Journal of King Saud University – Computer and Information Sciences (2018) 30, 404–415

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

Predictive Cloud resource management framework for enterprise workloads

Mahesh Balaji [a,*], Ch. Aswani Kumar [b], G. Subrahmanya V.R.K. Rao [a]

[a] Global Technology Office, Cognizant Technology Solutions, Chennai, India
[b] School of Information Technology & Engineering, VIT University, Vellore, India

Received 11 May 2016; revised 21 October 2016; accepted 23 October 2016
Available online 29 October 2016

KEYWORDS

Cloud computing;
Predictive modeling;
Resource management;
Enterprise workload

Abstract The study proposes an innovative Predictive Resource Management Framework (PRMF) to overcome the drawbacks of the reactive Cloud resource management approach. Performance of PRMF was compared with that of a reactive approach by deploying a timesheet application on the Cloud. Key metrics of the simulated workload patterns were monitored and analyzed offline using information gain module present in PRMF to determine the key evaluation metric. Subsequently, the best-fit model for the key evaluation metric among Autoregressive Integrated Moving Average (ARIMA) ($1 \leqslant p \leqslant 4$, $0 < d < 2$, $1 \leqslant q \leqslant 4$), exponential smoothening (Single, Double & Triple) and Hidden Markov Model present in the PRMF library were determined. Best-fit model was used for predicting key evaluation metric. During real time, the validation module of PRMF would continuously compare the actual and predicted key evaluation metric. Best-fit model would be re-evaluated if 95% confidence level of the predicted value breaches the actual metric. For experiments performed in the current study, Request Arrival and ARIMA (2, 1, 3) were found to be the key evaluation metric and the best-fit model respectively. Proposed predictive approach performed better than the reactive approach while provisioning/deprovisioning instances during the real time experiments.

1. Introduction

Enterprises have started to embrace Cloud as an IT delivery model by migrating majority of their application workloads on to it. These applications can be categorized as read, write and read–write intensive based on their resource consumption. Alternatively they could also be grouped into on-off, high-growth, periodic and aperiodic-burst depending on their workload (Mell and Mell, 2011; Mulia et al., 2013; Armbrust et al., 2010). Even though Cloud provides reliable resources, it also poses a significant challenge in allocating and managing resources dynamically across applications. There are several

* Corresponding author.
E-mail addresses: mahesh.balaji@cognizant.com (M. Balaji), cherukuri@acm.org (Ch. Aswani Kumar), subrahmanyavrk.rao@cognizant.com (G. Subrahmanya V.R.K. Rao).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.jksuci.2016.10.005
1319-1578

optimized resource management methods that are already available such as linear scheduling strategy, policy-based resource allocation and location-aware dynamic resource allocation (Anuradha and Sumathi, 2014; Bharti and Kaur, 2014; Mary, 2013). Majority of the above mentioned studies ensure Cloud resources are optimally used from a provider perspective. However, they fail to validate whether the provisioned resources were optimally used by a particular application. Both, the Cloud infrastructure service provider and the consumer, get optimal benefits when the appropriate amount of resources for an application is provisioned/deprovisioned on-demand. It must be noted that in a Cloud charging model, the end-user perspective is a pivotal evaluation metric to sustain business.

Existing resource management methods such as those provided by Amazon Web Services (AWS) Auto Scaling, Windows Azure Autoscaling Application Block, Google AppEngine orchestration framework and IBM Cloud virtual application pattern are reactive. Reactive approach is an effective way to improve system availability, optimize costs and reduce latency. However, it exhibits an inherent lag between identifying resource-demands and provisioning, which could lead to under or over provisioning. Under-provisioning will cause Service Level Agreements (SLA) violation while over provisioning will result in resource wastage and higher cost (Arlitt and Williamson, 1997; Gong et al., 2010; Herbst et al., 2013). Moreover reactive approach might not be able to fully address a wide range of enterprise use-cases like rapid spikes, stoppage and variable traffic patterns. Some of these challenges could be addressed by preemptive scaling, but this is often undesirable and may lead to instance trashing (scale up–scale down oscillations). An alternative option would be to run more servers than required, which is not optimal from a cost perspective (Varia, 2010; Mao et al., 2010; Bunch et al., 2012; Kochut et al., 2011). A more effective option would be using a predictive approach, which is an emerging area of research.

It must be noted that reports on predictive resource provisioning have made cardinal assumptions like (1) presence of homogeneous Cloud infrastructure (2) access to the physical hardware and (3) linear relationship between key metrics. However, a typical enterprise consists of diverse applications with different priorities, performance and resource requirements. Despite extensive studies on resource provisioning, literature has rarely reported studies about heterogeneity of the workload and infrastructure (Hu et al., 2010). Incompatibility that exists between the workload requirements and resource present in the provisioned instances will lead to long scheduling delays and overheads. Monitoring appropriate metrics to identify the need to provision/deprovision resources is a key factor. However, majority of the Cloud providers do not grant access to the physical hardware on which the applications are deployed. Hence, acquiring key metrics is a difficult proposition. Non-stationary nature of the enterprise workload patterns and requirement of a heterogeneous environment implies that most of these studies would not be viable in the real world context (Khajeh-Hosseini et al., 2010).

Above mentioned limitations could be overcome by adopting an appropriate framework to monitor, acquire and analyze key metrics involved in the Cloud resource management. An ideal framework should identify a workload, suitable metric and an appropriate algorithm to provision/deprovision resources on-demand. Given the diverse nature of enterprise workloads, identifying key evaluation metric(s) and an appropriate algorithm to provision/deprovision resources would be an efficient alternative to an ideal framework. Key contribution of the current study was to develop such a resource management framework (PRMF), which would employ statistical feature selection technique to identify key evaluation metric for a given workload pattern and determine a best-fit algorithm from a set of algorithms to provision/deprovision virtual machine (VM) instances using a predictive approach. The current study applied simulated workload patterns on an on-off application and evaluated the performance of PRMF as compared to a reactive approach. Results of the study have been presented along with the ways of extending it to other workload patterns.

2. Related work

Reports on the Predictive Resource Management Frameworks could be categorized as studies that use (a) reactive metrics, linear modeling and simulation (Iqbal et al., 2009, 2010; Xiong et al., 2011; Bennani and Menasce, 2005; Bhulai et al., 2007; Shi et al., 2011; Bankole and Ajila, 2013), (b) predictive metrics, linear modeling and simulation (Wang et al., 2014; Tammaro et al., 2011), (c) predictive algorithms and work on non-stationary workloads (Han et al., 2013; Deng et al., 2013; Calheiros et al., 2011, 2014), and (d) intrusive methods (Elprinc, 2013; Kim et al., 2011; Ali-Eldin et al., 2012; Kouki et al., 2014, 2015).

Approach	Metrics
Real-time load balancer logs analysis (Iqbal et al., 2009, 2010)	Response time
Machine learning techniques (Xiong et al., 2011)	Memory and CPU
Queue and combinatorial search techniques (Bennani and Menasce, 2005)	Response time and throughput
Queue based approach (Bhulai et al., 2007)	Response time
linear and queuing model (Shi et al., 2011)	Response time
Machine learning and Linear Regression (Bankole and Ajila, 2013)	Response time, Throughput and CPU utilization
Queue theory and exponential smoothing (Wang et al., 2014)	Arrival rate
Data-mining algorithms (Tammaro et al., 2011)	Arrival rate & Teardown time
ARIMA model Han et al., 2013	Arrival rate/departure rate
ARIMA model Deng et al., 2013; Calheiros et al., 2011, 2014	Request processed/ through put
Policy based provisioning (Elprinc, 2013)	CPU, Memory and response time
Machine learning technique (Kim et al., 2011)	Execution time/ Throughput
Queue based provisioning and deprovisioning (Ali-Eldin et al., 2012)	Request processed/ Throughput
Policy based provisioning and deprovisioning (Kouki et al., 2014, 2015)	Workload/Throughput

Non-stationary workload is a salient feature of an enterprise application. It must be noted that the linear modeling techniques would not be able to accommodate larger variance in the workload. PRMF consists of algorithms that could accurately predict non-stationary workloads and therefore would be able to overcome the limitations of the aforementioned studies (Iqbal et al., 2009, 2010; Xiong et al., 2011; Bennani and Menasce, 2005; Bhulai et al., 2007; Shi et al., 2011; Bankole and Ajila, 2013; Wang et al., 2014; Tammaro et al., 2011).

While a few studies have attempted to handle non-stationary workloads (Han et al., 2013; Deng et al., 2013; Calheiros et al., 2011, 2014), all of the above-mentioned studies have fixed metrics to identify the resource need and have no real-time feedback. In a marked difference, the current study (PRMF) derived the proactive metric based on the workload using statistical feature selection technique. PRMF is designed to support dynamic nature of the application workload using a real-time feedback mechanism.

Deprovisioning is also an important phenomena of the Cloud resource management and has significant bearing on Cloud charging model. All the studies mentioned above apart from Ali-Eldin et al. (2012), Kouki et al. (2014, 2015) have rarely reported on resource deprovisioning. While the current study is non-intrusive unlike (Elprinc, 2013; Kim et al., 2011; Ali-Eldin et al., 2012; Kouki et al., 2014, 2015), PRMF has considered both provisioning/deprovisioning for an enterprise application.

3. Proposed approach

Reactive rule-based techniques and majority of the reported predictive approaches use fixed metrics (such as CPU, memory and response time) irrespective of the workload to provision/deprovision resources as described in algorithm 1 (Varia, 2010; Mao et al., 2010; Bunch et al., 2012; Kochut et al., 2011; Iqbal et al., 2009, 2010; Xiong et al., 2011; Bennani and Menasce, 2005; Bhulai et al., 2007; Shi et al., 2011; Bankole and Ajila, 2013; Wang et al., 2014; Tammaro et al., 2011; Han et al., 2013; Deng et al., 2013; Calheiros et al., 2011, 2014; Elprinc, 2013; Kim et al., 2011; Ali-Eldin et al., 2012; Kouki et al., 2014, 2015). Deploying fixed metrics for varying workload may not ensure optimal provisioning/deprovisioning of resources. Therefore, the above mentioned studies might not perform as well as expected. As a marked difference, the current study, dynamically derives the key evaluation metric using a statistical technique. Such a methodology along with the feedback loop would enable the proposed framework to determine appropriate metrics in case of large variance in the workload. This key evaluation metric would be subsequently used to provision/deprovision resources as described in algorithm 2. It must also be noted that models are chosen based on the best-fit and might change according to the metrics.

Algorithm 1: Provisioning and deprovisioning in a rule-based approach

Step 1. Assign user request to VM instances
Step 2. Continuously monitor individual instances using metrics like CPU and memory
Step 3. If the monitored metrics imply insufficient infrastructure capacity to process the assigned request as per the governance policy
  Step 3a. Hold or terminate the user request
  Step 3b. Provision additional VM instances
  Step 3c. Resubmit the request to the new VM instance if the request was on hold
Step 4. Else assigned VM would continue processing the user requests
Step 5. If the resources are deemed surplus as defined by the governance policy
  Step 5a. Deprovision the excess instances
Step 7. Else retain all the active instances
Step 8: Repeat steps 1–8 for all the user requests

Algorithm 2: Provisioning and deprovisioning in the proposed approach

Step 1. Dry run application/workload using policy-driven approach and collect all available metrics along with the corresponding instance-state (provisioned/de-provisioned)

Offline analysis

Step 2. Select business relevant evaluation metrics from the collected metrics
Step 3. Identify the most impactful metric for provisioning/deprovisioning instances using statistical feature-selection technique
Step 4. Build the mapping between workload and the instance state based on the identified significant metric

Real time implementation

Step 5. Predict the identified significant metric using an appropriate model
Step 6. Monitor the real time significant metric and continuously compare it with the predicted metric
Step 7. If the real time metric is not within $\pm 95\%$ confidence level of the predicted metric
  Step 7a. Choose an appropriate model using the real time value
Step 8. Else continue
Step 9. If the current infrastructure capacity is deemed insufficient to process the predicted workload based on the mapping built in step 4
  Step 9a. Provision additional VM instances
Step 10. Assign user request to VM instances
Step 11. If the resources are deemed surplus to process the predicted workload
  Step 11a. Deprovision the excess VM instances
Step 12. Else retain all the active instances
Step 13. Repeat steps 5–12 for all the user requests

The experimental setup and the data flow of the proposed approach are detailed in the subsequent section.

4. Experimental setup and data flow

Timesheet is a typical example of a global enterprise On/Off application, which could make use of the Cloud features such as pay-per-use and elasticity most efficiently (Mulia et al., 2013). Very large number of end users and short active period make it an ideal candidate for the study. In the current study Amazon AWS t2.micro instance, Time and Expense Management System (TEMS), HP JMeter, AWS Elastic Load balancer, MYSQL DB etc., was used to build the experimental setup as shown in Fig. 1.

TEMS application was set up using Apache Webserver with MOD-PHP and MYSQL database, while java was used to setup HP JMeter. Web request from HP JMeter were sent to the application elastic load balancer, which then forwards them to the application server farm. User requests were redirected to the database server based on their actions performed on the TEMS System. Both TEMS and HP JMeter were setup on the AWS t2.micro instances. Detailed metrics of all the instances were acquired, using Amazon Cloud monitoring services (Amazon CloudWatch). Metrics were analyzed offline using PRMF, which consists of a library of time-series/stochastic models and a validation module. While the former module was used to analyze data offline, the latter would facilitate the system to adapt itself to drastic changes in the actual workload pattern by providing a feedback.

4.1. Simulation of end-user workload

Historical data, which comprises of information such as source IP, time stamp, request ID, data size of the request over a period of three years were extracted from the log files of our enterprise timesheet application. Request Arrival was determined by computing the number of request that the time sheet application received per minute. The current study applied Quantile–Quantile (QQ) plot on the Request Arrival to determine its distribution. QQ Plot is a commonly used statistical graph to validate the distributional assumption of a data set. Distribution of the data set is identified by visual inspection of the plot and hence QQ plot is not an objective measure. Therefore, after identifying the statistical distribution using QQ plot, Shapiro–Wilk, a commonly used statistical test, was used to objectively validate the same (Dixon and Massey, 1957).

HP JMeter was used to simulate Web requests, which would follow the historically observed enterprise patterns based on the identified statistical distribution. Current study assumes 200,000 employees filling their time sheet once during a fortnight. The experiments were run to simulate the time sheet filling activity for a duration of four hours. Three separate JMeter scripts were used to generate (1) Consistent load, (2) Short duration rapid spikes and (3) Long duration rapid spikes. Rapid spikes are an enterprise phenomenon, which necessitates appropriate provisioning and deprovisioning of resources. A typical workload pattern would consist of a steady state-load intertwined with short and long spikes. The current study achieved this using shell scripts and a java program as detailed below.

One experimental run consists of applying the workload pattern described above on the TEMS application. For such a run, the consistent load would have 9/12 concurrent users per second (representing ~150 k unique user requests). Mutu-

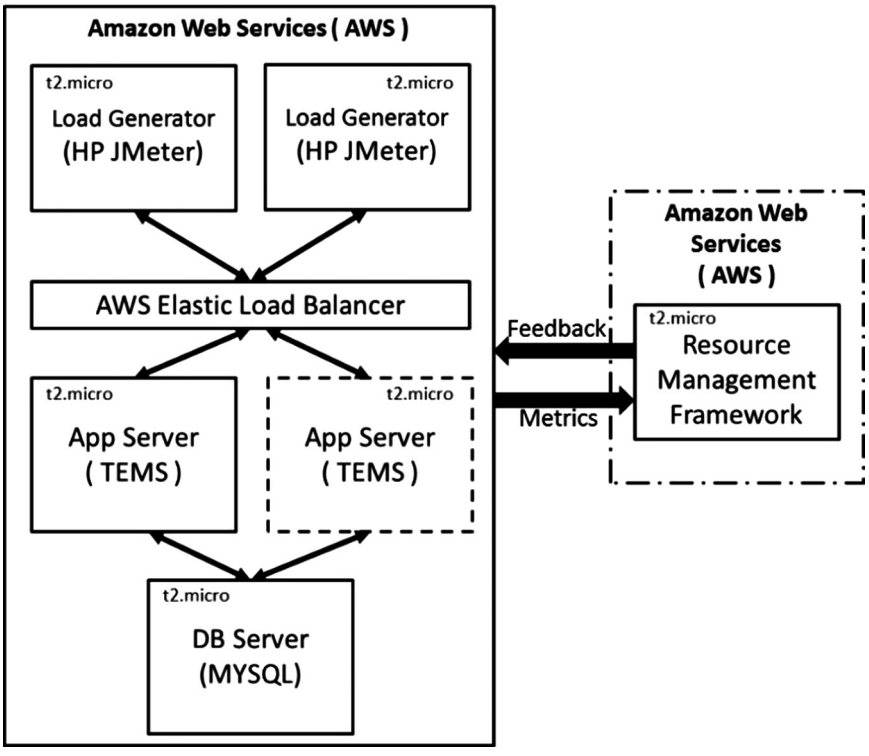

Figure 1 Experimental setup of the TEMS application and the resource management framework.

ally exclusive smaller spikes (about 16/17 concurrent user, for a 20 min window using a 20 k user requests) and larger spikes (32/33 concurrent users, for a 20 min window using 40 k user requests) were overlaid randomly on the steady workload to introduce smaller and larger variances respectively as shown in Fig. 2. After completion of every experimental run the simulation environment was reset. Thirty experimental runs were performed during the current study.

4.2. Rules for provisioning/deprovisioning instances

Based on the reactive resource management policies observed in enterprises, Amazon Auto Scale (AAS) group policy was setup to provision/deprovision the instances (Varia, 2010). Accordingly, minimum and maximum number of t2 micro instances was set to 1 and 5 respectively. A cool-off period of about 300 s was set between instance provisioning and deprovisioning. Instances were provisioned when the average CPU usage of the group reached more than 75% for 5 consecutive readings or when the sum of 5 consecutive readings of HTTP_5XX errors reached more than 50. Server instances were deprovisioned if the average CPU usage of the group reached below 25% for 5 consecutive readings.

4.3. Data acquisition

Instances were provisioned based on the average CPU usage. During each run, CloudWatch monitoring tool was used to acquire all the available evaluation metrics at the rate of one per minute along with the instance state. Latency, Surge Queue Length, Spill over Count, Request Count, HTTP Code_ELB_5XX, HTTP Code_Backend_2XX, HTTP Code_Backend_3XX, HTTP Code_Backend_4XX, HTTP Code_Backend_5XX, CPU Utilization, Memory Utilization, Disk Read Bytes, Disk Write Bytes, Network-In Bytes and Network-Out Bytes were the acquired metrics.

Parameters such as Disk Read/Write Operations and Network Packets In/Out Operations were captured but not considered for the current study as they were similar to Disk Read/Write Bytes and Network In/Out Bytes respectively. Request Count and HTTPCode_Backend_2XX represent the same attribute and hence the latter was only considered. Latency and Request Arrival are inversely proportional to each other and hence the latter was used. Request Arrival is a derived metric and has been computed using the formula below.

$$\begin{aligned}\text{Request Arrival} = (&\text{SurgeQueueLength} + \text{SpilloverCount}\\ &+ \text{HTTPCode\_ELB\_5XX}\\ &+ \text{HTTPCode\_Backend\_2XX}\\ &+ \text{HTTPCode\_Backend\_3XX}\\ &+ \text{HTTPCode\_Backend\_4XX}\\ &+ \text{HTTPCode\_Backend\_5XX})\end{aligned}$$

5. Predictive Resource Management Framework (PRMF)

PRMF in its current form consists of a statistical feature selection technique, a library of time-series/stochastic algorithm and a validation module. For a given workload, the key evaluation metric and the appropriate algorithm would be evaluated through an offline analysis. The best-fit algorithm for the particular workload would be applied on the key evaluation metric. Subsequently, PRMF also applies feedback by constantly monitoring the current workloads and updates the predictive algorithm if required.

5.1. Statistical feature selection technique

Most commonly used feature selection techniques are Mutual Information (MI), Information Gain (IG) and Chi-Square Test (Hua et al., 2010; Forman, 2004). MI of two random variables is defined as a measure of the mutual dependence between them and quantifies the "amount of information" obtained about one random variable, through the other random variable. Information gain is used to define a preferred sequence of attributes to investigate and most rapidly narrow down the state of random variable. Chi-square test is a statistical method for assessing the goodness-of-fit between a set of observed values and those expected theoretically. The current study applied information gain on Request Arrival, CPU Utilization, Memory Utilization, Disk Read Bytes, Disk Write Bytes, Network-In Bytes and Network-Out Bytes to determine which one of the metric has maximal impact on the instance state (provisioning/deprovisioning of instance).

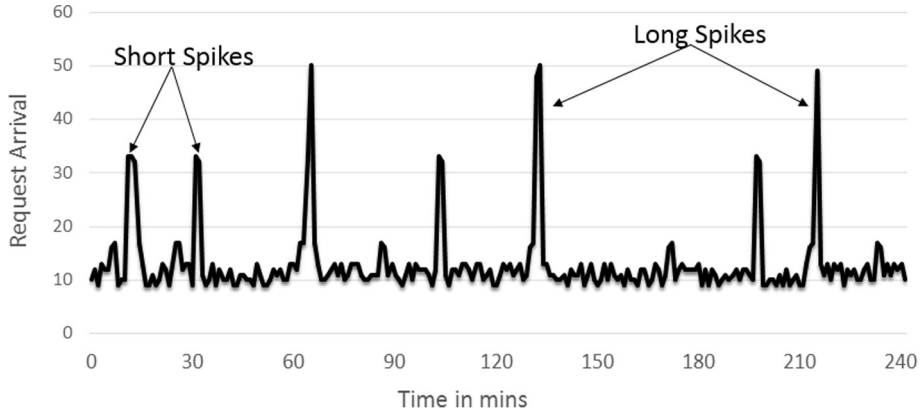

Figure 2 Sample workload pattern for an experimental run.

5.2. IG definition

Let **T** denote a set of training examples, each of the form (X, Y) = (X_1, X_2, X_3, …, X_k, Y) where $X_a \in$ Vals (a) is the value of the ath attribute of example X and Y is the corresponding class label. The information gain for an attribute "a" is defined in terms of entropy H () as follows:

$$IG(T, a) = H(T) - \sum_{v \in vals(a)} \frac{|\{X \in T | Xa = V\}|}{|T|} . H(\{X \in T | Xa = V\})$$

5.3. Instance-state mapping

Subsequently, the key evaluation metric and the corresponding instance-state along with their respective time stamps for the thirty experimental runs were stored. Median values for the key evaluation metric and instance-state were computed for every time-instance and decimal numbers if present were rounded to their nearest integer values. Polynomial regression technique was used to model the relationship that existed between the key evaluation metric and instance-state (Dixon and Massey, 1957). Regression equation thus obtained was leveraged in the real-time implementation to initiate additional VM request or to deprovision excess ones.

5.4. Algorithms in PRMF library

ARIMA and exponential smoothening models are the most widely used techniques for solving time-series predictions and both of them provide complimentary approaches to the same (Dixon and Massey, 1957; Huber and Ronchetti, 2011; Hurvich and Tsai, 1988). While exponential smoothing models are based on trend and seasonality in the data, ARIMA models describe the autocorrelations present in the data. It must be noted that the enterprise workload has change points of probability. Under such circumstances, a time series model might not be able to accommodate larger variance. Therefore Hidden Markov Model (HMM) was also included in the PRMF library.

Hidden Markov Models (HMM) are a class of discrete-time stochastic process consisting of recorded observations y_t and hidden states $x_t\{1, 2 \ldots J\}$ generated by a Markov chain. For observations $y = \{y_1, y_2 \ldots y_t\}$ and hidden states $x = \{x_1, x_2 \ldots x_t\}$, the model's joint distribution could be defined as

$$p(x, y) = \pi_0(x_1) p(y_1|x_1) \prod_{t=2}^{t} p(x_t|x_{t-1}, A) p(y_{1t}|x_t, \varnothing)$$

where $A = [A_{ij}]_{i,j=1}^{k}$ is the transition matrix with $A_{ij} = \Pr(x_t = j | x_{t-1} = i)$, $\theta = \{\theta_k\}_{k=1}^{k}$ as the emission parameters and π_0 as the initial probability distribution (Jackson, 2011). All experimental output patterns (key evaluation metric) were stored in a matrix and 95th percentile was computed for the same, which was used as input to determine the best predictive model as shown in Fig. 3.

Key evaluation metric was analyzed offline using ARIMA ($1 \leqslant p \leqslant 4$, $0 < d < 2$, $1 \leqslant q \leqslant 4$), Exponential smoothening techniques (Single, Double & Triple) and HMM (with three states namely (a) low (b) steady and (c) high).

5.5. Evaluation of best-fit models

Akaike Information Criterion (AIC) is a measure of the relative quality of statistical models for a given set of data. AIC of the model could be computed using the following equation;

$$AIC = 2k - 2\ln(L)$$

L is the maximum value of the likelihood function for the model and k is the number of estimated parameters in the model. Model that exhibits the smallest AIC value is the best-fit among the competing models. In the current study, best-fit among ARIMA models was determined using AIC metric (Akaike, 1987).

Best among the different groups of predictive models (best ARIMA, exponential techniques and HMM present in the PRMF library) could be determined by using either quantitative measures like (a) error metrics in the estimation/validation period, (d) residual diagnostics and (c) goodness-of-fit test or qualitative evaluation (Maddala and Lahir, 1992). It must be noted that among all the error metrics, Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE)/Mean Absolute Percentage Error (MAPE) are used to compare models for which errors are measured in the same units. However RMSE is more sensitive to larger variance (error) and hence MAE or MAPE is a more relevant criterion. Owing to the aforementioned reasons, MAE was used to identify the best-fit model among ARIMA/exponential smoothening techniques/HMM (i.e., the one which would fit majority of experimental output patterns).

5.6. Real-Time Implementation using validation module

During the real-time implementation, validation module (a java program) would continuously compare the actual metrics with the model output as shown in Fig. 4.

Based on the predicted value, the feedback method of the validation module will invoke AWS command-line APIs to provision/deprovision instances. PRMF will re-evaluate the best-fit algorithm using real-time metrics if the same breaches the confidence interval of the model output ($\pm 95\%$) for five continuous sample reading. Aforementioned approach was executed for 30 real-time runs to verify the efficiency of the PRMF. Time lag between resource demand and supply observed during reactive and proposed predictive approach were compared quantitatively.

6. Results and observations

Results of applying the QQ plot on historical record are shown in Fig. 5. Visual inspection of the same indicates that the historical data follow normal distribution.

Shapiro–Wilk statistic was found to be $W = 0.9034$ (± 0.052) and $p = 0.06584$ (± 0.0367). This also substantiates that the historical data follow the normal distribution. Request Arrival was found to be the metric with maximum information gain in all the thirty experimental runs and hence was used as the evaluation metric as shown in Table 1.

Polynomial Regression equation as shown in Fig. 6 was used for designing the map. By substituting different values of the predicted Request Arrival the current study could determine the instance state.

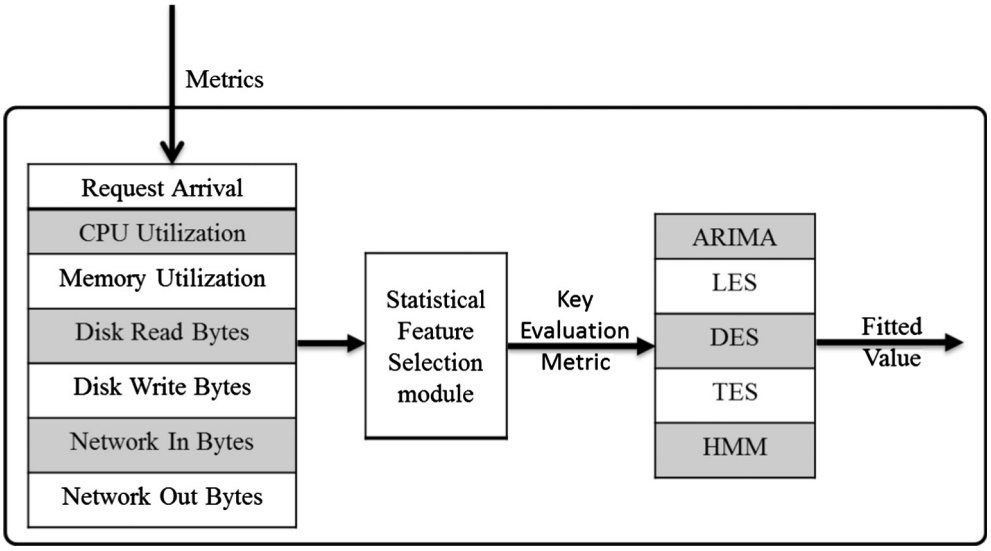

Figure 3 Initial (offline) experimental view of Predictive Resource Management Framework.

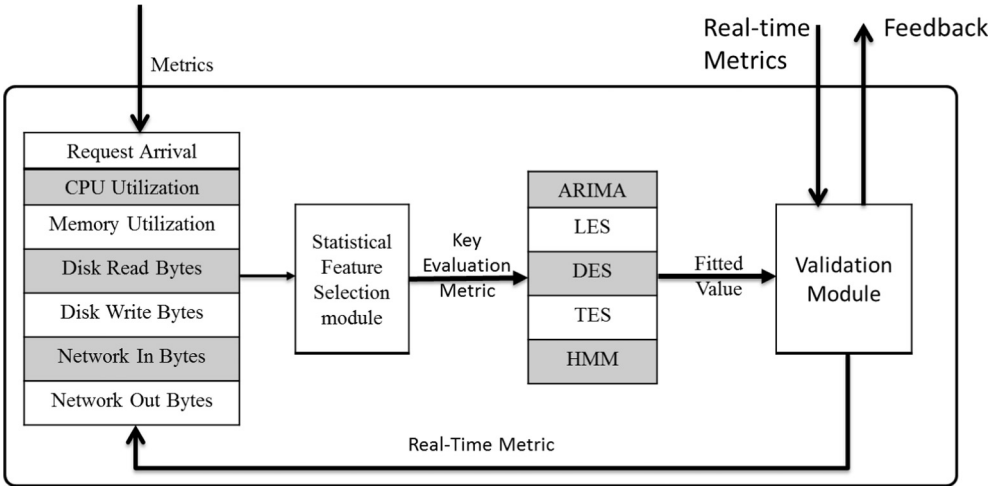

Figure 4 Real-time implementation of the Predictive Resource Management Framework.

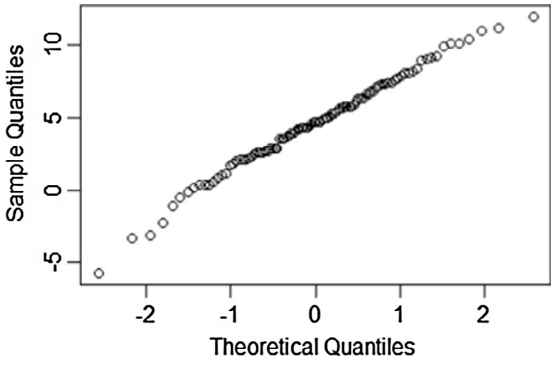

Figure 5 Q–Q Plot for the historical records.

ARIMA (2, 1, 3) model was found to be the best-fit among selected ARIMA models, whose corresponding AIC values are shown in Table 2.

MAE values of the five different models (ARIMA (2, 1, 3), LES, DES, TES and HMM) are shown in Table 3. ARIMA was found to be the best-fit and hence used as the algorithm of choice during real time experiments.

PRMF was able to predict the arrival rate within the set confidence level ($\pm 95\%$) for all the real-time runs. The best-fit ARIMA model and the real-time user request for a sample run is shown in Fig. 7a.

Instance count observed during the proposed approach was found to be sensitive to swings in the workload patterns when compared to the same during the reactive approach as shown in Fig. 7b. This leads to appropriate amount of instances provisioned/deprovisioned at a given time.

It was also observed that the reactive resource-provisioning approach led to a larger number of user request rejections (error count) as compared to that of the proposed approach as shown in Fig. 8. A reduction in request rejection leads to reduced wait time, more request processed and efficient utilization of the available resources.

Table 1 IG weights of the metrics considered across the different experimental runs.

Run #	Request Arrival	CPU Utilization	Memory Utilization	Disk Read Bytes	Disk Write Bytes	Network In Bytes	Network Out Bytes
1	0.02316	0.01192	0.01872	0.01346	0.01318	0.01160	0.01372
2	0.02806	0.01809	0.01904	0.01558	0.01364	0.01902	0.01361
3	0.02418	0.01507	0.01045	0.00000	0.01296	0.01151	0.00000
4	0.02256	0.01370	0.01523	0.01883	0.01340	0.01436	0.01759
5	0.02764	0.01839	0.01077	0.01696	0.01037	0.01265	0.01045
6	0.02666	0.01510	0.01245	0.01050	0.01181	0.01344	0.01717
7	0.02968	0.01948	0.01770	0.01922	0.01255	0.00000	0.01036
8	0.02894	0.01809	0.01374	0.01016	0.01434	0.01566	0.01427
9	0.02104	0.01830	0.01078	0.00000	0.00000	0.01081	0.01302
10	0.02130	0.01486	0.01467	0.01079	0.01289	0.01188	0.01679
11	0.02059	0.01568	0.01050	0.01404	0.01380	0.01659	0.01106
12	0.02086	0.01479	0.01623	0.01673	0.01854	0.01936	0.01721
13	0.01956	0.01472	0.01714	0.01266	0.00000	0.01887	0.00000
14	0.02388	0.01468	0.01601	0.01278	0.01027	0.00000	0.01885
15	0.02404	0.01606	0.01864	0.01698	0.01529	0.01145	0.01799
16	0.02683	0.01857	0.01027	0.01041	0.01383	0.01523	0.01275
17	0.02630	0.01436	0.01139	0.00000	0.01768	0.01610	0.00000
18	0.01907	0.01097	0.01902	0.01321	0.01026	0.00000	0.01327
19	0.02762	0.01213	0.01638	0.01635	0.01148	0.01606	0.01210
20	0.02103	0.01112	0.01265	0.00000	0.01434	0.00000	0.01179
21	0.02965	0.01253	0.01409	0.01589	0.00000	0.01352	0.01277
22	0.02490	0.01797	0.01473	0.01113	0.01563	0.01219	0.01098
23	0.02462	0.01061	0.01307	0.00000	0.01172	0.01543	0.01532
24	0.02007	0.01634	0.01433	0.01223	0.01376	0.00000	0.01860
25	0.02489	0.01812	0.01739	0.00000	0.01574	0.01564	0.01931
26	0.01922	0.01703	0.01789	0.01936	0.00000	0.01120	0.01515
27	0.02584	0.01802	0.01234	0.01299	0.01110	0.01706	0.01455
28	0.02899	0.01905	0.01416	0.01567	0.00000	0.01010	0.00000
29	0.01931	0.01909	0.01522	0.00000	0.00000	0.01574	0.01599
30	0.02051	0.01361	0.01096	0.01050	0.01662	0.01495	0.01017

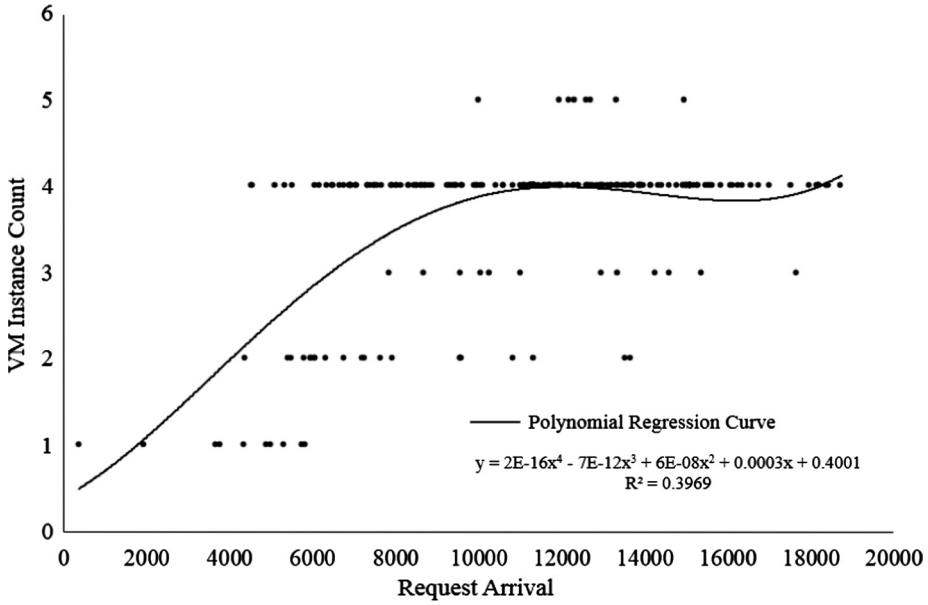

Figure 6 Determination of Instance state mapping by applying Polynomial Regression between median Request Arrival values and instance state obtained during experimental runs.

Table 2 Different ARIMA Models and their AIC values.

ARIMA model order	AIC
111	4514.716
211	4516.468
312	4516.962
112	4516.570
311	4517.191
113	4517.408
212	4517.817
412	4518.336
413	4514.287
411	4519.158
213	4510.903
114	4519.372
414	4521.947
214	4518.103
313	4512.287
314	4519.995

Table 3 Different models considered and their MAE values.

Models considered	MAE
ARIMA (2, 1, 3)	1201.934
LES	1639.805
DES	1682.127
TES	1242.409
HMM	1568.420

7. Discussion and future work

Timesheet is neither a compute-intensive nor a memory-intensive application and hence AWS t2.micro instances were chosen for the study. It must be noted that server instances of AWS could be spread across geography. AWS best practices recommend to have the application setup spread across multiple geographies for higher fault tolerance. In a marked difference, experimental setup for the current study was chosen from the same geographical location to avoid significant time lag between instance request and provisioning/deprovisioning. Owing to the aforementioned reasons, the efficiency of the proposed approach would be significantly more if the server instances were spread across geography. AWS has also published best practices to reduce the instance startup time. These suggested practices will have similar impact irrespective of the resource provisioning approach (reactive or proposed). Therefore, these were not considered in the current experimental study. It must be noted that average of any metric is susceptible to the outliers. Hence, to accommodate larger variances spread across the experimental runs, 95th percentile was used as the key evaluation metric on the study.

Few studies on resource management have used a similar methodology to that of the current study. However, objective of those studies were to predict the resource demand based on the reactive metrics (average CPU, average Memory), which exhibit inherent time lag (Iqbal et al., 2009, 2010; Xiong et al., 2011; Bennani and Menasce, 2005; Bhulai et al., 2007; Shi et al., 2011; Bankole and Ajila, 2013). It must be noted that the arrival rates have significant impact on the server response time and hence could be a potentially better metric than average load (Wang et al., 2014; Tammaro et al., 2011; Han et al., 2013). Results of the current study also concur with the same.

Container technology primarily concentrates on the application portability and has limited influence on the VM's startup time. VM could be provisioned at a very short time period using multiple container technologies such as Dockers (Merkel, 2014). Applications on the VM would need an approach to reduce the significant time lag that exists between instantiated and active VM. Predictive systems, just like that described in the current study, aim to reduce the time lag mentioned above and could be visualized as an accelerator complimenting the existing container technologies.

An enterprise scenario would present various workload patterns. It must be noted that the evaluation metric, which determines the Cloud resource management is dependent on the application workload. Framework proposed in the current study has used a statistical feature selection technique to identify key evaluation metric of any given workload pattern. For different workload types the key evaluation metric identified could be different and such a technique could reduce the necessity for the dynamic identification of a workload pattern. Although there are machine-learning techniques available to identify the workload patterns, scope of the current study was limited to propose a framework which, could identify a

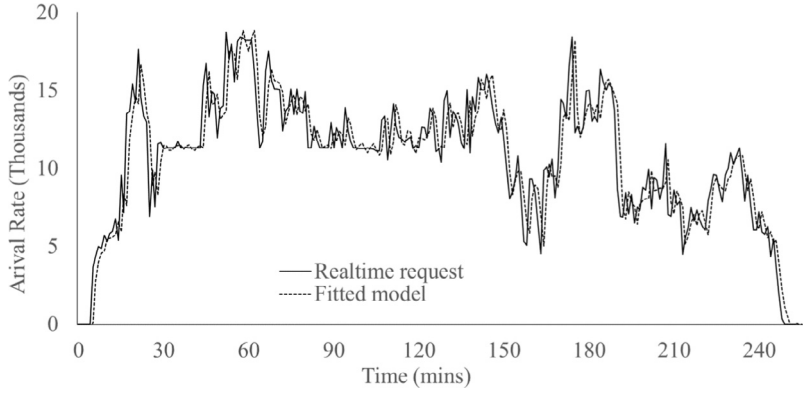

Figure 7a Sample real time request with the fitted workload pattern.

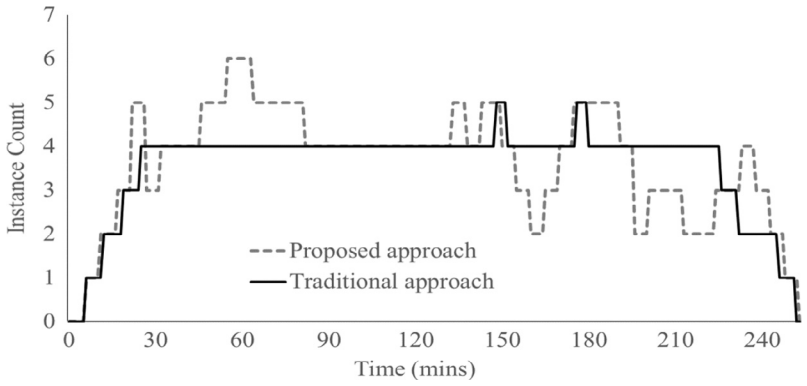

Figure 7b Observed sample instance counts of reactive and proposed approaches.

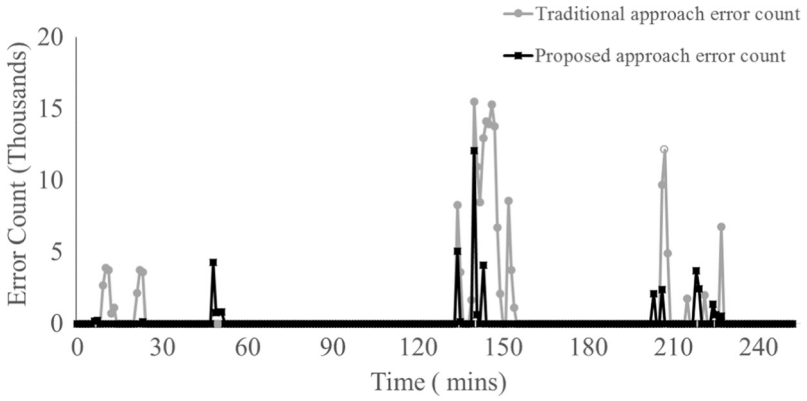

Figure 8 Error count computed for the reactive and proposed approaches of a sample experimental run.

significant metric and an algorithm to provision/deprovision resources in an automated way.

Majority of the studies have assumed steady state load and hence handling of controlled non stationary events was the objective of the proposed approach. Experimental results indicate that the proposed methodology could be considered as an efficient alternative to the reactive method of Cloud resource management while deploying applications exhibiting non-stationary workload patterns. In the current study, simulations of the workload were generated based on a comprehensive study of the historic patterns. This represents a controlled environment leading to the presence of non-linearity/uncertainty in the simulated workload. In such a case, a time-series model could work better than that of stochastic prediction technique (Chatfield, 2016; Box et al., 2015). The results of our study concur with the same. However, when the proposed PRMF is expanded to other workloads with very large uncontrolled variance, the time-series models present in the same, might not work as expected. Under such circumstances stochastic models might work better and hence HMM model has been included in the PRMF library. Many such stochastic models would be part of the PRMF library in future.

The objective of the current study was not to delve upon charging model, but to deprovision idle resources on-demand. Irrespective of the resource utilization, existing chargeback model ensures that the consumer has to pay for the resources per hour (Woitaszek and Tufo, 2010). Therefore deprovisioning using current approach might not prove more beneficial to the end user when compared to the reactive approach. Therefore, methodologies to suggest an efficient charging model would be part of the evolution of this study.

8. Conclusion

Resource management methods provided by the leading Cloud service-providers are based on predetermined rules and are reactive. Reactive approach has an inherent time-lag between resource demand and provisioning, which leads to under or over-provisioning. The study has attempted to overcome this limitation by adopting a predictive method for managing resources on Cloud. A read-intensive on-off time sheet application was deployed on AWS t2 micro instance. Workload patterns for the time sheet application were simulated based on the historical data. Real-time resource management was studied using both the reactive and proposed approach during all the thirty experimental runs. Arrival rate was found to be the key evaluation metric for the workload considered. ARIMA models, LES, DES, TES and HMM were all applied on the key evaluation metric. ARIMA (2, 1, 3) was found to be better-fit among all the chosen models and hence was used in real time implementation. It was observed that the proposed approach had reduced user- request rejections (error count), shorter user-wait time, higher request processed and efficient utilization of available Cloud resources compared to reactive approach.

References

Akaike, H., 1987. Factor analysis and AIC. Psychometrika 52 (3), 317–332.

Ali-Eldin, A., Tordsson, J., Elmroth, E., 2012. An adaptive hybrid elasticity controller for Cloud infrastructures, 2012 IEEE Network Operations and Management Symposium. IEEE, pp. 204–212.

Anuradha, V.P., Sumathi, D., 2014. A survey on resource allocation strategies in Cloud computing. In: International Conference on Information Communication and Embedded Systems (ICICES2014), 3(6), pp. 1–7.

Arlitt, M.F., Williamson, C.L., 1997. Internet Web servers: workload characterization and performance implications. IEEE/ACM Trans. Networking 5 (5), 631–645.

Armbrust, M. et al, 2010. A view of Cloud computing. Commun. ACM 53 (4), 50.

Bankole, A.A., Ajila, S.A., 2013. Cloud client prediction models for Cloud resource provisioning in a multitier web application environment, 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering. IEEE, pp. 156–161 [Internet].

Bennani, M.N., Menasce, D.A., 2005. Resource allocation for autonomic data centers using analytic performance models, Second International Conference on Autonomic Computing (ICAC'05). IEEE, pp. 229–240.

Bharti, K., Kaur, K., 2014. A survey of resource allocation techniques in Cloud computing. Int. J. Adv. Comput. Eng. Comm. Tec. (2), 31–35

Bhulai, S. et al, 2007. Modeling and predicting end-to-end response times in multi-tier internet applications, ITC20'07 Proceedings of the 20th International Teletraffic Conference on Managing Traffic Performance in Converged Networks, pp. 519–532.

Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M., 2015. Time Series Analysis: Forecasting and Control. John Wiley & Sons.

Bunch, C. et al, 2012. A pluggable autoscaling service for Open Cloud PaaS systems, 2012 IEEE Fifth International Conference on Utility and Cloud Computing. IEEE, pp. 191–194.

Calheiros, R.N., Ranjan, R., Buyya, R., 2011. Virtual machine provisioning based on analytical performance and QoS in Cloud computing environments, 2011 International Conference on Parallel Processing. IEEE, pp. 295–304.

Calheiros, R. et al, 2014. Workload prediction Using ARIMA model and its impact on Cloud applications' QoS. IEEE Trans. Cloud Comput. XX (c). 1–1.

Chatfield, C., 2016. The Analysis of Time Series: An Introduction. CRC Press.

Deng, D., Lu, Z., Fang, W., Wu, J., 2013. CloudStreamMedia: a Cloud assistant global video on demand leasing scheme, 2013 IEEE International Conference on Services Computing. IEEE, pp. 486–493 [Internet].

Dixon, W.J., Massey, F.J., 1957. Introduction to Statistical Analysis. .. Population, 12(4)., p. 729.

Elprince, N., 2013. Autonomous resource provision in virtual data centers, Integrated Network Management (IM 2013), pp. 295–304.

Forman, G., 2004. A pitfall and solution in multi-class feature selection for text classification, Proceedings of the Twenty-first International Conference on Machine Learning. ACM Press, New York, New York, USA, p. 38.

Gong, Zhenhuan, Gu, Xiaohui, Wilkes, J., Wilkes, J., 2010. PRESS: PRedictive Elastic ReSource Scaling for Cloud systems, 2010 International Conference on Network and Service Management. IEEE, pp. 9–16.

Han, Y., Chan, J., Leckie, C., 2013. Analysing virtual machine usage in Cloud computing, 2013 IEEE Ninth World Congress on Services. IEEE, pp. 370–377 [Internet].

Herbst, Roman N., Kounev, S., Reussner, R., Wilkes, J., 2013. Elasticity in Cloud computing: what it is, and what it is not, 10th International Conference on Autonomic Computing, pp. 361–362.

Hu, Jinhua et al, 2010. A scheduling strategy on load balancing of virtual machine resources in Cloud computing environment, 2010 3rd International Symposium on Parallel Architectures, Algorithms and Programming. IEEE, pp. 89–96.

Hua, G. et al, 2010. Hierarchical feature selection for ranking, Proceedings of the 19th International Conference on World Wide Web – WWW'10. ACM Press, New York, New York, USA, p. 1113.

Huber, P.J., Ronchetti, E.M., 2011. Robust Statistics. John Wiley & Sons.

Hurvich, C.M., Tsai, C.-L., 1988. Regression and time series model selection in small samples.pdf. Biometrika 76 (2), 297–307.

Iqbal, W., Dailey, M., Carrera, D., 2009. SLA-driven adaptive resource management for web applications on a heterogeneous compute Cloud, CloudCom '09 Proceedings of the 1st International Conference on Cloud Computing, pp. 243–253 [Internet].

Iqbal, W., Dailey, M.N., Carrera, D., 2010. SLA-driven dynamic resource management for multi-tier web applications in a Cloud, 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. IEEE, pp. 832–837 [Internet].

Jackson, P., 2011. HMM Tutorial. Centre for Vision Speech & Signal Processing, University of Surrey, Guildford.

Khajeh-Hosseini, A., Sommerville, I., Sriram, I., 2010. Research challenges for enterprise Cloud computing. Information Security. http(1960), p.//arxivorg/abs/10013257.

Kim, S., Koh, J.-I., Kim, Y., Kim, C., 2011. A science Cloud resource provisioning model using statistical analysis of job history, 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing. IEEE, pp. 792–793 [Internet].

Kochut, A. et al, 2011. Evolution of the IBM Cloud: enabling an enterprise Cloud services ecosystem. IBM J. Res. Dev. 55 (6), 7:1–7:13.

Kouki, Y. et al, 2014. A language support for Cloud elasticity management, 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, pp. 206–215.

Kouki, Y., Hasan, M.S., Ledoux, T., 2015. Delta scaling: how resources scalability/termination can be taken place economically?, 2015 IEEE World Congress on Services IEEE, pp. 55–62.

Maddala, G.S., Lahir, K., 1992. Introduction to Econometrics. Macmillan, New York.

Mao, M., Li, J., Humphrey, M., 2010. Cloud auto-scaling with deadline and budget constraints, 2010 11th IEEE/ACM International Conference on Grid Computing. IEEE, pp. 41–48.

Mary, M.A., 2013. Survey on resource management technique in Cloud computing. Int. J. Eng. Res. Tec. 2 (12), 1723–1731.

Mell, P.M., Grance, T., 2011. The NIST Definition of Cloud Computing. In Special Publication (NIST SP) – 800-145.

Merkel, Dirk, 2014. Docker: lightweight linux containers for consistent development and deployment. Linux J. 2014 (239), 2.

Mulia, W. et al, 2013. Cloud workload characterization. IETE Techn. Rev. 30 (5), 382.

Shi, Y., Jiang, X., Ye, K., 2011. An energy-efficient scheme for Cloud resource provisioning based on cloudsim, 2011 IEEE International Conference on Cluster Computing. IEEE, pp. 595–599.

Tammaro, D., Doumith, E.A., Al Zahr, S., Smets, J.-P., Gagnaire, M., 2011. Dynamic resource allocation in Cloud environment under time-variant job requests, 2011 IEEE Third International Conference on Cloud Computing Technology and Science. IEEE, pp. 592–598 [Internet].

Varia, J., 2010. Architecting for the Cloud: Best Practices.

Wang, C.-F., Hung, W.-Y., Yang, C.-S., 2014. A prediction based energy conserving resources allocation scheme for Cloud computing, 2014 IEEE International Conference on Granular Computing (GrC). IEEE, pp. 320–324 [Internet].

Woitaszek, M., Tufo, H.M., 2010. Developing a Cloud computing charging model for high-performance computing resources, 2010 10th IEEE International Conference on Computer and Information Technology. IEEE, pp. 210–217.

Xiong, P., Chi, Y., Zhu, S., Moon, H.J., Pu, C., Hacigumus, H., 2011. Intelligent management of virtualized resources for database systems in Cloud environment, 2011 IEEE 27th International Conference on Data Engineering. IEEE, pp. 87–98 [Internet].