A Coordinated Reactive and Predictive Approach to Cloud Elasticity

Laura R. Moore, Kathryn Bean and Tariq Ellahi
SAP Next Business and Technology
SAP (UK) Ltd
Belfast, UK
Email: {laura.moore, kathryn.bean, tariq.ellahi}@sap.com

***Abstract*—Based on pay-per-use service-oriented architectures, the cloud computing paradigm promises cost-efficient IT solutions. To meet fluctuating demands efficiently, Platform-as-a-Service solutions offer shared environments with on-demand scalability. It remains an open challenge for service providers to implement elastic scalability mechanisms capable of optimally utilizing resource whilst simultaneously guaranteeing that application performance continues to meet Quality of Service metrics. Typically, cloud providers offer only reactive rule-based mechanisms for triggering scaling actions. We introduce a new elasticity management framework that combines reactive and predictive controllers. Our elasticity controller builds predictive models online based on the reactive rules, representing a natural extension to the common offering. We discuss the underlying architecture of the framework and describe how the controllers operate in tandem and complement each other. We present a case study based on real datasets that demonstrates the feasibility of our real-time cloud capacity framework.**

***Keywords*-elasticity; predictive; auto-scaling; platform-as-a-service.**

I. INTRODUCTION

Cloud computing, with its promise of cost-effective computing for end-users and improved resource utilization for cloud providers, continues to grow in popularity. A recent Gartner report predicts a compound annual growth rate of 36% for Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS) from $7.6B in 2011 to $35.5B in 2016 [1]. This increase in user demand, coupled with new technologies, is driving a dramatic increase in cloud infrastructure scale, heterogeneity and complexity [2][3]. To efficiently handle their resources, cloud providers require intelligent methods of automated dynamic infrastructure management.

One of the key features of cloud computing is elasticity. Elasticity refers to the ability of a system to grow and shrink dynamically such that it only uses resources that are necessary to cope with the current load. This paper presents details of the design and current implementation of a real-time cloud capacity framework, Platform Insights. The particular contribution of the paper is the design of an elasticity controller that:

- Couples reactive and predictive elasticity management techniques and coordinates auto-scaling requests
- Can be used without off-line training
- Utilizes multi-timeframe information to allow short-term auto-scaling decisions to be made in the context of the expected longer-term workload demand

The rest of the paper is organized as follows. Section II presents a summary of related work. Section III gives details on the architecture of Platform Insights. Section IV describes the implementation of Platform Insights, giving an overview of the configuration options and the integrated predictive models. Section V presents a case study in which the ClarkNet [4] and the 1998 World Cup data access logs [4][5] are used to simulate driving the SPECjEnterprise2010 benchmark. Resulting QoS statistics and resource provisioning decisions are evaluated. Concluding remarks are given in Section VI.

II. RELATED WORK

Auto-scaling techniques can be classified as either reactive (the system reacts to changes but does not anticipate them) or predictive (the system tries to predict future resource requirements in order to ensure sufficient resource is available ahead of time) [6]. Reactive rule-based methods define scaling conditions based on a target metric reaching some threshold and are offered by several cloud providers such as Amazon [7] or third party tools such as RightScale [8] or AzureWatch [9]. Beyond static thresholds, [10] proposes a regression method to dynamically adapt thresholds to meet QoS targets, but does not predict future workload.

Predictive auto-scaling approaches tend to be based on time series analysis, control theory, reinforcement learning, or queuing theory. One strategy is to use a workload predictor and then use a performance model to determine the number of servers required to service the predicted demand. A variety of performance models has been proposed in the literature. Examples include the use of splines to map the request rate to the observed percentage of slow requests given the number of active servers [11], queuing networks [12], and cost optimization models [13]. The predictive controller component of Platform Insights also uses workload forecasts and performance models: the workload forecast is used to estimate a mid-term trend in demand, which is used as an input to a performance model mapping workload-per-server to future QoS.

 ISBN: 978-1-61208-271-4

Hybrid methods, coupling reactive and predictive controllers, have also been proposed. Proposals include using a predictive method and a reactive method to determine when to provision resource over a long time-scale (hours and days) and a short time-scale respectively [14], or using a predictive controller to control scale in and a reactive controller for scale out [15][16]. In [15] a regression-based method is used. In [16] the authors base their models on queuing theory and they find that SLA violations are reduced by a factor of 2 to 10 compared to a purely reactive controller.

We also implement a hybrid approach, but our reactive and predictive controllers are both capable of triggering scale in and scale out actions. Auto-scaling decisions are coordinated and conservative policies are applied to avoid premature decommissioning of resource.

III. Architecture of Platform Insights

Typical enterprise applications are composed of a number of services that run on multi-tier architectures. To provide adequate resource to handle client demand, each tier requires monitoring and elasticity. Platform Insights monitors each component of the platform stack individually and evaluates appropriate elastic scaling actions. In Section III-A the architecture of the reactive controller is described, and this is then extended in Section III-B to show how a predictive controller operates in conjunction with the reactive controller.

A. Reactive Elasticity Management

Reactive elasticity management takes place by monitoring scaling rules, which are configured by application architects and administrators. The form of the scaling rules is discussed later in Section IV. This section is dedicated to describing the software agents that make up the reactive auto-scaling alerter component of Platform Insights. Figure 1 shows the steps taken by the system to register a new elasticity rule when it is submitted by an administrator. In its current form, Platform Insights allows rules to be submitted or deleted at any time for running applications, but does not take responsibility for checking that rules do not conflict; this functionality is left to future work. Figure 2 shows the operation of the reactive auto-scaling alerter component. The two figures show the agents that are involved at each stage and their interactions. Each agent is now discussed in turn.

- *The Request Manager* receives requests submitted to the platform by users of the web portal. The web portal has facilities for the application architect to a) monitor resource usage consumed by each instance (real-time or historical), b) configure and manage elasticity rules, c) monitor utilization metrics associated with elasticity rules, and d) receive relevant alerts and/or log messages.
- *The Rule Creator* receives instructions from the *Request Manager* to set up new elasticity rules. When it receives a new scaling rule, it liaises with other components to coordinate the instantiation of the new rule.

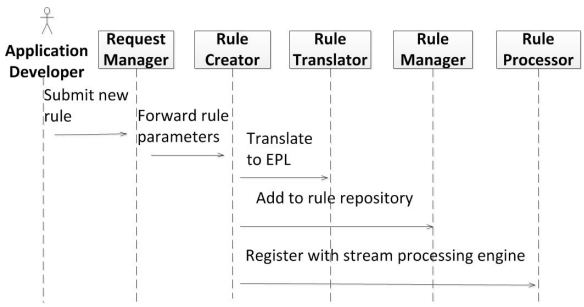

Figure 1. Scaling rule submission sequence diagram.

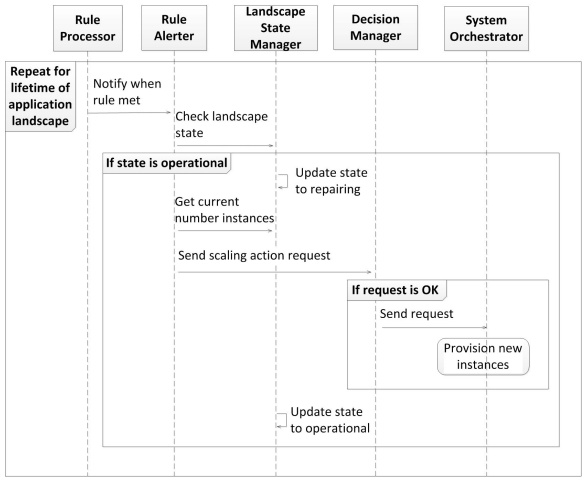

Figure 2. Reactive auto-scaling sequence diagram.

- *The Rule Translator* is responsible for translating the configured attributes of a newly submitted scaling rule into EPL (Event Processing Language) statements that can be monitored by the *Rule Processor*. It maintains a dictionary of pre-set statement templates and parses the incoming data against these templates.
- *The Rule Manager* is responsible for the lifecycle management of elasticity scaling rules. It maintains a repository of scaling rules for each application landscape and of associated EPL statements registered with the *Rule Processor*.
- *The Rule Processor* is based on a complex event processing (CEP) engine. Currently Esper is used as the CEP engine as it is lightweight, can be easily embedded in a Java application and allows new queries to be registered dynamically so that scaling rules can be submitted at any time [17].
- *The Rule Alerter* is responsible for determining what scaling action should be taken upon violation of a scaling rule and broadcasting any relevant information to enable the platform to execute the scaling action.
- *The Landscape State Manager* stores and monitors the application state throughout its entire lifecycle.

 ISBN: 978-1-61208-271-4

Four states are defined for each application landscape: offline, starting, operational and repairing.

- *The Decision Manager* is a centralized component receiving requests from both the *Rule Alerter* and the *Tier Manager* (predictive component, see Section III-B). It is responsible for ensuring coordination of auto-scaling requests; this process is described in more detail in Section IV-D below. Validated requests are broadcast over the messaging bus.
- *The System Orchestrator* listens for the broadcast auto-scaling requests. It is responsible for executing the requests by provisioning and removing server instances.

The particular form of the scaling rules used by Platform Insights is discussed in Section IV below. Rule-based elasticity management only enables the system to scale after some condition has already been met, and so predictive elasticity management is also utilized in Platform Insights.

B. Predictive Elasticity Management

The Platform Insights predictive analysis engine estimates the resource requirements needed by the workload in the near future to satisfy QoS constraints. Platform monitoring data is aggregated on each tier prior to being fed in to the predictive models. Aggregating data at the tier level is acceptable as the platform components are assumed to have load balancers. Esper [17] is used to perform this pre-aggregation of the data (both workload and QoS metrics). More information on the specific data aggregations and predictive models will be given in Section IV-C. Esper listens to the underlying stream of monitoring data, aggregates it as appropriate, and then publishes the aggregated data using the messaging system. Listeners for the predictive models subscribe for all relevant data. The predictive engine comprises the following software agents:

- *The Data Listener* subscribes for relevant data that has been aggregated by Esper and published over the messaging bus and distributes it to the *Data Processor*.
- *The Data Processor* has the responsibility of feeding the data to the appropriate *Model Updater*.
- *The Model Updater* deals with the new data by updating its model and/or doing a prediction using the new data. On the basis of the prediction, it may decide an auto-scaling action is necessary, in which case it sends a request to the *Tier Manager* for assessment.
- *The Tier Manager* evaluates all auto-scaling actions requested for the tier by the *Model Updater*. If the *Tier Manager* agrees that the scaling action is appropriate then a request is sent to the *Decision Manager* (see Section III-A above).

This section has described the underlying architecture of both the reactive and predictive components of the Platform Insights elasticity management framework; the next section gives details on their implementation.

IV. Implementation of Platform Insights

This section discusses the nature of the predictive models built from the scaling rules and how they enable auto-scaling decisions to be made. Section IV-A describes the configuration of the scaling rules, Sections IV-B and IV-C describe the implementation of the rule processing and predictive models respectively, and Section IV-D describes how the *Decision Manager* coordinates auto-scaling requests.

A. Configuration of Scaling Rules

The scaling rule strategy is to first perform some aggregation of metric data pertaining to each server instance over some time window. These per-server values are then further aggregated to a single tier value, which is compared against a threshold value. If the rule condition is met then an action is triggered to add or remove some number of instances whilst staying within some limits. The rules are composed of the following elements. *Metric*: one of the metrics exposed on the server and monitored by the system. *Operator*: the comparison operator to be used in evaluating the metric value; allowed operators are ‘EQ’, ‘LT’, ‘LTE’, ‘GT’and ‘GTE’. *Value*: the value threshold for the metric being observed. *Aggregate Function*: the statistical aggregate function to be used for metric evaluation; allowed values are ‘average’, ‘sum’, ‘median’and ‘raw’(which indicates no aggregation). *Scope*: the metric scope with respect to the all the server instances in the tier; allowed values are ‘min’, ‘max’and ‘sum’. *Time Window*: the length of the sliding time window over which to continuously monitor the metrics and evaluate whether or not they meet the scaling rule condition. *Min Time Between Alerts*: the minimum time between auto-scaling actions. *Limit*: specification of the maximum or minimum number of instances allowed in the tier. *Scale By Type*: used to indicate that scaling should be implemented by changing the number of currently running instances by either a set number or by a given percent; allowed values are ‘Number’and ‘Percent’. *Scale By Value*: the value in units of *Scale By Type* by which to change the number of currently running instances when scaling. *Rule Type*: used to specify whether the rule is based on metric values or on projected values calculated through a linear regression; allowed values are ‘Static’and ‘LinearRegression’. *Time Ahead*: for rules based on projected metric values, this element defines how far into the future to extrapolate the linear fit.

Scale out rules have a further element, *Use As QoS* with allowed values of true or false. If it is true then the scaling rule is additionally used by Platform Insights to form a QoS condition to be taken account of by the predictive controller, and in this case a predictive rule must also be submitted with the following elements. *Workload Metric*: a metric representing workload demand exposed on the server and monitored by the system. *Aggregate Function*: the statistical aggregate function to be used for workload metric evaluation; allowed values are ‘average’, ‘sum’and ‘median’.

 ISBN: 978-1-61208-271-4

Time Window: the length of the batch time window over which the aggregate function is to be applied to the workload metric. *Min Instances* and *Max Instances*: the minimum and maximum number of instances in the tier. *Confidence Level*: the confidence level used by the time series forecaster to compute confidence bounds on the predictions it makes.

B. Rule Processing

Upon submission of a scaling rule, the sequences of events depicted in Figures 1 and 2 take place. If the scaling rule is to be used as the basis of a QoS condition, then a prediction rule is also submitted, which triggers the creation of data aggregation statements and their registration with the Esper engine. These aggregation statements cover the workload metric aggregated over the batch time window requested in the prediction rule and both the workload and QoS metrics aggregated over the sliding time window requested in the scaling rule. The Esper engine again subscribes for relevant data and outputs aggregated data for use as input to the predictive models.

C. Predictive Models

The predictive auto-scaling algorithm currently employed in Platform Insights comprises three models operating on mid-term and short-term timescales. The first model is a time series forecaster that estimates the workload at some future point in time. The model takes as input the total number of requests made by the application (*Workload Metric* is 'number of requests', *Aggregate Function* is 'sum') and predicts the future workload some time later; typically the prediction horizon is set to one hour (*Time Window* is 'one hour'). The *Confidence Level* is set to 95% and used to calculate confidence intervals on the forecast.

By comparing the forecast value against the four-hour moving average value this model also classifies the current workload trend as 'Increasing'(if $>$10% difference), 'Decreasing'($<$-10% difference) or 'Steady'(otherwise). Changes in enterprise workload demand should be observable over a four hour period since typical workload cycles exhibit daily or weekly trends [18].

The second model is an updateable Naive Bayes model that learns the relation between the current workload per server and the current QoS classification. The QoS classification is a binary classification according to whether or not the QoS condition is met or violated. The threshold for making this classification is set to 5% below the actual QoS target value to reduce the risk of under-provisioning, an approach also adopted by others [19]. This model is used to predict mid-term resource requirements taking as input the forecast confidence interval output from the first model.

The third model is also an updateable Naive Bayes model and it maps the total workload per server to a QoS classification some time into the future, typically 30 minutes as this will allow time to build confidence in any output auto-scaling requests and to provision additional servers. This model takes as input the current workload and the trend output from the first model. It is used to estimate the optimal number of server instances required to handle the short-term workload, by finding the minimum number of servers such that there is less than 5% chance of QoS violation in the next 30 minutes. If this estimate, N_{Est}, differs to the current number of servers, N, then an auto-scaling decision is made to add $N_{\mathrm{Est}} - N$ server instances.

The Weka machine learning library is used to implement the predictive models [20]. One of the main reasons for choosing Weka is that it provides classifiers that are updateable incrementally. Because such classifiers can be updated one training instance at a time, in line with the arrival of the new data, this feature is particularly relevant for Platform Insights in analysing steady streams of monitoring data. Platform Insights uses the time series forecasting plug-in and incrementally updateable Naive Bayes models. An in-depth discussion of the implementation and performance of these algorithms is presented in [21]; the focus of this paper is to demonstrate the feasibility of the approach.

D. Coordination of Controllers

When an application starts running, both controllers are activated. The predictive controller can be used without off-line training because it takes advantage of online incremental learning techniques. If the predictive controller is uncertain of what action to take then it does not make any auto-scaling decisions, instead it continues to learn. In these cases auto-scaling decisions can still be made by the reactive controller since it is always running as a stand-by.

Both controllers are capable of triggering scale in and scale out actions. The reactive controller is only able to do so if sufficient time (*Min Time Between Alerts*) has passed since the previous scaling action and if the landscape state is 'operational'. The predictive controller can submit scaling requests at any time. Both controllers act independently but forward their requests to the centralized *Decision Manager* to deliver a coordinated elasticity mechanism.

The *Decision Manager* validates received requests with information it has access to, or can obtain from the *Landscape State Manager*, regarding the current number of instances running, any outstanding scaling requests currently being executed, and specified limits on the number of servers. This validation process may revise the request in several ways. Firstly, the request may be rejected. This can occur if it duplicates a request already being executed, or if it instructs to scale in whilst a scale out action is currently being executed. We choose to implement a conservative policy stipulating that scale out takes precedence over scale in so as to minimize QoS violations. Secondly, the requested number of servers may be modified. This can happen to enforce the specified limits. It will also happen if a) the

 ISBN: 978-1-61208-271-4

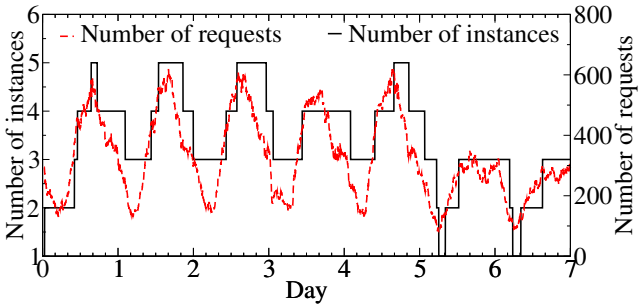

Figure 3. Auto-scaling simulation using ClarkNet traces.

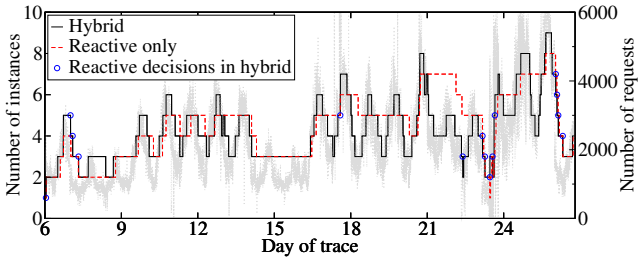

Figure 4. Auto-scaling simulation using 1998 FIFA World Cup trace. Grey line: number of requests in each two minute interval (right axis).

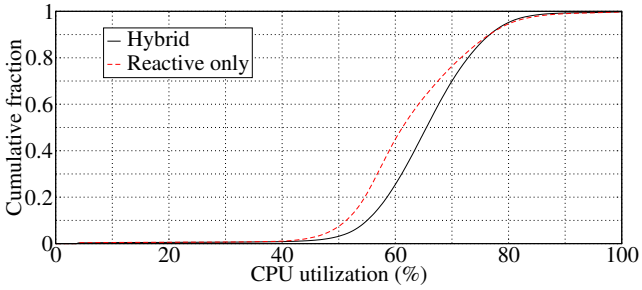

Figure 5. Cumulative distribution of CPU utilization.

request would result in the number of server instances going outside the mid-term resource requirements predicted from the forecast confidence interval, or b) the request instructs the addition of n_2 servers whilst an earlier scale out action instructing the addition of n_1 servers is still being executed, in which case the request is revised to $\max(n_2 - n_1, 0)$.

Upon successful validation, the *Decision Manager* forwards the (possibly revised) request to the *System Orchestrator* for execution. It also updates the time of the last scaling action and updates the state of the application landscape held by the *Landscape State Manager* to 'repairing'. The landscape state only returns to 'operational'once the *Landscape State Manager* detects the requested change in the number of current running instances, indicating that the request has been successfully carried out.

V. Case Study

To evaluate the performance of the elasticity controller, a simulation of the elasticity of the application server tier has been carried out. Two real datasets, the ClarkNet web server trace logs [4] and the FIFA 1998 Word Cup Access logs [4][5], were used to simulate the incoming load to the system. The log files were summarized to extract the number of requests arriving every 2 minutes and then used to simulate driving the SPECjEnterprise2010 benchmark. The benchmark response times were observed to be in excess of the target time of 2 seconds when the CPU utilization went beyond 80%. A scale out rule was configured as: if the minimum median value of CPU utilization over the past 40 minutes is $>$ 80% then increase the number of instances by 1. Similarly for scale-in: if the maximum median value of CPU utilization over the past 60 minutes is $<$ 50% then decrease the number of instances by 1. After auto-scaling, a period of at least the same time window again must pass before the next auto-scaling decision can be made. In the simulation it is assumed that the provisioning of a new instance takes 10 minutes [22]. The QoS condition extracted from the scale out rule was: minimum median CPU utilization over a window of 40 minutes must be $<$ 80%.

The number of requests using the August ClarkNet trace together with the number of simulated running instances are shown in Figure 3. In this simulation, the reactive controller was only responsible for the first two scaling actions; all subsequent scaling actions were generated by the predictive controller. The figure demonstrates that the hybrid controller is capable of making appropriate scaling actions and is stable. The QoS metric is monitored throughout the course of the run: less than 1% of all collected QoS values violated the QoS condition. To quantify this fully, there were 28 two minute periods where the QoS metric went above 80%, all of which were at the very start of the run.

The FIFA World Cup access logs exhibit a higher degree of burstiness than do the ClarkNet logs. Figure 4 shows the auto-scaling decisions made by both the hybrid and the purely reactive controllers, for days 6 to 25 of the logs. Day 26 in the figure is a repeat of day 6. The actions initiated by the reactive controller in the hybrid model (blue circles) are generally taken at times when the workload is different to historical workload, and hence the predictive controller has not built sufficient confidence in its online models, or when the workload exhibits more burstiness than normal, highlighting the advantage of operating the reactive and predictive controllers in a coordinated parallel manner. The figure suggests that the hybrid controller dynamically adjusts resource more appropriately, and hence will result in better utilization, than does the purely reactive controller. Figure 5 verifies this: CPU utilization is consistently maintained within the target range of 50% to 80% ($<$50% less than 3% of the time and $>$80% less than 5% of the time). Again, there are only a few QoS violations: 41 in total, which relate to 3 separate incidents characterized by bursty workloads.

 ISBN: 978-1-61208-271-4

VI. Conclusion and Future Work

In this paper, we described the architecture, design and implementation of a new real-time cloud capacity framework, Platform Insights. Platform Insights is a hybrid elasticity controller employing both reactive rule-based and predictive model-based elasticity mechanisms together in a coordinated manner. The approach has been validated by using traces based on two real datasets to simulate driving a benchmark application. In both cases, Platform Insights was able to provision resource for the application server tier more appropriately than the reactive controller alone, yielding very few QoS violations and maintaining consistently high CPU utilization.

In the short-term, we will carry out further comparisons of our approach with other auto-scaling methods and integrate our framework in a real cloud infrastructure. For future work, we intend to extend Platform Insights to handle multiple QoS objectives at once and to incorporate an algorithm to detect change in workload mix.

Acknowledgment

This research is funded by the CumuloNimbo project in the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. FP7-257993.

References

[1] Gartner, "Market Trends: Platform as a Service, Worldwide, 2012-2016, 2H12 Update," http://www.gartner.com/id=2188816 Accessed 12th March 2013.

[2] IDC, "Quantitative Estimates of the Demand for Cloud Computing in Europe and the Likely Barriers to Up, SMART 2011/0045, Final Report," 2012 http://ec.europa.eu/information_society/activities/cloudcomputing/docs/quantitative_estimates.pdf Accessed 12th March 2013.

[3] L. Schubert and K. Jeffery, "Advances in Clouds: Research in Future Cloud Computing," Expert Group Report, Public Version 1.0, 2012 http://cordis.europa.eu/fp7/ict/ssai/docs/future-cc-2may-finalreport-experts.pdf Accessed 12th March 2013.

[4] The Internet Traffic Archive http://ita.ee.lbl.gov/html/traces.html Accessed 12th March 2013.

[5] M. Arlitt and T Jin, "Workload Characterization of the 1998 World Cup Web Site," Technical Report HPL-1999-35R1, HP Laboratories, 1999. The trace is available from [4].

[6] T. Lorido-Botrán, J. Miguel-Alonso and J.A. Lozano, "Auto-scaling Techniques for Elastic Applications in Cloud Environments," Technical Report EHU-KAT-IK-09-12, 2012.

[7] Amazon Elastic Compute Cloud http://aws.amazon.com/ec2/ Accessed 12th March 2013.

[8] RightScale http://www.rightscale.com/ Accessed 12th March 2013.

[9] AzureWatch http://www.paraleap.com/azurewatch Accessed 12th March 2013.

[10] D. Breitgand, E. Henis and O. Shehory, "Automated and adaptive threshold setting: enabling technology for autonomy and self-management," in *Proceedings of the 2nd International Conference on Automatic Computing (ICAC '05)*, 2005, pp. 204-215.

[11] P. Bodik, R. Griffith, C. Sutton, A. Fox, M. Jordan and D. Patterson, "Statistical machine learning makes automatic control practical for internet datacenters," in *Proceedings of the 2009 conference on Hot topics in cloud computing (HotCloud'09)*, 2009, Article No. 12.

[12] 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 2011 International Conference on Parallel Processing (ICPP '11)*, 2011, pp. 295-304.

[13] N. Roy, A. Dubey and A. Gokhale, "Efficient autoscaling in the cloud using predictive models for workload forecasting," in *Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing (CLOUD '11)*, 2011, pp. 500-507.

[14] B. Urgaonkar, P. Shenoy, A. Chandra, P. Goyal and T. Wood, "Agile dynamic provisioning of multi-tier internet applications," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 3, no.1, pp. 1-39, 2008.

[15] W. Iqbal, M.N. Dailey, D. Carrera and P. Janecek, "Adaptive resource provisioning for read intensive multi-tier applications in the cloud," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 871-879, 2011.

[16] A. Ali-Eldin, J. Tordsson and E. Elmroth, "An adaptive hybrid elasticity controller for cloud infrastructures," in *Network Operations and Management Symposium (NOMS)*, IEEE, 2012, pp. 204-212.

[17] EsperTech Event Stream Intelligence http://esper.codehaus.org/index.html, a product of EsperTech Inc. Accessed 12th March 2013.

[18] D. Gmach, J. Rolia, L. Cherkasova and A. Kemper, "Resource pool management: Reactive versus proactive or let's be friends," *Computer Networks*, vol. 53, no. 17, pp. 2905-2922, 2009.

[19] Z. Gong, X. Gu and J. Wilkes, "Predictive elastic resource scaling for cloud systems," in *2010 International Conference on Network and Service Management (CNSM)*, 2010, pp. 9-16.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I.H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, vol. 11, no. 1, pp. 10-18, 2009.

[21] L.R. Moore, K. Bean and T. Ellahi, "Transforming reactive auto-scaling into proactive auto-scaling,". Accepted for publication, 2013.

[22] A. Li, X. Yang, S. Kandula and M. Zhang, "CloudCmp: comparing public cloud providers," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC '10)*, 2010, pp. 1-14.

 ISBN: 978-1-61208-271-4