters," *Proc. of the 2nd ACM Symposium on Cloud Computing*, Cascais, Portugal, pp. 3:1–3:14 (2011).

[8] Reiss, C., Tumanov, A., Ganger, G. R., et al., "Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis," *Proc. of the Third ACM Symposium on Cloud Computing. San Jose*, CA, USA, pp. 7:1–7:14 (2012).

[9] Khan, A., Yan, X., Tao, S., et al., "Workload Characterization and Prediction in the Cloud: A Multiple Time Series Approach," *Proc. of 2012 IEEE Network Operations and Management Symposium (NOMS)*, Maui, Havaii, USA, pp. 1287–1294 (2012). doi: 10.1109/NOMS.2012.6212065

[10] Barnes, B. J., Rountree, B., Lowenthal, D. K., et al., "A Regression-based Approach to Scalability Prediction," *Proc. of the 22nd Annual International Conference on Supercomputing*, Island of Kos, Greece, pp. 368–377 (2008). doi: 10.1145/1375527.1375580

[11] Roy, N., Dubey, A. and Gokhale, A., "Efficient Autoscaling in the Cloud Using Predictive Models for Workload Forecasting," *Proc. of the 2011 IEEE 4th International Conference on Cloud Computing*, Washington, DC, USA, pp. 500–507 (2011). doi: 10.1109/CLOUD.2011.42

[12] Saripalli, P., Kiran, G. V. R., Shankar, R. R., et al., "Load Prediction and Hot Spot Detection Models for Autonomic Cloud Computing," *Proc. of 2011 Fourth IEEE International Conference on Utility and Cloud Computing*, Melbourne, Australia, pp. 397–402 (2011). doi: 10.1109/UCC.2011.66

[13] Gong, Z., Gu, X. and Wilkes, J., "PRESS: PRedictive Elastic ReSource Scaling for Cloud Systems," *Proc. of the 2010 International Conference on Network and Service Management*, Niagara Falls, Canada, pp. 9–16 (2010). doi: 10.1109/CNSM.2010.5691343

[14] Carrington, L., Snavely, A. and Wolter, N., "A Performance Prediction Framework for Scientific Applications," *Future Generation Computer Systems*, Vol. 22, No. 3, pp. 336–346 (2006). doi: 10.1016/j.future.2004.11.019

[15] Khazaei, H., Mišić, J. and Mišić, V. B., "Performance Analysis of Cloud Computing Centers Using m/g/m/m+ r Queuing Systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 23, No. 5, pp. 936–943 (2012). doi: 10.1109/TPDS.2011.199

[16] Yang, Q., Peng, C., Zhao, H., et al., "A New Method Based on PSR and EA-GMDH for Host Load Prediction in Cloud Computing System," *The Journal of Supercomputing*, Vol. 68, No. 3, pp. 1402–1417 (2014). doi: 10.1007/s11227-014-1097-x

[17] Gu, Z., Chang, C., He, L., et al., "Developing a Pattern Discovery Model for Host Load Data," *Proc. of 2014 IEEE 17th International Conference on Computational Science and Engineering*, Chengdu, China, pp. 265–271 (2014). doi: 10.1109/CSE.2014.78

[18] Gmach, D., Rolia, J., Cherkasova, L. and Kemper, A., "Capacity Management and Demand Prediction for Next Generation Data Centers," *Proc. of International Conference on Web Services, Salt Lake City*, Utah, USA, pp. 1–8 (2007). doi: 10.1109/ICWS.2007.62

[19] Govindan, S., Choi, J., Urgaonkar, B., et al., "Statistical Profiling-based Techniques for Effective Power Provisioning in Data Centers," *Proc. of the 4th ACM European Conference on Computer Systems, Nuremberg*, Germany, pp. 317–330 (2011). doi: 10.1145/1519065.1519099

[20] Wood, T., Cherkasova, L., Ozonat, K., et al., "Profiling and Modeling Resource Usage of Virtualized Applications," *Proc. of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Leuven, Belgium, pp. 366–387 (2008). doi: 10.1007/978-3-540-89856-6_19

***Manuscript Received: Nov. 27, 2015***
***Accepted: May 5, 2016***