Research Journal of Information Technology 7(1): 1-13, 2016
DOI:10.19026/rjit.7.2806
ISSN: 2041-3106; e-ISSN: 2041-3114

Submitted: October 9, 2015 Accepted: December 5, 2015 Published: May 05, 2016

Research Article

A New Workload Recognition Strategy to Improve the Speed of Resource Provisioning in PaaS Layer of Cloud for Real-Time Demands

Babak Esmaeilpour Ghouchani, Azizol Abdullah, Nor Asila Wati Abdul Hamid and Amir Rizaan Abdul Rahiman
Faculty of Computer Science and Information Technologi, UPM Malaysia, Malaysia

Abstract: The real-time system should guarantee that all critical timing constraints will be met in advance. Many distributed systems such as a cloud environment have a nondeterministic structure and it would cause a serious problem for real time, but the user can access a large number of shared resources. Also launching a new resource in the IaaS layer of a Cloud is not instantaneous. Prediction model, risk management in PaaS and monitoring in IaaS are the most important parts that a real-time system should have because they must face a challenge in understanding the system and the behavior of workload completely. The results of analyzing, monitoring and prediction have serious impacts on system reaction. Understanding the workload is an important challenge in all systems and they use different models to identify the types or predict changes over the time. A prediction model must have the ability to produce and shape the pattern of workloads with low overhead. In this study, we propose an enhancement for profiling process with continues Markov chain to make hosts deterministic for users. The effectiveness and the accuracy of the proposed model measured in the evolution part. Also, the number of the failed tasks counted in this new model to show how proposed model is successful.

Keywords: Anomaly detection, cloud computing, prediction model, real-time, time series

INTRODUCTION

In Cloud, a user has an opportunity to access a large number of resources. However, these resources are often shared with other users and many of the available resources are greatly over time (Bible *et al*., year). Also, the delay in IaaS layer for resource initializing may cause a failure or system delay during a process. The system could apparently work well in a period, but it could collapse in certain rare, but possible situations (Bible *et al*., year). The general structure of Cloud environment is depicted in Fig. 1. If all the critical time constraints cannot be verified, it could collapse because scheduling algorithm in core system layer in cloud system does not include specific mechanisms for handling real-time tasks (Wang, 2012). For programs that are running in a cloud, resource provisioning is one of the key issues. Allocating resources far beyond the request have a negative effect on cost and utilization for users and may cause over provisioning problem for providers. Also, allocating resources less than the request have a major impact on system delay and task failure. This means, in order to maximize the application performance, the user must carefully select a subset of the resources and schedule the application to run on these resources before an application is launched to run in the cloud, Dynamic resource scaling is one of the key characteristics that distinguish the Cloud systems from the traditional computing hosts (Kusic and Kandasamy, 2007).

Initialization time for a new virtual instance in PaaS layer of the Cloud is not immediate and it has several minutes delay for hardware resource allocation in IaaS layer hosting platforms. The perspective of the current technologies showed reduction of VM initialization time is possible (Islam *et al*., 2012). Some technologies like streaming VM allows the customer to preview the VM before it is completely ready. The simple solution is to ask all the customers to determine future VM requests, so the cloud service provider in the SaaS layer can prepare all the VMs on time. However, it seems impossible because first, The customers have no duty to propose their schedule. Second, the customers, are unable to know when the computing resources are needed. Third, the combination of customers, is always changing. Fourth, the actual schedule may change at any time (Jiang *et al*., 2011). In the researchers of this study's view, there is only one solution for overcoming the technology limitations and user constraint satisfaction and it is to predict the

Corresponding Author: Babak Esmaeilpour Ghouchani, Faculty of Computer Science and Information Technologi, UPM Malaysia, Malaysia

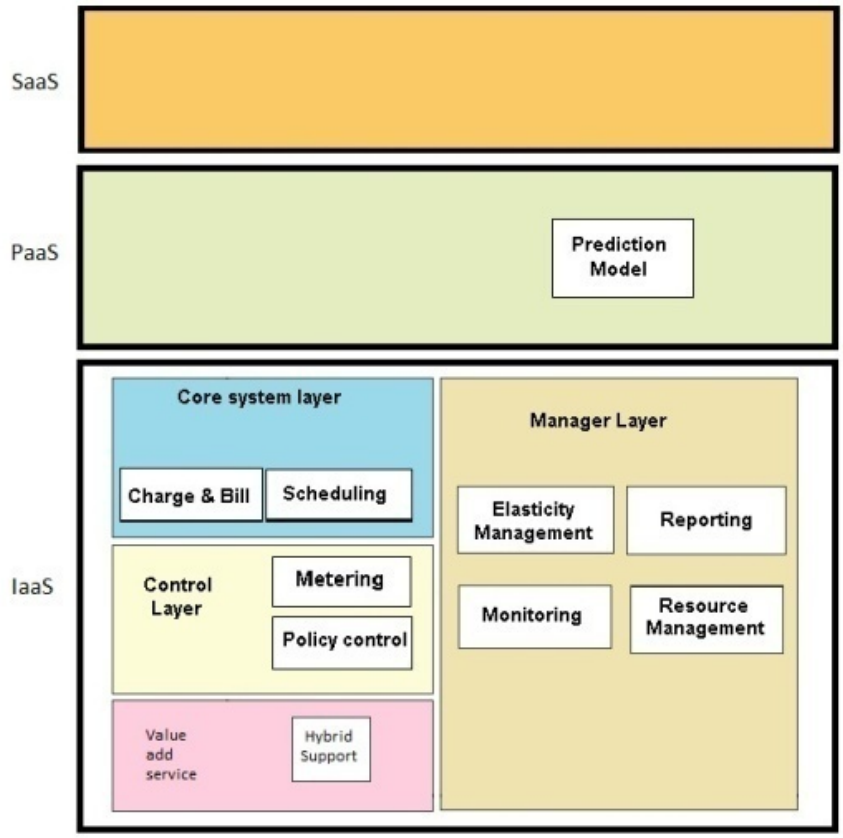


Fig. 1: General structure of cloud environment

demands and prepare the VMs in advance. Predicting and monitoring the user's demand is a fundamental issue when tasks are running on a virtualized system (Jiang *et al*., 2011).

Performance analysis and prediction model need a potent understanding of the system. This is mainly because, the real-time control completely depends on sensory input data and environmental conditions. The system must be analyzable to achieve a desired level of performance and predict the consequences in the workload such as burstiness. Because, a workload has a critical impact on resource provisioning and performance of the cloud-based applications. Most of the predictors and performance analyzers face a challenge to understand workload completely and model them. They use different techniques to identify the type of a workload and predict the changes in that type over the time (Yin *et al*., 2014; Elnaffar and Martin, 2009).

In this study, we will make the following important objectives. In the first step we clearly introduce real-time workload characteristics and constraints, besides we explain, how to find the pattern of workload. In next step we use the pattern to introduce the prototype implementation of our model for performing real-time tasks in Cloud.

MATERIALS AND METHODS

In this section we described how it's possible to define workload. The models and methods have described the real-time workload in different way to make that more predictable.

Real time workload requirements and characteristics: Resource management requires hard timing constraints on tasks' execution and it needs to be supported by the proper prediction model. Predictability can be achieved only by introducing fundamental changes in the basic design paradigm. If a task cannot be guaranteed within its time constraints, the system must notify it in advance, to take alternative actions (Buttazzo, 2011).

Predictability is one of the most important characteristics that a hard real-time system should have. With predictability, the system should be able to predict the evolution of the tasks and guarantee that all critical timing constraints will be meet in advance. The proposed prediction models for real time must used to assist the derivation of actions and the uncertainty of the prediction model must taken into account. As we can see in Fig. 2, all tasks must be finished before a deadline and to ensure avoidance of failure the slack time should be considered in the hard real-time system.

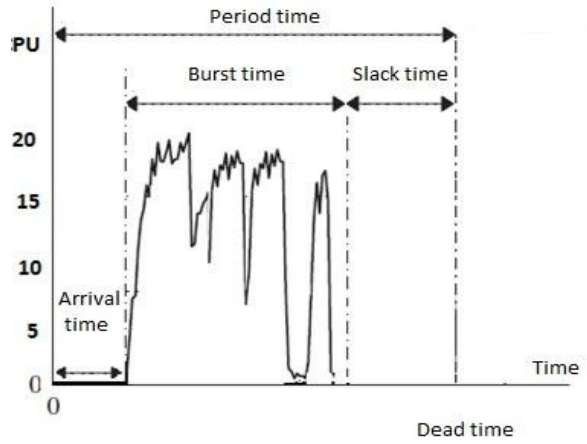


Fig. 2: Periodic real-time workload

The slack time has a positive impact on an opportunity to deal with the uncertainty (Su *et al*., 2013). As in introduction noted the customers have no duty to propose their schedule and the customers are unable to know when the computing resources needed. Thus, the deterministic behavior of a component is desired, because it simplifies the understanding of the real-time behavior and the time evolution of the system is predictable. In all deterministic systems the following issues must be completely clarified:

- Timeline
- Logical reasoning based on a deterministic cause and effect relationship
- Testability of a system (Systems, 2012)

Deterministic behavior is accessible with an estimated probability. The real-time imp lementation can fail to meet this wanted property of determinism for the subsequent reasons (Systems, 2012):

- The base of the computation is not precisely defined.
- When there is a hardware failure.
- The concept of time is unclear.
- The system contains Non-Deterministic design constructs

In case of indeterminism, the user considers the system predictable if it allows computing temporal bounds to its outputs within a reasonable time. In this research, the different characteristics of uncertain and undetermined data involved in this context to handle uncertainty appropriately in real time. First, the system needs the representation of uncertainty on the level of attribute values in the prediction model for the real-time system. Second, the comprehensive models must consider both of the aspects:

- Uncertainty over arbitrary domains for long-term prediction
- The temporal uncertainty that is relevant to the processes of planning and forecasting the events can occur in undetermined way overtime (Eisenreich *et al*., 2011)

All tasks on real-time computer systems require completing the computation within a pre-determined deadline. In this case, the results are computed in a reliable way and are accurate. Furthermore, favorable algorithms provide a high level of locality and parallelism. For large real-time scale architectures it would be very attractive to arrange a common high-level algorithm that solves major problems, dominates real-time concept and has maximum available resources utilization.

The impact of profiling in non-deterministic systems: Most of the prediction models are forecast based on historical knowledge for short-term requests (Mallick *et al*., 2012). The historical knowledge token from the monitoring service, which logs information continuously as a profile in a searchable database (Anderson *et al*., 1997) and the whole process showed in Fig. 3. The proper analysis tool dissects the stored profile information at several levels. The information that are produced by the analysis tools leads users to explain the static and dynamic changes incurred in detail (Verboven *et al*., 2013). Profile creation process has four steps: data granularity, monitoring, processing and storing. This process faces different challenges in all steps to deal starts with, data definition till storing. These challenges are consistency, stability, extra overhead, over sizing, efficiency and the integrity of distributed knowledge in wide system ranges (Anderson *et al*., 1997; Verboven *et al*., 2013 and Ren *et al*., 2010).

The resource request planned after estimating the resources based on a performance model and a workload model. Both performance and workload models use past knowledge for training. They construct

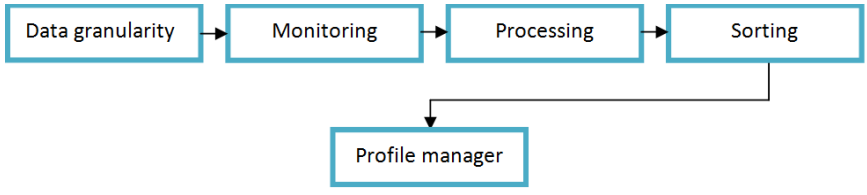


Fig. 3: Profile creation process

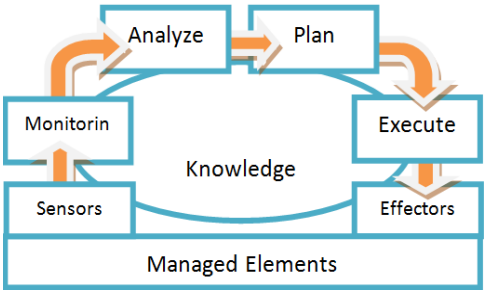


Fig. 4: Profiling and forecasting model

the forecast model based on historical knowledge from local workload traces. The output of the machine learning is the feedback to the resource allocator (Hameed *et al*., 2014). Rafael Weingarten introduced MAPE-K autonomic loop to show how knowledge produced. In this model, many parts and components have a serious impact on the knowledge creation process, which showed in Fig. 4. Rafael divided his model into two important parts: profiling and forecasting. Sensing, monitoring and analyzing belong to the profiling process and the plan, execute and effector parts. As you can see, Fig. 4 belongs to the forecasting process (Hameed *et al*., 2014 and Weingärtner *et al*., 2015). The Plan part in a forecasting process handles optimizing resource utilization and maintaining QoS and QoE (quality of experience). It should take an appropriate action based on its responsibility. The QoE is the behavior that is perceived by end users and it is a way to understand end users (Weingärtner *et al*., 2015).

Methodology for Workload pattern recognition: In this section we discussed about our model and all methods we applied. The prototype has been prepared for cloud to perform real-time tasks.

Workload pattern recognition flow chart: The performance of a prediction model, highly depends on the workload (Hutchison and Mitchell, 2005). Also, this is an attempt to find an accurate characterization that can reproduce the performance from historical workload traces (Zhang *et al*., 2011) such as CPU utilization, waiting time, virtual machine cost, response time, etc. The influence of changes could be determined accurately by using a historical workload to minimize the risk of performance regressions. For this purpose, the characteristic of workload must be well achieved. If well-understood, the provider will be able to model the workload (Hameed *et al*., 2014). In all reviewed papers, there are three techniques to estimate the workload for the next upcoming tasks: First, workload profiling. Second, workload modeling. Third, workload predicting. Statistical estimation techniques are used in profiling to extract reliable workload statistics, although they may not be very appropriate for predicting the workload with large variation. In the second method, many researchers build the model for the workload to compute the prediction for upcoming tasks in the workload by observing the characteristics of the specific applications (Gregoriades and Sutcliffe, 2008; Sun *et al*., 2013 and Calzarossa and Serazzi, 1993). The workload is probably predicted more accurately by the workload model, but this prediction cannot utilized in all applications. Workload prediction performs some specific strategies in a specific prediction model to predict the workload of upcoming tasks (Kuang *et al*., 2014). The main steps for the construction of workload models can be summarized as follows:

Formulation: A workload model is a conceptual description of the tasks parameters (Hutchison and Mitchell, 2005). All prediction models have task decomposition in their frameworks to find the workload pattern (Kousiouris *et al*., 2014). They define specific parameters for their works such as task scheduling, delay, machine resource utilization or required processing nodes to reach the desired performance (Hameed *et al*., 2014; Hutchison and Mitchell, 2005).

Collection of the parameters: The objective of this part is what data the system already has had and what additional data the prediction model will need to collect. This part, directly affects the whole model, because most of the prediction models work based on historical knowledge to forecast short-term user's resource request (Mallick *et al*., 2012). Monitoring cannot collect all parameters and metric's value, It should work on specific metrics and parameters from logging data value during time intervals while the workload is executed (Mallick *et al*., 2012). All prediction models have a raw data filter model in monitoring and they filter unnecessary information from raw data (Jiang *et al*., 2011).

Statistical analysis of the measured data: Monitored metrics used statistical analysis to understand the behavior of the full system to produce applicable outcomes. Monitoring techniques classified as: on-line, off-line and hybrid. The online prediction models use online monitoring techniques and they are more accurate than off-line. They involve lots of overhead because monitoring always calculates the parameters and resets them during the process. In off-line monitoring, there is no instruction to reset the parameters and the monitoring technique uses previously logged data. In the current situation, monitoring is less accurate than the on-line. Hybrid monitoring measures the parameters typically at fixed time intervals (Elnaffar and Martin, 2009). This monitoring technique instructs the model to reset its parameters in every specific time interval. All monitoring techniques consist of the following steps:

- Do collection for elementary analysis to extract the basic system behavior such as growth and descent trend in parameters (Hutchison and Mitchell, 2005)
- Transforming the original value of parameters to a new form and eliminate the outliers data. The most common approach, which used for transforming, is a distribution model (Hutchison and Mitchell, 2005)
- Pick a reasonable amount of knowledge as a sample, because the prediction model suffers from inadequate available performance data to train the machine in machine learning technique. Also, this sample must contain a small group of parameters and it is called data distillation. When the data is unstructured, messy and crude, the data distillation uses the extracting method to select relevant data. This distilled data is exported as a set to the next phase to filter data. Prediction model implements filter or use a normal distribution function to divide data into relevant and irrelevant data (Mallick *et al*., 2012; Jiang *et al*., 2011)
- Classify data for static analysis, because the classifier has a serious impact on monitoring. If the prediction model keeps the classifier active all the time, then it would help online monitoring to reduce the overhead (Elnaffar and Martin, 2009). A robust classification is obtained, when the classifier finds a similarity in some parameters and does it in common intervals

Representativeness: Use some tools for representing a workload. One or more parameters are used to interpret and model workload (Sharma *et al*., 2011).

Decision making: All decision makers have followed the same steps in their process, which is shown in Fig. 5. In the first step, events are monitored by event phase to find certainty or unpredictable events for future demands. In the action phase, the decision maker selects an action based course on certain criteria and find alternative actions. Eventually, the decision is made in the consequence phase and the resulting outcome is sent to resource provisions. These three parts are considered by most of the decision makers in all decisions: First, the available or alternative choices. Second, unpredictable events, which are not under the control of the decision maker. Third, the cost of the decision (Fredericks and Schneider, 2009).

Prediction evaluation: The evaluator measures some of the error metrics as metrics of evaluation. Most of the new prediction models have some checkpoints to evaluate the model during a run time as you can see in Fig. 6(a),. As depicted in Fig. 6(b), if the prediction error is high in one step, then the prediction coefficient will be fitted for the next step. Ideally, the prediction

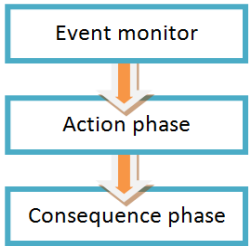
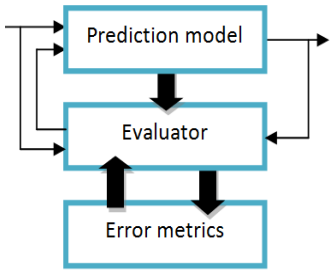
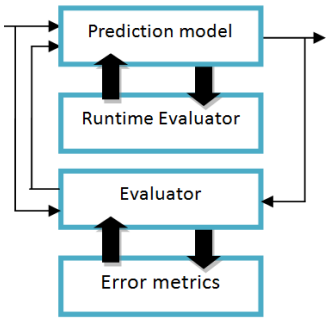


Fig. 5: Flow of decision-making



(a)



(b)

Fig. 6: (a) Prediction model without runtime evaluation; (b) A prediction model for runtime evaluation

error is normally distributed and helps predictor to be stable (Dinda, 2008).

Risk management and analyzer: The resource risk management in PaaS layer has a potential to lead the system to an undesirable situation; then there is a risk of penalty and customer dissatisfaction. Hence, risk analysis can be identified as a proper solution to evaluate these risks. However, the entire risk management process contains many steps and thus needs to be thoroughly discussed. The risk management process consists of the following steps: First, establish the context. Second, identify the risks involved. Third, evaluate each of the identified risks. Fourth, identify techniques to manage each risk. Fifth, create, implement and review the risk management plan (García *et al*., 2014).

The overall model for workload recognition: A sequence of events that are usually measured at consecutive times and placed at stable time intervals is

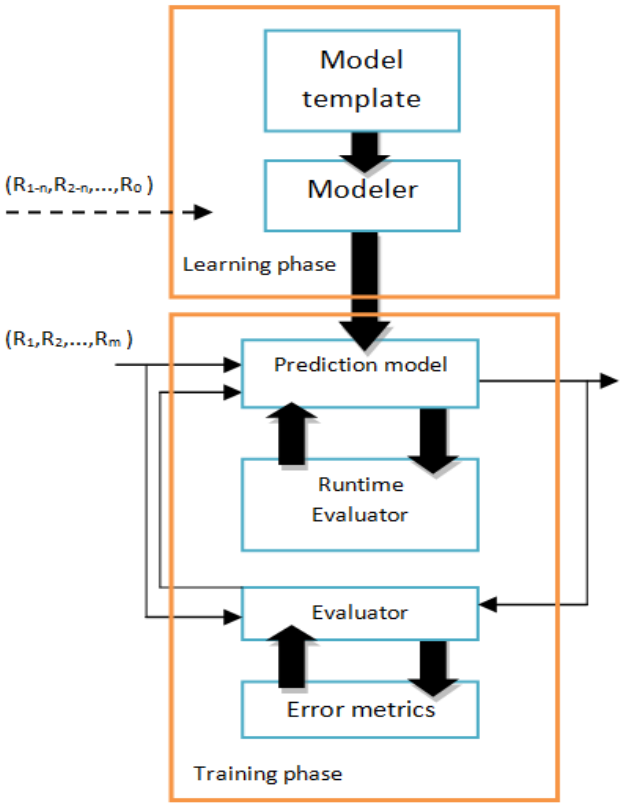


Fig. 7: Overview of the structure of the prediction model

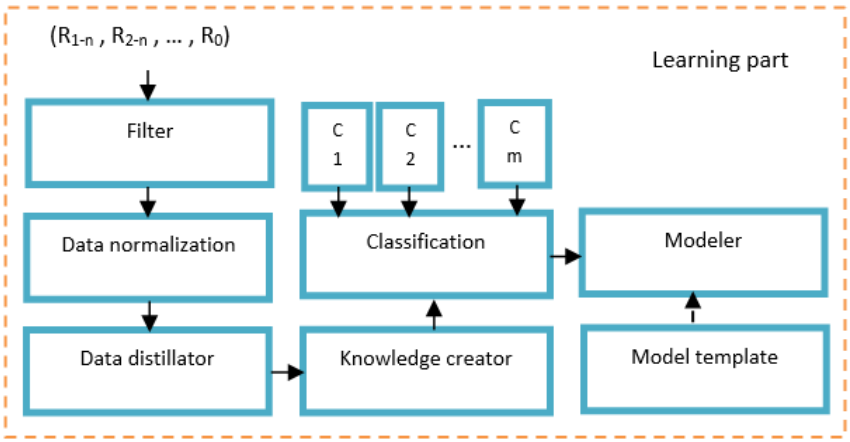


Fig. 8: The overview of the structure of the learning phase

a time series. All predictive methods use the correlation between its near future and the history of the process to make a reliable prediction model for the future events. Recursive models can define the next random variable in the time series by employing previous ones within a proper time window. In time series, a predicted value is presumed based on the real historical values Verboven *et al*. (2013); Deng *et al*. (2012). After reviewing some of the prediction models, the overview of prediction model would be like Fig. 7 (Jiang *et al*., 2011; Thottan *et al*., 2010).

Training: The proposed abstraction flow diagram for an initial prediction model is demonstrated in Fig. 8 based on (Yin *et al*., 2014; Elnaffar and Martin 2009; Mallick *et al*., 2012; Hameed *et al*., 2014; Doulamis *et al*., 2007 and Mian *et al*., 2013). Most of the methodologies can predict the future demand based on the recent request and historical knowledge. All needed metrics are collected during a measurement trace from representative environments to be imported to the learning part. The process in the learning part is the first step for most prediction models that use historical knowledge Eq. (1).

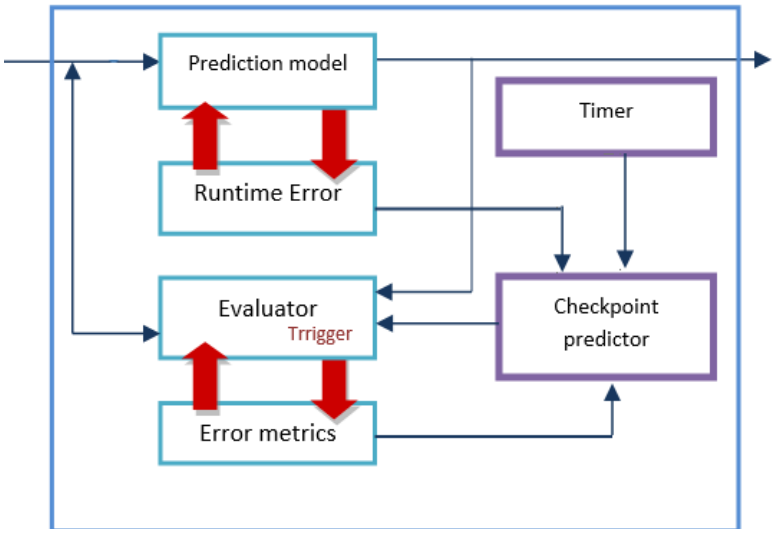


Fig. 9: Overview of the structure of testing and system enforcement

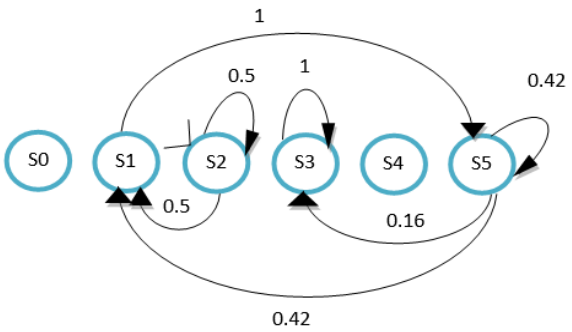


Fig. 10: Markov chain State diagram without task classification

The process begins with a measurement sequence value, which is collected at periodic intervals and then the modeler creates a model based on those values and the model template. The model template contains information about the structure of the desired model of users. These processes shape the training part of the prediction model. The returned model represents a fit to the model structure, which is described in the model template during the measurement sequence:

$$R^1 = (R^{1-n}, R^{2-n}, \ldots . R^0) \quad (1)$$

The result of this part is the initial prediction model, which is formed, based on the training. The user must consider proper learning algorithm to be naturally efficient and effective in the forecasting paradigm.

In this study, we worked with a real data that divided a data stream into three categories: Training, testing (evaluating) and performing. The first part of workload used for the warm-up to train the system to reach a steady-state (Doulamis *et al*., 2007; Dick *et al*., 2014). Some other researchers used a benchmark for training in their prediction model. Then, according to certain reasons such as the volume and distribution of training data, they filtered (Tobaruela *et al*., 2014), smoothed (Sallam *et al*., 2014) or refined (Yin *et al*., 2014 and Elnaffar and Martin, 2009). In the next step, the pure data used to create knowledge; then the results of monitored data, the static analysis and the initial results of performing are used to create knowledge. When the volume of knowledge is very high, knowledge and data are partitioned into mutually exclusive classes. The number of classes defined is various and depends on the scenarios and the users. After a classification, the modeler will try to find the proper model for those classes. Therefore, researchers of this study implemented normalization to avoid out of range data. Afterward, we distilled achieved data from tasks data to extract proper characteristics. Finally, we classified those data in different classes. We used K-means classification in our experiments (Fig. 9 and 10).

Testing and evaluating: The system tested and evaluated the model and during the training predictor used an *m* vector-valued prediction stream for comparison with actual observed values. The predictor also produced error estimations and these estimations will serve to compute a confidence interval for the prediction (Doulamis *et al*., 2007). Most testing parts used evaluation metrics as feedback. Testing and the evaluation metrics evaluated the prediction accuracy and in terms of the metrics computed the error correction and the system will apply them to fit a model. The evaluator compared the actual results with the forecasted results to achieve the accurately fitted model. The *complete process is shown in Fig. 11. The evaluator will produce much overhead in the prediction

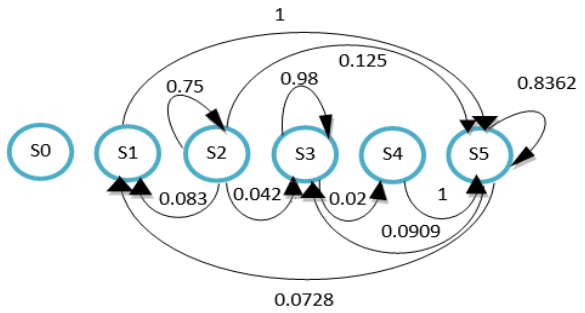


Fig. 11: Markov chain state diagram with five classes

model in each run and the checkpoint is a proper solution for that problem. Some systems have a continuous checkpointing technique to ensure fault-tolerance and accuracy in their prediction model such as Remus in Xen hypervisor (Cully *et al*., 2008). In all checkpoints, the evaluator is triggered to compute evaluation metrics. When the number of checkpoints is still a lot, the system will face an overhead problem. The first important challenge for users is how and when they must define checkpoints. Philipp Leitner suggested checkpoint predictor that showed in Fig. 9. He has a concern about where a prediction should be carried out. The hook is the exact point to trigger the checkpoint. These inputs define this point: First, concrete point that is determined by a user or a timer. Second, prediction error and facts. Third, the retraining strategy of the evaluator for rebuilding the checkpoint prediction. There is a limitation for checkpoint: If no or too little historical data is available, the checkpoint must be suspended by the predictor manager until enough training data has been collected (Leitner *et al*., 2010).

***Failure recovery strategy*:** This strategy designed for unstable Cloud system when Cloud failed to finish tasks without any deadline violation. Resources in the cloud shared among many customers this act caused overloaded resources.

The failed recovery strategy worked based on the number of customers' reduction and the users' share incrimination. The number of users' cut to half when in the first phase, Cloud has failed to perform real-time tasks within a deadline. In the proposed model if Cloud achieves success in any phase then the system starts to share resources between more customers. This method of sharing needs Failure recovery strategy, because if sharing does not go well and some tasks failed to finish performing before a deadline, the number of users will decrease in the next phase based on the following equation:

$$User^k[c+1] = \frac{(user^k[c]+user^k[s])}{2} \quad (2)$$

where,

S : The number of successful phase
c : The number of the current phase
K : The number of classes
$user^s$: The number of users in class k

Model construction for evaluation: In these aspects, for the model construction, Markov chain has been used as a multiple time series prediction models. Markov chain used information from the previous job to consider the sequential dependencies for the next job submission. Markov chain identified as a small set of relevant states and can move from one state to another with certain probabilities (Yin *et al*., 2014 and Mallick *et al*., 2012). Markov model uses some memory and it is possible to describe the whole model by a transition matrix. Markov model has a complicated construction; the number of states must be limited. If this model has many distinct states, then all of them must be considered in the trace. Markov model uses state space models; that means the state of a system contains all the information on the interdependence between the past and the future of the system and it works like some memory. Given the current state, the future evolution becomes independent from the past:

$$\begin{Bmatrix} x(t+1) = Ax(t) + \varepsilon_1(t) \\ y(t) = Cx(t) + \varepsilon_2(t) \end{Bmatrix} \quad (3)$$

$$x(t)^T = (y(t-p), \ldots, y(t-1))^T \quad (4)$$

where, x(t) is an unknown state and x(t+1) estimated based on that.

In Markov model if matrix A in Eq. (1) is stable then the future evolution becomes independent from the past:

$$x(t) = \sum_{k=1}^{\infty} A^{k-1}\varepsilon_1(t-k) \quad (5)$$

Therefore, the output of the model based on Eq. (3) Moreover, (5) is obtained from the following equation if u≥t:

$$y(t) = CA^{u-t}x(t) + \varepsilon_t(u) + \sum_{k=1}^{u-t} CA^{k-1}\varepsilon_1(u-k) \quad (6)$$

The Markov Model was used to explore the sequential correlations in workload pattern changes. This allows us to predict individual VM's workload based on the groups found in the previous step. This study is based on real measurement data collected from the real-time workload in CEA supercomputer, hence provides insights for administrators of the system to realize the typical cloud workload patterns and have a better resources managing.

Researchers of this study, used Markov chain as a prediction model and considered six different states in the proposed model. These states represented in Table 1.

They cover the whole possibility of experiments. Above mentioned states used in the proposed model to perform real-time tasks within a deadline. This model also controls the resource utilization separately in each class. This means each class can play individually. If

Table 1: The Markov chain states

State number	Description	Action
State 0	Model Failed to finish tasks until a deadline	The number of users turns to half the current users and number of resource starts to increase
State 1	40%<Utilization<100%	Start to absorb more resources and share them with more users
State 2	0%≤Utilization≤40%	Resource share between more users
State 3	0%≤Utilization≤100% and model reach maximum user sharing	System starts to adjust the number of resources to help the system to reach optimum utilization
State 4	Current Utilization >100% and the previous utilization is less than 100%	Make an average for number of resources between these two situations
State 5	Utilization = 100% and system is overloaded	Start to absorb more resources

Table 2: Number of failed tasks during 5 test performs

	Number of failed task in different test				
Model	First run	Second run	Third run	Fourth run	Fifth run
Markov chain without classification	0	0	0	0	0
Markov chain with 5 classes	0	0	0	0	0
Markov chain with 10 classes	0	0	0	0	0
Mean model without classification	583	0	186	107	0
Mean model with 5 classes	258	0	0	0	0
Mean model with 10 classes	225	0	0	0	0
Auto correlation without tasks classification	628	0	187	108	0
Auto correlation with 5 classes	296	45	0	0	0
Auto correlation with 10 classes	225	30	0	0	0

the results show any class of visited the 0 state in the Markov chain state diagram; it means proposed model failed to perform tasks within a deadline.

RESULTS

In this section, we show how CloudSim was set to implement our prototype. Also, we evaluate that prototype with the achieved results to show how the proposed prediction model can predict the future demand.

CloudSim setting: In this experiments, we settled a cloud environment in CloudSim and we considered five data centers to provide a large number of available resources in the resource pool. VMs are considered to have the same specification with VM in Amazon EC2. This experiment performed for 24 times to reach the maximum user sharing. Also, we considered 80% for stable point for resource utilization. System distillate data from received tasks based on the size of cloudlet and the number of requested CPUs. The maximum number of users considered a million (a big amount of users) then model with or without classification has the best effort to share resources among these customers. During a learning part, we used a simple time series MA(2) to reconsider some resources separately in each class. Also, VMs will be allocated exclusively to each class. At the end of the phase, each class would be in one of the six situations that are defined in Table 1. For evaluation part, we implemented Markov chain model to turn Cloud to a deterministic host for real-time tasks (Table 2).

Evaluation metrics: In this study, according to the objectives that we've been looking for, several metrics computed and compared to evaluate the performance of our model. The first objective of this research was to design a model that would ensure that, in the face of real-time tasks does not violate the time limit. For this reason, the number of time limit violation in all model reviewed and compared. The second objective is truly important, we want to know how our model is efficient and following metrics have been studied. The RMSE were considered to evaluate the efficiency of models and R^2 were used to determine how models are accurate. Also, the number of users: to understand which models are successful in sharing resources among more users, have been used as an evaluation metric, as well as the average of CPU utilization has been considered to understand the correlation between tasks failure and utilization.

Evaluation results: In the first experiment, we performed model without any classification and the following transition matrix Eq. (7) and state diagram have been achieved (Fig.10) after 24 time performs:

$$Transition\ Matrix \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.00 \\ 0 & 0.50 & 0.50 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.42 & 0.16 & 0 & 0 & 0.42 \end{bmatrix} \quad (7)$$

This experiment clearly shows our model can perform tasks within a time limit without any deadline violation because it has never met the state zero. As we explained before in Table 1; state zero happens when we have a deadline violation.

In next experiments we evaluate our model with tasks classification in the first step we distributed our tasks in 5 different classes. The state diagram (Fig. 11) and the transition matrix have been achieved as follow Eq. (8):

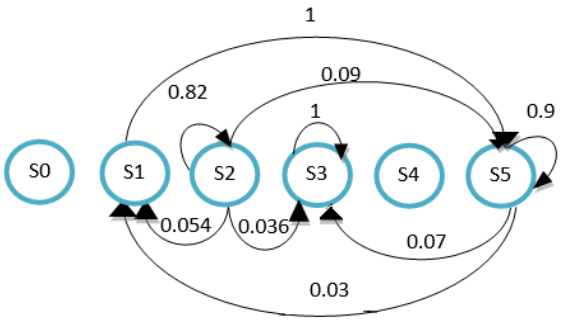


Fig. 12: Markov chain state diagram with 10 classes

Transition matrix

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.00 \\ 0 & 0.083 & 0.750 & 0.042 & 0 & 0.125 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0.0728 & 0 & 0 & 0.0909 & 0.8362 \end{bmatrix} \quad (8)$$

The results show the proposed model has never visited the state zero the same as the previous experience we did not have any failed tasks. For the third experience, we performed our model with ten classes then the following results have been achieved Eq. (9) (Fig. 12). The third experiment results also show the model with ten classes has never visited the state zero:

Transition Matrix

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.00 \\ 0 & 0.054 & 0.82 & 0.036 & 0 & 0.09 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.03 & 0 & 0.07 & 0 & 0.9 \end{bmatrix} \quad (9)$$

We used some non-feedback based model for training processes such as autocorrelation and Mean model for prediction. These models used with the maximum user sharing and performed for five times as like as the proposed model. Also, the classification has been used to improve the results, but the achievements show very clear the autocorrelation and Mean model cannot guarantee the deadline violation never happen. Also, the results show the recovery plan to avoid failure in all model works well. The whole model after five times performing reached to zero number of failed tasks. In a subsequent experiment, the number of end users after five times perform were identified.

The Table 3 shows how the proposed model is successful to share resources among more users. As the results show the failed recovery strategy decreased the number of end users until the Cloud system reaches a steady state. The Markov models-with or without classification-remain steady during five times performing.

Table 4 shows how many CPUs our real-time tasks received during these experiments. It is very clear the failure recovery strategy increased the resource share to solve the overloaded CPUs problem. Otherwise, proposed models (the Markov models with or without classifications) has reached to the state 3 in the Markov chain state diagram. The state 3 has a duty to stabilized Cloud system and establish load balancing. Also in all the experiments with ten classes of tasks, less number of CPUs absorbed, although according to the Table 4 they reached better user sharing (Table 5).

In all the experiments that have been tried the average of CPU utilization placed around 80%, because we reached to the following table in different tests. Hence, to find which model has better convergence (more than 80%) for utilization, all of the models evaluated. Table 6 shows the mean model and

Table 3: Number of users that resources are shared among them

	Number of end users in different test				
Model	First run	Second run	Third run	Fourth run	Fifth run
Markov chain without classification	1, 000, 000	1, 000, 000	1, 000, 000	1, 000, 000	1, 000, 000
Markov chain with 5 classes	1, 000, 000	1, 000, 000	1, 000, 000	1, 000, 000	1, 000, 000
Markov chain with 10 classes	1, 000, 000	1, 000, 000	1, 000, 000	1, 000, 000	1, 000, 000
Mean model without classification	1, 000, 000	500, 000	500, 000	250, 000	125, 000
Mean model with 5 classes	1, 000, 000	268, 817	268, 817	268, 817	268, 817
Mean model with 10 classes	1, 000, 000	617, 284	617, 284	617, 284	617, 284
Auto correlation without tasks classification	1, 000, 000	500, 000	500, 000	250, 000	125, 000
Auto correlation with 5 classes	1, 000, 000	294, 118	222, 892	222, 892	222, 892
Auto correlation with 10 classes	1, 000, 000	617, 284	601, 411	601, 411	601, 411

Table 4: The number of allocated CPUs

	Number of allocated CPUs to real-time workload				
Model	First run	Second run	Third run	Fourth run	Fifth run
Markov chain without classification	22946	22882	23010	22850	22978
Markov chain with 5 classes	23290	23162	23226	23226	23114
Markov chain with 10 classes	15804	15484	15884	16540	18204
Mean model without classification	26	2272	24386	21378	15010
Mean model with 5 classes	186	570	3306	8234	13994
Mean model with 10 classes	810	1310	2428	3420	4524
Auto correlation without tasks classification	10	496	22786	20546	15138
Auto correlation with 5 classes	170	332	1626	4890	9466
Auto correlation with 10 classes	810	1134	1660	2172	2540

Table 5: Situations based on utilization

Utilization	Risk
Utilization<0.4	No risk for task failure and ready to share between more customers
0.4<Utilization<0.8	Low risk for task failure, and ready to absorb a little bit more resources and share resources between more users
0.8<Utilization≤1	High risk for task failure, and ready to absorb more resources and share resources between more users
Utilization = 1 and there is a long waiting task queue	Very high risk for task failure No more resource sharing and system need more resources to solve the overloading problem

Table 6: The average of utilization

	The average of utilization percentage				
Model	First run	Second run	Third run	Fourth run	Fifth run
Markov chain without classification	80.93	79.8	80.93	79.08	80.36
Markov chain with 5 classes	81.11	83.87	83.56	83.08	83.11
Markov chain with 10 classes	131.15	112.07	101.77	87.64	83.78
Mean model without classification	19707	1085	67.23	50.46	32.45
Mean model with 5 classes	4031.17	1997.32	1396.82	517.13	147.22
Mean model with 10 classes	2150	554.7717	554.77	372.44	334.09
Auto correlation without tasks classification	24678	4721	70.25	53.62	37.84
Auto correlation with 5 classes	5244.83	2337.58	1691.93	1161	424
Auto correlation with 10 classes	2140.209	567.62	47.03	372.25	308.63

The percentage over 100 means overloading and there is a waiting task queue

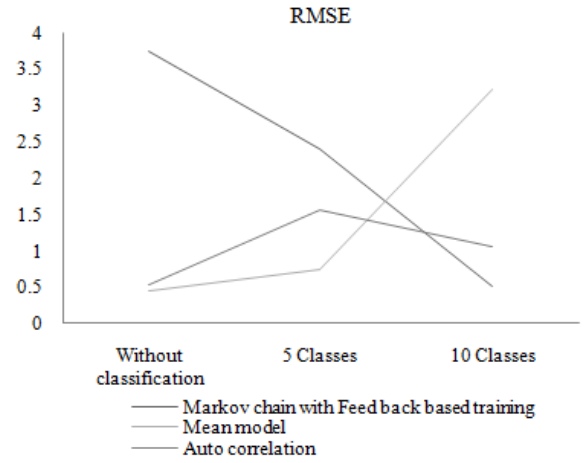


Fig. 13: The amount of RMSE

Table 7: The amout of R^2

Model	R^2
Markov chain without classification	0.7078
Markov chain with 5 classes	0.9321
Markov chain with 10 classes	0.9668
Mean model without classification	0.9901
Mean model with 5 classes	0.5963
Mean model with 10 classes	-0.6372
Autocorrelation without tasks classification	0.9921
Autocorrelation with 5 classes	0.317
Autocorrelation with 10 classes	-0.8480

autocorrelation prediction model have been caused the worth utilization in Cloud host in comparision with Markov chain prediction model.

When we look at the results in Table 6 and 2, Mean and Autocorrelation models were overloaded in first perform and the number of failed tasks are two high. However, failure recovery strategy shows its impact on the results from the second run. The number of deadline violation decreased since utilization had been improved, although it seems there is no convergence of utilization for prediction models based on the Mean and Auto Correlation.

Figure 13, we compute RMSE of models to evaluate which one of models is more efficient than the others. As you can see the Markov chain models with or without classification have better results among other models. The impact of classification shows the number of classes has a positive effect on efficiency.

In the last test models were evaluated in terms of the precision. As we mentioned before in "evaluation metrics"; R^2 is a common evaluation metric to find how much prediction model is accurate. This metric has been computed for all models (Table 7), the nearest R^2 to 1 has the best accuracy.

DISCUSSION

In this study, a new multi-objective model proposed and a new model to predict the demands and anomalies described. This model shared resources to prepare this host ready for performing real-time workload among more customers in Cloud system. The experiments showed feedback based prediction model can perform real-time tasks in Cloud system. However, there is no guarantee for Cloud to perform real-time tasks without any deadline violation. The proposed model can identify the required resources for real-time tasks (i.e., by using the Markov chain, a predictive model and automatic adaptive resource scaling and sharing). Cloud infrastructure providers adopted our approach not only to offer their customers with response time guarantees but also to minimize the resources allocated to the customers. It is difficult to say that task classification had a positive impact in all aspects. The experiments showed all models with task classification had slow convergence to reach the 80% of CPU utilization, but the proposed model improved the prediction efficiency and accuracy.

One of the way to extend this system is to support Real-time economic model on a cloud. Another way is to extend resource scaling strategy, which is currently only used to attract inexpensive resources, absorb idle resources and migrate them to associate with overloaded resources in advance in order to overcome the virtual machine boot-up latency problem is another open way for reasearchers.

ACKNOWLEDGMENT

Authors confirms there is no any Conflict of Interest

REFERENCES

Anderson, J.M., L.M. Berc, J. Dean, S. Ghemawat, M.R. Henzinger and *et al*., 1997. Continuous profiling: Where have all the cycles gone? ACM Trans. Comput. Syst., 15(4): 357-390.

Bible, C.C., W. Publishing and J. Wiley, year. Cloud Computing Bible. Wiley Publishing.

Buttazzo, G.C., 2011. Hard RealTime Computing Systems: Predictable Scheduling Algorithms and Applications. 3rd Edn., Springer Publishing Company.

Calzarossa, M. and G. Serazzi, 1993. Workload characterization: A survey. Proceedings of the IEEE, 81(8): 1136-1150.

Cully, B., G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson and A. Warfield, 2008. Remus: High availability via asynchronous virtual machine replication. Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI'08), pp: 161-174.

Deng, Y., X. Meng and J. Zhou, 2012. Self-similarity: Behind workload reshaping and prediction. Future Generat. Comput. Syst., 28(2): 350-357.

Dick, S., O. Yazdanbaksh, X. Tang, T. Huynh and J. Miller, 2014. An empirical investigation of Web session workloads: Can self-similarity be explained by deterministic chaos? Inform. Process. Manage., 50(1): 41-53.

Dinda, P.A., 2008. Design, implementation and performance of an extensible toolkit for resource prediction in distributed systems. Ieee t. parall. Distribut. Syst., 17(2): 160-173.

Doulamis, N., A. Doulamis, A. Litke, A. Panagakis, T. Varvarigou and E. Varvarigos, 2007. Adjusted fair scheduling and non-linear workload prediction for QoS guarantees in grid computing. Comput. Communi., 30(3): 499-515.

Eisenreich, K., G. Hackenbroich, V. Markl, P. Rosch and R. Schulze, 2011. Handling of uncertainty and temporal indeterminacy for what-if analysis. In: Castellanos, M., U. Dayal and V. Markl (Eds.): BIRTE 2010, LNBIP 84, Springer-Veralge Berlin Heidelberg, pp: 100-115.

Elnaffar, S. and P. Martin, 2009. The psychic-skeptic prediction framework for effective monitoring of DBMS workloads. Data Knowl. Eng., 68(4): 393-414.

Fredericks, E. and D.R. Schneider, 2009. Annals of Information Systems. In: Frada, B., B. Patrick and Z. Arkady (Eds.): Springer New York Dordrecht Heidelberg London.

García, A.G., I.B. Espert and V.H. García, 2014. SLA-driven dynamic cloud resource management. Future Generat. Comput. Syst., 31(1): 1-11.

Gregoriades, A. and A. Sutcliffe, 2008. Workload prediction for improved design and reliability of complex systems. Reliab. Eng. Syst. Safety, 93(4): 530-549.

Hameed, A., A. Khoshkbarforoushha, R. Ranjan, P.P. Jayaraman, J. Kolodziej and *et al*., 2014. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. Computing, DOI 10.1007/s00607-014-0407-8.

Hutchison, D. and J.C. Mitchell, 2005. Lecture Notes in Computer Science. Springer, Berlin.

Islam, S., J. Keung, K. Lee and A. Liu, 2012. Empirical prediction models for adaptive resource provisioning in the cloud. Future Generat. Comput. Syst., 28(1): 155-162.

Jiang, Y., C.s. Perng, T. Li and R. Chang, 2011. ASAP: A self-adaptive prediction system for instant cloud resource demand provisioning. Proceeding of the IEEE 11th International Conference on Data Mining (ICDM), pp: 1104-1109.

Kousiouris, G., A. Menychts, D. Kyriazis and S. Gogouvitis, 2014. Dynamic, behavioral-based estimation of resource provisioning based on high-level application terms in Cloud platforms. Future Generat. Comput. Syst., 32: 27-40.

Kuang, S.R., K.Y. Wu, B.C. Ke, J.H. Yeh and H.Y. Jheng, 2014. Efficient architecture and hardware implementation of hybrid fuzzy-Kalman filter for workload prediction. Integrat. VLSI J., 47(4): 408-416.

Kusic, D. and N. Kandasamy, 2007. Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing systems. Proceeding of the IEEE International Conference on Autonomic Computing (ICAC '06), pp: 74-83.

Leitner, P., B. Wetzstein, F. Rosenberg, S. Dustdar and F. Leymann, 2010. Runtime Prediction of Service Level Agreement Violations for Composite Services. In: Dan, A., F. Gittler and F. Toumani (Eds.): ICSOC/ServiceWave 2009, LNCS 6225, Springer-Verlage Berlin Heidelberg, pp:176-186.

Mallick, S., G. Hains and C.S. Deme, 2012. A resource prediction model for virtualization servers. Proceeding of the 2012 International Conference on High Performance Computing and Simulation (HPCS), pp: 667-671.

Mian, R., P. Martin and J.L. Vazquez-Poletti, 2013. Provisioning data analytic workloads in a cloud. Future Generat. Comput. Syst., 29(6): 1452-1458.

Ren, G., E. Tune, T. Moseley, Y. Shi, S. Rus, R. Hundt, 2010. Google-wide profiling: A continuous profiling infrastructure for data centers. IEEE Micro (2010), pp: 65-79.

Sallam, A., K. Li, A. Ouyang and Z. Li, 2014. Proactive workload management in dynamic virtualized environments. J. Comput. Syst. Sci., 80(8): 1504-1517.

Sharma, B., V. Chudnovsky, J.L. Hellerstein, R. Rifaat and C.R. Das, 2011. Modeling and synthesizing task placement constraints in Google compute clusters. Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC '11), pp: 1-14.

Su, S., J. Li, Q. Huang, X. Huang, K. Shuang and J. Wang, 2013. Cost-efficient task scheduling for executing large programs in the cloud. Parallel Comput., 39(4): 177-188.

Sun, Y.S., Y.F. Chen and M.C. Chen, 2013. A workload analysis of live event broadcast service in cloud. Procedia Comput. Sci., 19: 1028-1033.

Systems, A.R., 2012. Advances in Real-Time Systems. Springer Heidelberg Dordrecht London, New York.

Thottan, M., G. Liu and Ji, C. Anomaly, 2010. Algorithms for Next Generation Networks. In: Cormode, G. and M. Thottan, (Eds.): Algorithms for Next Generation Networks, Computer Communications and Networks. Springer London, pp: 239-261.

Tobaruela, G., W. Schuster, A. Majumdar, W.Y. Ochieng, L. Martinez and P. Hendrickx, 2014. A method to estimate air traffic controller mental workload based on traffic clearances. J. Air Transp. Manage., 39: 59-71.

Verboven, S., K. Vanmechelen and J. Broeckhove, 2013. Black box scheduling for resource intensive virtual machine workloads with interference models. Future Generat. Comput. Syst., 29(8): 1871-1884.

Wang, W., 2012. Dynamic Reconfi guration in Real-Time Systems. In: Weixun, W., M. Prabhat and R. Sanjay (Eds.): Springer New York Heidelberg Dordrecht London.

Weingärtner, R., G.B. Bräscher and C.B. Westphall, 2015. Cloud resource management: A survey on forecasting and profiling models. J. Network Comput. Appl., 47: 99-106.

Yin, J., X. Lu, H. Chen, X. Zhao and N.N. Xiong, 2014. System resource utilization analysis and prediction for cloud based applications under bursty workloads. Inform. Sci., 279: 338-357.

Zhang, Q., J.L.Hellerstein and R. Boutaba, 2011. Characterizing task usage shapes in google's compute clusters. Proceeding of the Large Scale Distributed Systems and Middleware Workshop (LADIS 2011).