IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 5, Issue 1, Feb - Mar, 2017
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

Load Balancing And Performance Evaluation In Saas Layer Using Autoregressive Model

S.Sankara Narayanan[1], Dr. M.Ramakrishnan[2]

[1]Research Scholar, Department of Computer Science and Engineering,
Anna University, Chennai
[2]Professor and Head, Department of Computer application, School of information Technology,
Madurai Kamaraj University, Tamil Nadu

Abstract— **In recent year, development of software and deployment in to the real world based on the new model of software as a service (SaaS) and service oriented architecture (SOA) convey lot of support to cloud user. Conversely, the developer and service provider have to deal with new challenging problem before utilizing the befits of the software. The software companies moved from desktop application to cloud based application set upped in the public cloud. The offering of service by the cloud provider is similar and it growing very fast manner. In order to withstand in the competitive market cloud based companies must provide good Quality of service (QoS) for cloud user. However, giving QoS service with less cost effective amount of resources is challenging tasks because workload experience is varying time to time. This problem can be resolved with the help of proactive dynamic provisioning of resources which estimate required resources in advance for software application. In this paper, we present the insight of a cloud workload prediction module for SaaS provider based on the Autoregressive model. Our model also evaluates its accuracy using future work load prediction using real sketch of request of web servers. The experimental results show that our model achieves good accuracy which shows efficiency in resource utilization with minimum QoS.**

Keywords— Software as a Service, workload predicition, Autoregressive model, webservice performance, Quality of service(Qos).

I. INTRODUCTION

Now a day, software industry uses cloud computing technologies as computing paradigm for efficient and flexible usage of resources. The services are provided over the internet just-in-time and are paid per usage. Cloud based companies offered [1] cloud based solution to end users by adopting new technologies in to software companies. The shift from desktop applications to public Cloud hosted Software as a Service (SaaS) business model has intensified the competition for Cloud providers. This is due to the presence of multiple providers in the current Cloud computing landscape that offer services under heterogeneous configurations.

Selecting particular Cloud service configuration (e.g., VM type, VM cores, VM speed, cost, and location) translates to a certain level of Quality of Service (QoS) in terms of response time, acceptance rate, reliability, etc. In order to survive in such a competitive market, Cloud providers must deliver acceptable QoS to end-users of the hosted SaaS applications. However, one issue that arises from the transition to a SaaS model is the fact that the pattern of access to the application varies according to the time of the day, day of the week, and part of the year. It means that in some periods there are many users trying to use the service at the same time, whereas in others only a few users are concurrently accessing the servers.

This makes static allocation of resources to the SaaS application ineffective, as during a period of low demand there will be excess of resources available, incurring unnecessary cost for the application provider, whereas during high utilization periods the available resources may be insufficient, leading to poor QoS and loss of costumers and revenue. Clouds can circumvent the above problem by enabling dynamic provisioning of resources to applications based on workload behavior patterns such as request arrival rate and service time distributions.

This means that extra resources can be allocated for peak periods and can be released during the low demand periods, increasing utilization of deployed resources and minimizing the investment in Cloud resources without loss of QoS to end users [2].

The dispute of dynamic provisioning is the determination of the correct amount of resources to be deployed in a given time in order to meet QoS expectations in the presence of variable workloads examined by Cloud applications. This challenge has been tackled mainly via reactive approaches [3, 4, 5] which increase or decrease resources when predefined thresholds are reached via proactive approaches [6,7,8] which react to future load variations before their occurrence.

The latter is typically achieved with techniques that can monitor, predict, adapt according to these prediction models, and capture the relationship between application QoS targets, current Cloud resource allocation, and changes in workload patterns, to adjust resource allocation configuration on-the-fly.

In existing system, we introduced architecture for proactive dynamic provisioning via workload prediction—which determines how many requests per second are expected in the near future combined with analytical models to determine the

optimal number of resources in the presence of the predicted load.

Even though the proposed architecture renowned the need for workload prediction, it did not propose a concrete method for workload prediction. In this paper we present the design and evaluation of an awareness of its workload prediction model using the Autoregressive (AR) model [9]. AR is a method for non-stationary time series prediction that is composed of an autoregressive and a moving average model, and was successfully utilized for time series prediction in different domains such as finance. The key contributions of this paper are:

Our system uses Auto regressive model to design, and develop a workload prediction module using the ARIMA model. Our work applies feedback from latest observed loads to update the model on the run. The predicted load is used to dynamically provision VMs in an elastic Cloud environment for serving the predicted requests taking into consideration QoS parameters such as response time and rejection rate; We conduct an evaluation of the impact of the achieved accuracy in terms of efficiency in resource utilization and QoS of user requests.

The experiment results show that our component achieves accuracy of up to 80%, which leads to effectiveness in resource utilization with minimal impact in QoS for users.

The rest of paper organized as follows: section 2 present related work on load balancing in SaaS, section 3 present application and system model section 4 discuss proposed architecture and its components, section 5 present performance evaluation the accuracy of our proposed prediction architecture. Section 6 present conclusion.

II. RELATED WORK

Workload prediction techniques in cloud computing can be classified into reactive and proactive methods. Among reactive methods, Zhu and Agrawal [3] propose a method based on control theory to vertically scale resource configurations such as VM types, VM cores, VM speed, and VM memory. Vertical scaling is the process of increasing the resources available to each Virtual machine. Horizontal scaling is process of increasing the number of VM. Their approach also addresses the financial plan constraints related to the workload implementation. They apply the ARMAX model to predict CPU cycle and memory configurations required for hosting an application component.

In difference to this approach, we pertain the ARIMA model to predict the future application workload behavior, which is fed into the queueing model for calculating the required VM configuration. Bonvin et al. [4] propose a reactive method that scales servers based on the expected performance and profit generated by changes in the provisioning. This technique is able to execute both horizontal and vertical scaling.

Similar to Bonvin et al., Yang et al. [5] propose a reactive method for changing the resource configuration of cluster resources driven by the load incurred by the hosted application. It is based on user-defined threshold conditions and scaling rules that are automatically enacted over a virtualized cluster. Zhang et al. [10] propose a reactive workload factoring architecture for hybrid Clouds that decomposes incoming workload in base workload and trespassing workload. The first one is derived from AR based prediction and handled by the local infrastructure, whereas the second is handled by a public Cloud.

The limitation of reactive platforms is that they react to changes in workload only after the change in utilization and throughput is observed in the system. Therefore, if the change is quicker than the reconfiguration time, end users will observe poor QoS until the extra resources are available. Let Considering that changes in the workload typically follow patterns that are time-dependent, prediction techniques can avoid the above problem by triggering the reconfiguration before the expected increase of demand, so when the situation arises, the system is already prepared to handle it.

Caron et al. [6] propose a method based on pattern matching for prediction of grid-like workloads in public Clouds. Gong et al. [11] propose a method for predicting resource demand of VMs based on predicted application workload. Islam et al. [8] apply Artificial Neural Networks and linear regression for prediction of resources required for applications. Sladescu et al. [7] presents a system based on ANN to predict the workload to be experienced by an online auction in terms of intensity and location of the peaks.

Although techniques such as linear regression can generate predictions quicker than ARIMA, they also demand workloads that have simpler behavior than those that time series and ANN-based methods can accurately predict. Furthermore, studies [12, 13] show that web and data center workloads tend to present behavior that can be effectively captured by time series-based models. Thus, to increase the applicability of the proposed architecture, we adopt ARIMA-based prediction for our proposed architecture.

Other domain-specific proactive approaches that are related to Clouds include the approach by Nae et al. for Massively Multiplayer Online Games [14]. PachecoSanchez et al. [15] apply a Markovian model to predict server performance in Clouds. Roy et al. [12] apply the ARMA model for workload prediction in Clouds with the goal of minimizing cost, whereas the main objective of our approach is meeting QoS target of applications such as minimizing the request rejection rate, or maximizing resource utilization.

III. APPLICATION AND SYSTEM MODEL

The proposed system architecture module consist of public cloud provider that offer SaaS services backed by the Platform as a service (PaaS) shown in figure 1. The PaaS in turn interacts with an IaaS provider that can be a third party

provider. The target SaaS provider receives web requests, which are processed by the machines that are located at the IaaS layer. For scaling up the infrastructure, the target provider deploys a number of Virtual Machines (VM) that process end user requests. To simplify the management of the infrastructure and to take advantage of profiling information, a single VM configuration, consisting of CPU power, amount of memory, and amount of storage is utilized by the SaaS provider.

Our system assumes that the application has been profiled in the chosen VM configuration, so the provider has information about the Virtual machine expected performance. An application instance executes on each VM, and since current Cloud providers do not support dynamic changes in the VM's specifications without downtime, increasing and decreasing the total number of VMs running the application is the most suitable option for utilization of elastic computing infrastructure, as it brings additional benefits such as higher fault tolerance and higher resilience to performance degradation caused by VM failures (as the crash of one of the VMs will not affect the others, enabling the application to continue serving customers using the VMs that are running).

The target application is web applications. Client requests consist of http requests that are processed by a web server running on the VMs. QoS targets of relevance to the system are response time RT_S, defined as the maximum negotiated time in the SLA for serving a user's request and rejection rate RejeR(G_S), which is the proportion of incoming requests that cannot be served without violating RT_S[16].

IV. PROPOSED SYSTEM ARCHITECTURE

The important key characteristics of the public cloud are elasticity which enables the infrastructure to be scaled or down to meet the demand of applications. However, instantiation of new VMs is not an immediate operation. Depending on Cloud providers' infrastructure architecture and their hypervisor policies, launching a new VM involves a non-negligible start-up time.

Even though Standby VM instances may be helpful for tolerating sudden increases in number of requests, those standby VMs are more likely to be idle most of the times reducing the overall system utilization while increasing the operational cost. Furthermore, if the increase in the number of requests exceeds the load that standby VMs can handle, the problem of poor QoS occurs again. Thus, a different approach must be sought for the Cloud provisioning problem.

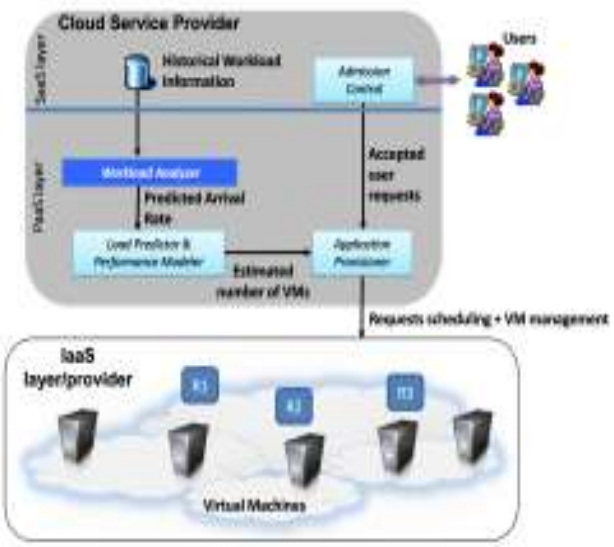

Figure 1. Adaptive cloud provisioning and it components

One approach that has been explored is based on workload prediction: accurate predictions of the number of end-users' future service requests enable SLA's QoS targets to be met with reduced utilization of Cloud resources. As requests pattern vary depending on the application type, this paper focuses on request patterns that exhibit seasonal behavior, such as requests to Web or online gaming servers [14,17]. To overcome the uncertainty in workload patterns in Cloud environments and minimize the estimation error in predicting future requests while maintaining optimal system utilization, in previous work [16] we proposed an adaptive provisioning mechanism in order to achieve the following QoS targets:

Automation: The whole process of provisioning should be transparent to users;

Adaptation: The provisioner should be aware of dynamic and uncertain changes in the workload and react to them accordingly;

Performance assurance: In order to meet QoS targets, resource allocation in the system must be dynamic. The key components of the proposed provisioning system, depicted in Architecture figure 1.

Application Provisioner: Receives accepted requests from the Admission Control module and forwards them to VMs that have enough capacity to process them. It also keeps track of the performance of the VMs. This information is passed to the Load Predictor and Performance Modeler. The Application Provisioner also receives from such module the expected number of VMs required by the application. If the expected number of VMs differs from the number of provisioned VMs, the number is adjusted accordingly by either provisioning new VMs or decommissioning unnecessary VMs.

Load Predictor and Performance Modeler: Decides the number of VMs to be allocated, based on the predicted demand by the Workload Analyzer module and on the

observed performance of running VMs by the Application Provisioner.

The performance is modeled via queueing networks, which, based on the predicted arrival rate of requests, return the minimum number of VMs that is able to meet the QoS metrics. Workload Analyzer: Generates an estimation of future demand for the application. This information is then passed to the Load Predictor and Performance Modeler module. To construct the proposed architecture effective, a strong knowledge about the application workload behavior is required by the system so the performance model can be accurate. Therefore, the most suitable deployment model for the architecture is Software as a Service, where a queueing model can be built for each application offered to end users as a service.

In the SaaS layer, the admission control module ensures that no application instance will get further requests if the capacity of the queue is exhausted. In this case, all the upcoming requests are rejected, because otherwise it is most likely that T_s would be violated. Accepted requests are forwarded to the Cloud provider's PaaS layer where the proposed system is implemented.

In our Existing work [9], the system architecture was presented in a high-level view, without presenting a concrete implementation of each of its components. In this paper, we present a realization of the Workload Analyzer component of the architecture. The prediction method uses the Auto-Regressive (AR) model. The prediction gives the Application Provisioner enough time to react against any precipitous increase in workload by starting new VMs without compromising T_s while maintaining the overall system utilization above a given threshold.

A. Work load analyzer

Our system uses Auto Regressive time series process to predict the workload cable by each virtual machine. AR has been chosen for the implementation of our module because the underlying workload fits well in the model: previous research observed that web workloads tend to present strong auto-correlation [18, 19] From the beginning of the execution, the historical workload data is fed into the Workload Analyzer, where it fits the AR model on them.

When the system is operational, it delivers an estimation of the workload with one time-interval in advance. The length of the time interval can be adjusted to better fit the specific application. The only requirement for efficient system utilization is that the time interval should be long enough to allow extra VMs to be deployed. Therefore, time windows as short as 10 minutes could be suitable depending on the selected Cloud provider [18].

The request time series contains the number of observed requests at each time interval. It is implemented as a cyclic buffer so that at the next prediction cycle, the actual number of requests (obtained from the original dataset) is added to the time series used in prediction while discarding the oldest value. After constructing the request time series, the process of fitting the ARIMA model is initiated based on the Box-Jenkins method [9].

According to this method, the time series must be transformed into a stationary time series, that is, for each (X_t $X_{t+\tau}$), being the time difference between two data points, the mean and variance of the process must be constant and independent of t. In addition, the auto-covariance between X_t and $X_{t+\tau}$ should be affected only by τ. This transformation is achieved by differencing the original time series. The number of times the original time series has to be differenced until it becomes stationary constitutes the d parameter of the AR (X,Y,Z) model.

The values of X and Z are determined by analyzing the autocorrelation and partial autocorrelation plots of the historical data, respectively. In the context of this work, historical data means the observed number of requests per second received by the system in some past time interval. The autocorrelation plot is used to determine how random a dataset is. In the case of random data, the autocorrelation values approach zero for all time-lagged values, otherwise, one or more autocorrelation values approach 1 or -1. In the autocorrelation plot, the horizontal axis represents the time lags. Values on the vertical axis are calculated using the autocorrelation coefficient R_h:

$$R_h = \frac{C_h}{C_0} \qquad (1)$$

Where C_h is the auto-covariance function defined as:

$$C_h = \frac{1}{N}\sum_{t=1}^{N-\tau}(X_T - \overline{x})(X_{t+1} - x) \qquad (2)$$

where N is the number of samples and X is the average of samples X_t; t = 1:::N . C_0 is the variance function:

$$C_h = \frac{1}{N}\sum_{t=1}^{N}(X_T - \overline{x}) \qquad (3)$$

The partial autocorrelation at τ is the autocorrelation between X_t and $X_{t\_}$ τ that is accounted only by lags above τ - 1. If a stationary time series has an auto regression component of order p, its partial autocorrelation plot falls below the significant level at _ = p + 1. The number of lags before the autocorrelation values drop below the significant level is the value of q for the moving average component of the ARIMA model. Using the above method to determine the terms X, Y, and q of the ARIMA model, the historical workload information is fit to the model to be used for prediction of future workload values.

B. Auto Regeressive sysem design

The class diagram of the ARIMA-based workload prediction system is shown on Figure 2. The ARIMA Workload Analyzer is the core component of the system and realizes the Workload Analyzer component of Figure 1. By

implementing the IFeedbackable interface, it is capable of taking feedbacks. In our system, current workload information, received from external components is modeled as a feedback signals and fed into the ARIMA Workload Analyzer to make it aware of the most recent workload changes.

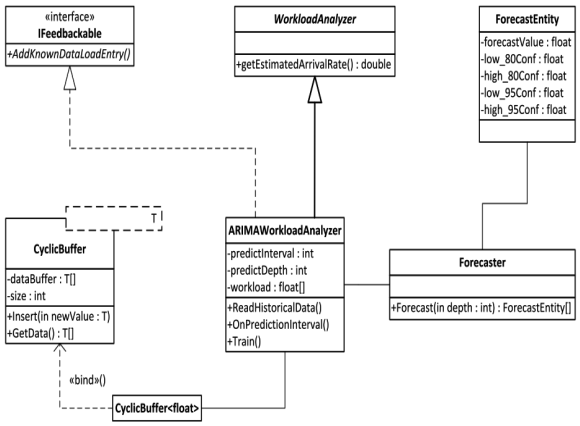

Figure 2. Class diagram for work load predication

ARIMA Workload Analyzer stores the given feedback signals into a cyclic buffer. The length of this buffer, which corresponds to the number of time intervals in past affecting the current prediction value, is configured at the start of the system based on characteristics of the application workload (number of received requests per second). ARIMA Workload accomplishes the workload prediction through the Forecaster class. This class has a connection to a statistical backend (the R forecast package [19], which for simplicity is not presented on the diagram).

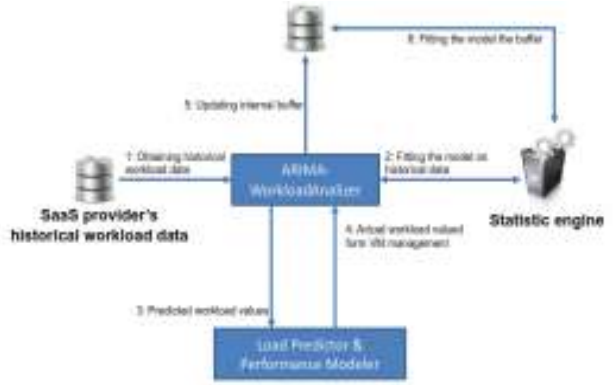

Figure 3 workload prediction steps and its components

It accepts a time series from the ARIMA Workload Analyzer and prepares it for submission to the statistic engine, where ARIMA model is fitted on them. For a given time series, the statistical back-end replies with a predicted value, along with its corresponding 80% and 95% confidence levels. The Forecaster class then parses and encapsulates this reply into an instance of Forecast Entity class and passes it back to the ARIMA Workload Analyzer. The accuracy of the ARIMA-based workload prediction is evaluated in the next section. The steps of the prediction procedure and its components are shown on Figure 3.

V. PERFORMANCE EVALUATION

The system was evaluated with real traces of requests to the web servers from the Wikimedia Foundation. These traces contain the number of http requests received for each of the project's resources (static pages, images, etc) aggregated in 1-hour intervals and are publicly available for download. It also contains the project name associated with each resource being requested and the language of each accessed resource.

We consider only requests to English Wikipedia resources in these experiments. An analysis of patterns of web requests to Wikipedia servers was presented by Urdaneta et al. [17]. In order to observe weekly patterns, we use four weeks of the traces, dated from midnight, 01 January 2015 to 5 pm, 04 February 2015.

The first three weeks are used for training purposes. The requests corresponding to such a period are transformed to a time series process (i.e. the values X; Y and Z of the ARIMA model are defined). At runtime, the model is constantly updated: whenever new requests arrive, they are incorporated to the time series and older data is removed from the time series in the same amount.

The fourth week is used for evaluation purposes. Based on the training dataset, the demand for each hour of the fourth week is predicted. The output of the prediction procedure is a number, accompanied by two confidence ranges, covering the 80% and 95% bands, for each hour of the fourth week. Figure 4 presents the predicted and actual values (i.e., the value observed in the traces) and corresponding confidence ranges, for the fourth week of the workload. The accuracy of the prediction is evaluated using various error metrics.

Metric	Predicted	Low	High
Root Mean square deviation RMSD	1156	1580	1985
Normalized Root Mean square deviation (NRMSD)	0.13	0.22	0.26
Mean absolute deviation (MAD)	886.98	1154.65	1468.36
Mean absolute percentage error (MAPE)	0.08	0.12	0.18

Table1. Prediction accuracy by various metrics

The results are presented in Table 1. The Predicted column contains the accuracy according to different metrics. The Low 80% and High 80% contain the limits for the 80% confidence interval for the prediction. The table also reports the same for the 95% confidence interval. The output of the confidence

intervals can be used when one is willing to sacrifice SLA in favor of utilization (by choosing the lower 80% or 95%) or decreasing utilization in order to provide better response times (by choosing the higher 80% and 95%).

VI. CONCLUSION

Our system introduced the prediction based on the ARIMA model and evaluated its accuracy of future workload prediction using real traces of requests to web servers from the Wikimedia Foundation. We also evaluated the impact of the achieved accuracy in terms of efficiency in resource utilization and QoS. Simulation results showed that our model is able to achieve an accuracy of up to 91%, which leads to efficiency in resource utilization with minimal impact in response time for users.

In future, we plan to integrate to the architecture a reactive module that can act as a second line of defense against poor QoS by compensating errors in the prediction with ad hoc decision on dynamic provisioning. We also plan to explore more robust techniques for workload prediction, able to predict peak in resource utilization that cannot be fit in the ARIMA model. With these techniques available, we plan to investigate methods for automatic selection of the best approach for workload modeling and load prediction given user-defined accuracy and computational requirement trade-offs.

REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599–616, Jun. 2009

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski,G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," Communications of the ACM, vol. 53,no. 4, pp. 50–58, Apr. 2010.

[3] Q. Zhu and G. Agrawal, "Resource provisioning with budget constraints for adaptive applications in cloud environments," in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC'10). Chicago, USA

[4] N. Bonvin, T. G. Papaioannou, and K. Aberer, "Autonomic SLAdriven provisioning for cloud applications," in Proceedings of the 11th International Symposium on Cluster, Cloud and Grid Computing (CCGrid'11). Newport Beach, USA: IEEE Computer Society, May 2011, pp. 434–443.

[5] J. Yang, T. Yu, L. R. Jian, J. Qiu, and Y. Li, "An extreme automation framework for scaling cloud applications," IBM Journal of Research and Development, vol. 55, no. 6, pp. 8:1–8:12, Nov. 2011.

[6] E. Caron, F. Desprez, and A. Muresan, "Forecasting for grid and cloud computing on-demand resources based on pattern matching," in Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom'10). Indianapolis, USA: IEEE Computer Society, Dec. 2010, pp. 456–463.

[7] M. Sladescu, A. Fekete, K. Lee, and A. Liu, "Event aware workload prediction: A study using auction events," in Proceedings of the 13th International Conference on Web Information Systems Engineering (WISE'12), ser. Lecture Notes in Computer Science, X. S. Wang, I. F. Cruz, A. Delis, and G. Huang, Eds. Berlin, Germany: Springer, 2012, vol. 7651, pp. 368–381.

[8] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," Future Generation Computer Systems, vol. 28, no. 1, pp. 155–162, Jan. 2012.

[9] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, Time Series Analysis: Forecasting and Control, 4th ed. Hobokem, USA: Wiley, 2008.

[10] H. Zhang, G. Jiang, K. Kenji Yoshihira, H. Chen, and A. Saxena, "Intelligent workload factoring for a hybrid cloud computing model," in Proceedings of the 2009 IEEE Congress on Services (SERVICES'09). Los Angeles, USA: IEEE Computer Society, Jul. 2009.

[11] Z. Gong, X. Gu, and J. Wilkes, "PRESS: PRedictive Elastic ReSource Scaling for cloud systems," in Proceedings of the 6th International Conference on Network and Service Management (CNSM'10).Niagara Falls, Canada: IEEE, Oct. 2010, pp. 9–16.

[12] N. Roy, A. Dubey, and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in Proceedings of the 4th International Conference on Cloud Computing (CLOUD'11). Washington DC, USA: IEEE Computer Society, Jul.2011, pp. 500–507.

[13] V. G. Tran, V. Debusschere, and S. Bacha, "Hourly server workload forecasting up to 168 hours ahead using seasonal ARIMA model," in Proceedings of the 13th International Conference on Industrial Technology (ICIT'12). Athens, Greece: IEEE, Mar. 2012, pp. 1127–1131.

[14] V. Nae, A. Iosup, and R. Prodan, "Dynamic resource provisioning in massively multiplayer online games," IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 3, pp. 380–395, Mar. 2011.

[15] S. Pacheco-Sanchez, G. Casale, B. Scotney, S. McClean, G. Parr, and S. Dawson, "Markovian workload characterization for QoS prediction in the cloud," in Proceedings of the 4th International Conference on Cloud Computing (CLOUD'11). Washington DC, USA: IEEE Computer Society, Jul. 2011, pp. 147–154.

[16] R. N. Calheiros, R. Ranjan, and R. Buyya, "Virtual machine provisioning based on analytical performance and QoS in cloud computing environments," in Proceedings of the 40th InternationalConference on Parallel Processing (ICPP'11). Taipei, Taiwan: IEEE Computer Society, Sept. 2011, pp. 295–304.

[17] G. Urdaneta, G. Pierre, and M. van Steen, "Wikipedia workload analysis for decentralized hosting," Computer Networks, vol. 53, no. 11, pp. 1830–1845, Jul. 2009.

[18] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: The forecast package for R," Journal of Statistical Software, vol. 27, no. 3, pp. 1–22, Jul. 2008

[19] M. Arlitt and T. Jin, "A workload characterization study of the 1998 World Cup Web site," IEEE Network, vol. 14, no. 3, pp. 30–37, May 2000.