Event Aware Elasticity Control for Cloud Applications

Matthew Sladescu, Alan Fekete (supervisor)
School of Information Technologies, The University of Sydney, Australia
National ICT Australia, Sydney
matthew.sladescu@nicta.com.au, alan.fekete@nicta.com.au

***Abstract*— A distinctive component of cloud-based applications is the elasticity control. This component facilitates the adaptation necessary for an application to maintain service quality in the presence of fluctuating demand. Elasticity control achieves this adaptation at runtime by managing the expansion and contraction of resource capacity in response to demand. How to design the rules of elasticity control is a central challenge when deploying cloud-based software. Many application providers express the need to manage the large fluctuations in demand associated with planned events, like marketing events. Existing reactive and predictive elasticity control strategies can be ineffective in managing the surges in workload associated with such planned events. This report will introduce a novel control strategy that integrates expert knowledge about planned events, along with runtime measurements and trend prediction from recent history. We will evaluate how well this strategy can maintain quality of service as planned events alter the load. The initial results presented in this paper are promising and suggest that making an elasticity controller aware of upcoming events is an effective strategy for dealing with event-associated surges in workload.**

Keywords-control; adaptive; cloud computing; autonomic computing; control; elasticity; flash crowd; prediction;

I. INTRODUCTION

Deploying software on a cloud-computing platform is very attractive for many organizations, due to the replacement of capital costs by operational ones, which can scale up (and down) along with demand and income. However, there are distinctive issues that a software designer needs to consider, when software is developed for cloud environments, or migrated to them [1]. The unique characteristic of cloud computing that allows resources to be scaled out or in on demand is known as elasticity. To enable elasticity, one requires a defined set of rules that dictate the time and magnitude of scale out and scale in actions. These rules define the adaptation behavior of an elasticity controller. An elasticity controller that behaves correctly in response to changes like fluctuations in demand can be an enabler for meeting a range of non-functional requirements and customer needs. Many application providers like those mentioned in [2] have identified the need to be able to cost-effectively scale computing resources during planned events, like marketing events, in order to maintain acceptable service levels for their customers. When an announcement for an event, like an advertisement for an upcoming sale, is made there is usually an associated surge in workload known as a flash-crowd. This phenomenon is known as the slash-dot effect [3]. The Znn Exemplar problem [4] promoted by the ICSE SEAMS program in 2010 [5] encouraged research into adaptive systems that can maintain acceptable service levels for applications around the time of a slash-dot event, while at the same time minimizing costs. As discussed in section IV, existing control strategies cannot effectively provision resources to match the flash crowd resource demand that is usually associated with planned events. In Section V and VI, a novel control strategy that makes use of expert knowledge (EK) about upcoming events is introduced, which provides a solution to this flash-crowd problem for planned events. An early initial evaluation for this novel approach is given in Section VII. Section III introduces an abstract model that can be used to describe the reactive; history-based predictive; and EK-based predictive control policies discussed in sections III to VI. This model can be used to encapsulate the behavior associated with decisions and actions made by elasticity controllers, making this behavior transparent to an application developer's efforts in implementing an application's business requirements.

II. MOTIVATION & PROBLEM: MATCHING RESOURCE DEMAND FOR PLANNED EVENTS

One of the most commonly cited customer needs in [2] is the need to be able to match the resource demand generated by planned events, like marketing events, with sufficient acquired resource capacity. Some examples for application providers who need to manage planned events include:

- woot!– "woot-off" events hosted at ad-hoc times can generate "tens of thousands of connections a second".
- iFixit– "teardown" events - known to "nearly cripple" the site with associated surges in resource demand.
- QTAC– Responsible for announcing student results – traffic resulting from student result announcement caused the site to crash [6].

Application providers desire a close match between the resource demand and the acquired resource capacity, so that they can meet QoS agreements for their customers, but also so that they can minimize the costs incurred for acquiring compute resources from the cloud. Ideally, the demand for resources should equal the acquired capacity for all time [7].

When the resource demand exceeds the resource capacity, a system is said to be under-provisioned, during which time, customers experience increased response times, and reduced QoS. This reduced QoS can be detrimental to a company's reputation, as well as to their business. For example, Amazon reports that a 100ms increase in response latency can equate to a 1% loss of sales [8].

When the resource capacity exceeds the resource demand, the application provider is paying more for compute resources than necessary to meet an application's service level objectives (SLOs).

How can one design an elasticity control policy so that resource capacity can match resource demand as closely as possible in the context of workload spikes induced by planned events? Ideally, such a policy would minimize the level of over-provisioning and under-provisioning for a system during event-associated workload spikes. As illustrated in section IV, the existing control policies found in literature and promoted by practitioners cannot anticipate such workload spikes, and because a lag is associated with resource initialization, the system can remain under provisioned during this initialization period.

The main research question then becomes: ***How can one anticipate and provision the right amount of resource capacity on time to match the level of resource demand that is generated by planned events*****?**

This report will introduce a novel expert knowledge (EK) based control policy that can more effectively match the resource demand generated by planned events with available resource capacity.

III. Existing Control Policy Models

Before illustrating the problems faced by existing control policies, we first describe what these existing control policies look like, and also introduce a generic model that can be used to represent these control policies. A control policy is composed from the model constructs shown in Figure 1. This control policy is used to encapsulate the rules that dictate when scale out and scale in actions occur. The execution engine shown on the top of Figure 1 enforces the control policy it is linked to on each execution step. An execution step represents an iteration in the adaptation control loop discussed in [9] as illustrated in Figure 2. Each iteration first retrieves information about the current system context [9] in the form of sensor readings. A condition subsequently uses these sensor readings to determine if the system requires adaptation for the current system context. When a condition evaluates to true, the associated action, (associated through a rule), is executed with a condition result. The action makes use of an action parameter generator to determine action parameters, (like the number of resources to acquire or release), based on the condition result.

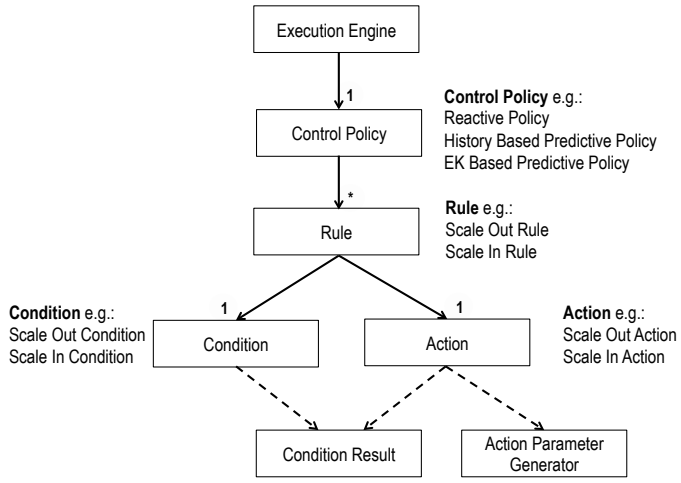

Figure 1 – Control Policy Model

Most of the elasticity control strategies that exist today can be classified [10] as either being reactive or pro-active (predictive). Strategies in both of these classes usually share the goal [9] of keeping service levels like response time acceptable for customers whilst also minimizing costs.

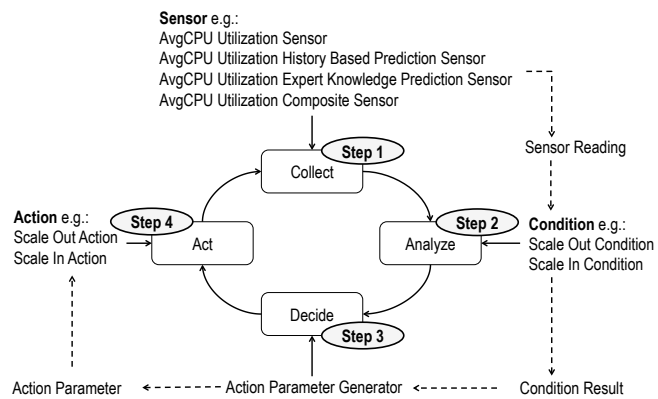

Figure 2 – Adaptation Control Loop

To allow for assessment in how well different control strategies meet this shared goal, three different control policies have been implemented thus far:

1. Reactive Policy: This policy monitors the current resource demand level (D_C) and includes a condition to scale out when D_C exceeds a configured high-level threshold (D_H). A condition to scale in evaluates to true when D_C falls below a calculated low level threshold (determined using the proportional threshold technique described in [11]).
2. History-Based Predictive Policy: As illustrated in the following section, a reactive policy will not always be capable of provisioning compute resources on time to match resource demand (D_C). A history based predictive policy makes use of history-based predictions for resource demand in order to pro-actively allocate

resources ahead of time in order to match an anticipated upcoming surge in workload.

3. Expert Knowledge (EK) Based Predictive Policy: As will be discussed in the following section, history-based predictive policies may also be unable to provision compute resources on time to match resource demand (D_C). An EK based policy aims to go further in provisioning resources on time for planned events. This novel type of control policy will be introduced section V.

IV. Problem: Existing Approaches Cannot Effectively Manage Resource Demand For Planned Events

The reactive and history-based predictive policies described in the previous section can be ineffective in provisioning resources on time for the type of flash-crowds that are associated with planned events. This is due to the resource initialization lag problem documented in [12, 13].

Figure 3 illustrates a scenario that can leave a system using a reactive control policy under-provisioned. In this scenario, the resource demand (D_C) increases, and eventually reaches a configured high-level threshold (D_H) at t_1. At this point in time, a command to request more resources is issued. However, because resources take time (Δt) to initialize, the resources only become available at $t_1+\Delta t$. This leaves the system under-provisioned for a period of time, as highlighted in Figure 3. This means reduced QoS for the system's users.

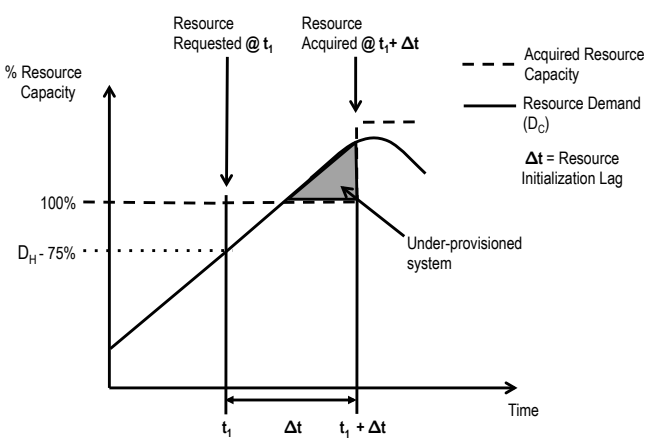

Figure 3 – Reactive Policy – Under-provisioning occurring due to resource initialization lag

History-based predictive policies attempt to compensate for the initialization lag by predicting surges, and requesting resources early so that they are ready for a predicted surge in workload. Figure 4 illustrates conceptually how a history based predictive policy looks at the resource demand history to predict if, when, and by how much resource capacity will be exceeded by resource demand in the future. In the example of Figure 4, the history-based predictive policy predicts that more resource capacity will be required at t_1; even before the resource demand (D_C) exceeds a configured high-level threshold (D_H). History based predictive policies can look at long term history or short term history, or a combination of both to anticipate when scale out is required. Long-term history can be used to predict periodic scaling needs, like scale out during the day and scale down during the night, or scale out during every Christmas season. These long-term history based prediction techniques lack the ability to deal with unforeseen events like ad-hoc sales events or ad-hoc slash-dot type advertisements. Short-term history based predictions try to complement long-term history predictions by identifying when scale out or scale in is necessary based on recent load gradient. However, if the workload surge gradient is steep, the resource initialization lag problem can still yield an under-provisioned system using short-term history predictions, as shown in Figure 4 conceptually, and as shown in the results of section VII. Therefore, history based predictive control policies can still leave a system under-provisioned, meaning reduced service levels for the system. The main research question then becomes: ***How can one anticipate and provision the right amount of resource capacity on time to match the level of resource demand that is generated by planned events*****?**

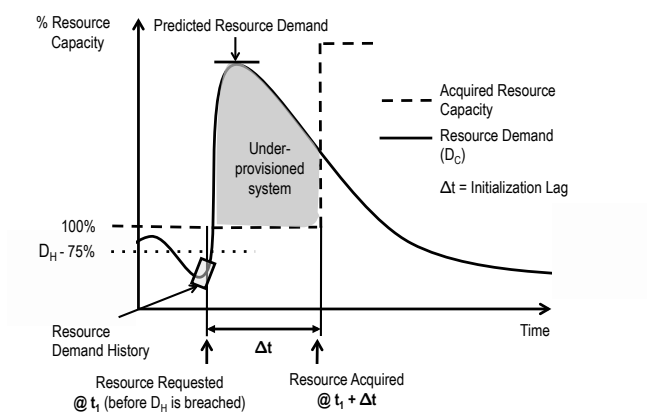

Figure 4 – History-Based Predictive Policy – Under-provisioning occurring due to resource initialization lag

V. Proposed Approach: Use Expert Knowledge To Move Towards Event Aware Elasticity Contol

In this section, we introduce an expert-knowledge based predictive control policy, which can address the problems reactive and history-based predictive control policies have in dealing with ad-hoc, non-periodic marketing events. The time and type of events, like the "woot-off" and "teardown" events mentioned in section II, are usually pre-planned. Experts like marketing managers usually have some level of knowledge about the popularity of an upcoming event that they can contribute. If one could translate this EK about an upcoming event into an estimate for resource demand, one could more effectively provision sufficient compute

resources on time to match the resource demand for an upcoming event. An example of knowledge that an expert can contribute might take the form of:

"The announcement of product A will take place on July 28, 2012 at 11am, and this product will be equal in popularity to product B".

Event characteristics like the event starting time and reference to a similar event can be extracted from the EK given above. If history for the resource demand experienced around the announcement time for product B exists, another event characteristic known as the "shock-level" [14] can also be derived. The shock-level for an event is described in [14] as being the ratio between the maximum hit rate during an eventful period and the average hit rate during non-eventful periods. The event characteristics of shock-level and event starting time can be used to derive an estimate for resource demand for a given time using the resource demand function proposed in [14]. (Note: The function proposed in [14] is limited in only considering the shock level and event starting time characteristics in its computation; the project this report was written for plans to explore more characteristics.) One of the important questions flagged for future research in [9] asks about the reliability of sensor readings. The EK provided might not always be accurate, and current or history-predictive resource demand sensors might indicate that more or fewer resources are required than what is estimated for an event based on EK alone. In such cases, a policy that combines EK based resource demand predictions with history-based resource demand predictions would be more useful in making decisions about how many compute resources are required.

VI. EXPERT KNOWLEDGE CONTROL POLICY

The scale out rule shown in Figure 7 is designed to take advantage of EK about planned events when making scaling decisions. In providing a more reliable prediction for resource demand, the composite prediction sensor used in this rule makes use of multiple readings for resource demand. The first reading used by the composite sensor is the maximum prediction for demand over the next 15 minutes (where the length of this prediction window is configurable) based on expert knowledge of upcoming events.

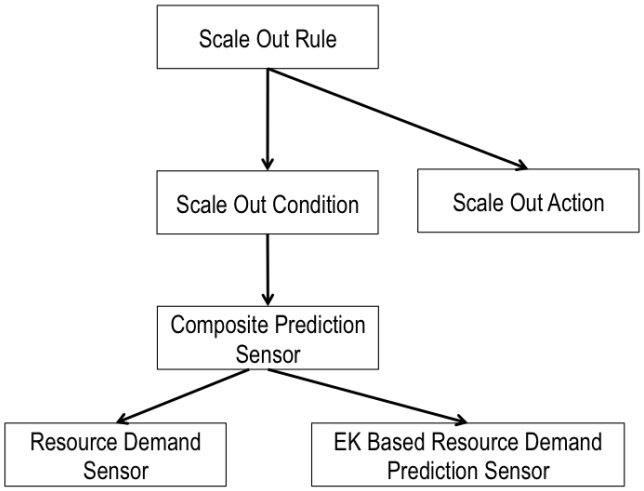

Figure 7 – EK Based Scale Out Rule

As shown in Figure 8, the EK Resource Demand Prediction sensor makes use of a workload spike model similar to the model described in [14], a repository of known upcoming events, and a repository of past events in order to provide predictions for resource demand. The input variables required by the workload spike model will dictate what information is required about upcoming events, and about past events. In this case, the repository of upcoming events will contain details about the event starting time, the event duration, reference to a similar event that has occurred in the past, as well as relative popularity to the referenced event (relative shock-level). The shock level [14] represents the ratio between the maximum hit-rate during an eventful period, and the average hit-rate during a non-eventful period. For events that have occurred in the past, this shock-level can be calculated based on hit-rate history. For upcoming events, an expert can specify how the upcoming event compares in popularity, (more specifically how it compares in shock-level), to an event in the past. This expert knowledge can be used to calculate an expected shock level for an upcoming event. This EK-based shock level, augmented with the earlier mentioned information about upcoming events can be sufficient to parameterize the model in [14] and consequently to provide a prediction for resource demand.

The composite sensor also makes use of a reading that represents the current resource demand to yield a demand prediction. For the experiment described in this report, the composite sensor take the maximum of these two demand readings and reports this as the resource demand prediction so that every opportunity can be taken to reduce under-provisioning levels.

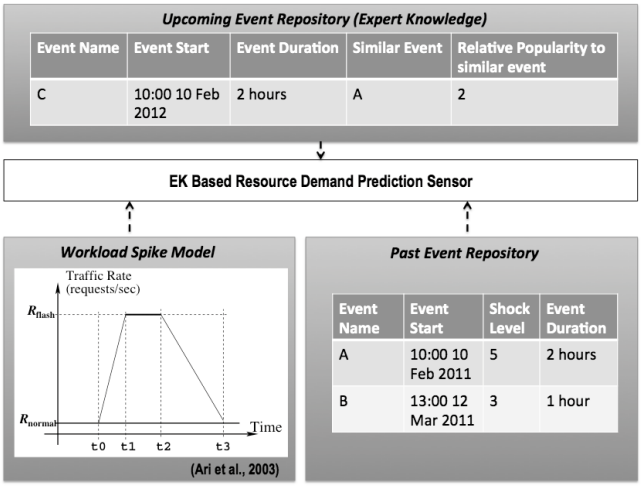

Figure 8 – EK Based Resource Demand Prediction Sensor Inputs makes use of a Workload Spike Model and a repository of Upcoming events

The scale out condition evaluates to true if the demand reading provided by the composite sensor exceeds a configured high-level threshold. When the condition evaluates to true, the amount of excess above the high-level threshold is used to calculate how many more resources are required to maintain a given SLO.

In this report we investigate the effects of using an EK based policy, (that includes a scale out rule similar to the one shown in Figure 7), on the under-provisioning levels of an application. The initial results are provided in the next section and confirm the benefits of using EK to maintain application SLAs. In future work, we aim to also investigate the effects of EK based policies on over-provisioning levels, and how different composite sensor reading combination strategies influence this.

VII. VALIDATION

As an initial step towards providing a proof of concept for the benefits of using EK, an experiment was conducted with the aim of comparing under-provisioning levels across different policies, when an application was subjected to an exponential slash-dot type [15] workload.

A. Experimental Aim

To compare the under-provisioning levels for an application subjected to an exponential workload, when the following control policies are used:

1. *History-Based Prediction Control Policy:*
   In this policy, linear regression is used to predict the maximum demand expected to occur within the next 15 minutes. History from the last 3 minutes, (where one reading of average CPU utilization across EC2 instances is taken every minute), is used to make this prediction.

2. *EK-Based Prediction Control Policy:*
   This policy is the same conceptually as described in section VI, and uses a modified but similar workload spike model to the one described in [14].

B. Experimental Setup

A TPC-W [16] based e-commerce application is hosted on small instances on Amazon EC2. A TPC-W application was chosen for this experiment such that the results of this experiment can be compared to results published for other reactive scaling strategies published in [15]. Requests are distributed to the TPC-W small instances via an EC2 Load balancer. The load balancer receives requests to distribute between the small instances from an httperf [17] workload generator. The workload generator is hosted in a small instance on EC2, and subjects the TPC-W application to a slower surge in workload ($e^{0.5t}$) and a quicker surge in workload (e^{t}). As shown in Figures 10 and 11, t represents the workload partition.

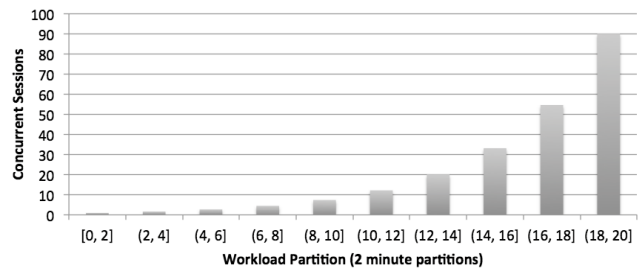

Figure 10 – $e^{0.5t}$ workload profile

Figure 11 – e^{t} workload profile

Each of the workload profiles is composed of 10 discrete workload partitions, where each partition spans an arbitrary 2 minutes. The number of concurrent sessions sending requests to the load balancer increases exponentially with each partition, and each session distributes an arbitrary 20 hits per minute to the load balancer. Exponential workload patterns were chosen since these tend to match the type of workload profile seen for slash-dot type events [15].

The elasticity controller is hosted outside of the cloud, and makes decisions to scale out or scale in based on the control

policy used with the controller. The two control policies that will be attached to the controller for this experiment are the History Based Prediction Control Policy and the EK-based Prediction Control Policy described in the Experimental Aim.

C. Experimental Method

In order to produce a report, which compares the under-provisioning penalty between a controller that makes use of expert knowledge, and one that doesn't, the following steps will be carried out:

1. Provision one virtual machine instance, and host one small instance of the TPC-W web-application in Amazon EC2.

2. Launch the elasticity controller with a history based predictive control policy on a node external to the cloud.

3. Project workload pattern from Figure 10, onto the TPC-W application through the load balancer.

4. Record results for response time, and the number of successful and failed requests.

5. Repeat steps 1 through to 4; however for step 2, use an EK-based control policy.

6. Repeat steps 1 through to 5, while varying the closeness with which the anticipated event workload and the actual event workload match.

7. Repeat steps 1 through to 6 with the workload profile shown in Figure 11.

For each of the experimental runs a log of request response times and failed requests will be produced. This information is used to derive an under-provisioning penalty metric as described in [15]. A higher penalty magnitude corresponds to a higher level of under-provisioning. In this case, the penalty increases with the number of lost requests and the number of requests that exceed a 3 second response time.

In order to develop an initial idea about the influence of using inaccurate expert knowledge with the EK-based policy, as mentioned in step 6, tests where the anticipated workload was higher and lower than the actual input workload were also run.

D. Initial Results

The initial policy comparison results for this experiment are shown in Figure 5. These are early results, and the confidence intervals shown are wider than desirable due to a relatively low number of test runs (at most 3 test runs being included for each under-provisioning reading reported). Further test runs are scheduled to improve these confidence intervals. The bars shown in the darker shade indicate the under-provisioning penalty yielded when the e^t input workload profile is used, and the bars shown in the lighter shade illustrate the under-provisioning penalty yielded when the $e^{0.5t}$ profile is used. The first two bars show the under-provisioning penalty yielded by the history-based predictive policy, and the last four bars indicate the under-provisioning penalty yielded by EK-based predictive policies. Two different EK-based policies were used in order to observe the effect of including inaccuracies in the EK. The two bars in the middle correspond to an EK-based policy that anticipates the $e^{0.5t}$ workload profile, and the last two bars correspond to an EK-based policy that anticipates the e^t profile. For the higher e^t input workload profile, the results indicate that EK based policies always outperform the history-based predictive policy even in the face of inaccuracies in the EK (where the anticipated workload profile was $e^{0.5t}$). For the slower surge in workload $e^{0.5t}$, EK based policies still usually outperform the history-based predictive policy.

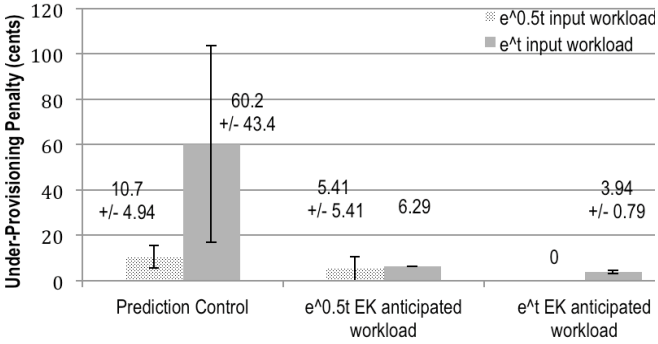

Figure 5 – Under-provisioning Penalty for Predictive vs. EK based control policies

In Section IV, we discussed the problems reactive and history based predictive control policies have in dealing with sudden surges in workload. The results of Figure 5 confirm that, for prediction control, the quicker the surge in workload, the higher the penalty for under-provisioning that is incurred.

In Section V and VI we introduced the concept of an EK based predictive policy that could manage these sudden surges in workload more effectively by making use of prior knowledge (EK) about marketing events. This initial experiment has shown that an EK based control policy can be more effective than history-based predictive policies in provisioning resources on time to match the resource demand from an exponential workload and minimizing the level of under-provisioning that occurs.

VIII. Future Work

This report has shown some promising initial results to indicate that EK is beneficial for application providers who need to scale their resources for planned events. In this

section we elaborate on future work that can build on these initial results.

A. Updates to Validation Experiment

1) Reporting on Over-Provisioning Levels

While minimizing the level of under-provisioning is important, it is also important to minimize the level of over-provisioning experienced when pro-actively provisioning resources to match anticipated resource demand. In future experiments we will report on these over-provisioning levels and explore how various control policies influence these levels.

2) Comparison against well known scaling engine

In order to provide a common, well known, baseline for how well the EK-based prediction policy implementation performs, an Amazon Auto Scaling implementation will be created. Under-provisioning penalty yielded through the use of Amazon's auto-scaling engine [18] will be compared to the penalty yielded when using an expert-knowledge based policy.

3) Test EK based policy with more levels of innaccuracy

In understanding the influence of inaccuracies in the EK used on under-provisioning and over-provisioning levels, we wish to report on more levels of in-accuracy for each test. These inaccuracy levels may take the form of using EK that indicates the required resources will be 25%, 50%. 75%, 125%, 150%, 200% or 1000% of the actual input workload.

4) Use of Real-World Workload Traces

The experiments discussed in this paper make use of an artificial exponential workload. In order to provide a real-world proof of concept, the use of an input workload from the following sources will be explored:

a. **Wikipedia page hit statistics [19]:** analyze resource demand history for a topic that became popular because of an announcement.

b. **Soccer World Cup 1998 Website Traffic Statistics [20]:** in this case events correspond to soccer matches.

c. **eBay bidding workload:** In this case events correspond to the auctions taking place.

5) Run more Test Runs to improve confidence intervals

As mentioned earlier, the results reported for comparing the under-provisioning levels across different control policies include wider than desirable confidence intervals. Further test runs are scheduled to improve on these confidence intervals.

B. Expert Knowldege Demand Prediction Model

For the experiments described in this report, we have made use of the demand prediction model presented in [14]. This model attempts to provide universal-model for flash-crowd workload profiles. There are however, parameters that may be not be common to all flash-crowd marketing events. For example, the 1998 world cup matches [20] not only include a match starting time, (which may be classified as an event time), but also an ending time – which affects when the ramp-down phase begins. This ending time may also be observed for the "woot-off" marketing event mentioned earlier, but is not observed for any of the other event types surveyed. Some flash crowd events may be anticipated by the general public (eg: the impeding launch of the apple iPhone products) and hence may induce an increase in workload prior to the event date. Other flash-crowd events, like the Stephen Alder publications [3] are un-anticipated by the general public, and hence the associated surge in workload would always occur after the event date. This indicates a need to include event characteristics when defining the functions that model flash-crowd event profiles. In future work, we aim to provide a model that can account for different types of event characteristics, and hence provide a more accurate EK-based demand prediction model.

C. Incorporate Reliability Estimates for Expert Knowledge Based Predictions

The expert knowledge used by an EK-based prediction control policy will not always be accurate. When multiple sources for demand prediction are used, (as is the case with the composite prediction sensor discussed in section VI), an estimate for how reliable each prediction is can help in understanding how to combine predictions to yield a more reliable composite prediction.

IX. Conclusion

In order to make elasticity controllers in the cloud more effective and useful for application providers it is imperative that their needs are addressed. In this paper we have identified that many application providers need to host planned events and that application providers usually know the type and time of these events. In reviewing existing elasticity controllers, we have identified a gap in the inability of existing solutions to effectively provision resources on time for such planned events. We've illustrated through promising initial results that expert knowledge about an upcoming event can assist in making more informed elasticity control decisions, and hence, in more effectively absorbing surges in workload that are experienced during such events. In adapting cloud applications to event-associated surges in resource demand, further research is required to understand how expert

knowledge can be used to assist in making elasticity control decisions.

X. ACKNOWLEDGEMENTS

This research is funded by the Australian Government through an Australian Postgraduate Award, and by National ICT Australia through a NICTA Local Postgraduate Award. Amazon Web Services provided a research grant that provided funds for running experiments on Amazon EC2.

REFERENCES

[1] V. Tran, J. Keung, A. Liu, and A. Fekete, "Application migration to cloud: a taxonomy of critical factors," in *International Workshop on Software Engineering for Cloud Computing*, Honolulu, Hawaii, 2011, pp. 22-28.

[2] RightScale. (2011, 2 May). *Case Studies*. Available: http://www.rightscale.com/customers/case_studies.php

[3] S. Adler, "The Slashdot effect: an analysis of three Internet publications," *Linux Gazette,* vol. 38, p. 2, 1999.

[4] SEAMS. (2010, 24 November). *Model Problem: Znn Exemplar*. Available: http://seams.self-adapt.org/wiki/ModelProblem:ZnnExemplar

[5] B. H. C. Cheng, R. de Lemos, D. Garlan, H. Giese, M. Litoiu, J. Magee, H. A. Müller, M. Pezzè, and R. Taylor, "Fifth workshop on software engineering for adaptive and self-managing systems (seams 2010)," in *Software Engineering for Adaptive and Self-Managing Systems*, Cape Town, South Africa, 2010, pp. 447-448.

[6] T. Chilcott. (2012, 1 April). *Students get QTAC texts but website unavailable as 1000 hits a second cause it to go slow*. Available: http://www.couriermail.com.au/news/queensland/students-await-countdown-to-37000-qtac-tertiary-studies-places/story-e6freoof-1226241877096

[7] J. Weinman. (2011, 2 May). Time is Money: The value of "On Demand". Available: http://www.joeweinman.com/Resources/Joe_Weinman_Time_Is_Money.pdf

[8] G. Linden. (2006, 12 March). *Make Data Useful*. Available: https://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-28.ppt

[9] B. H. C. Cheng, H. Giese, P. Inverardi, J. Magee, and R. De Lemos, "Software engineering for self-adaptive systems: A research road map," *Software Engineering for Self-Adaptive Systems,* vol. 8031, 2008.

[10] J. K. Sadeka Islam, Kevin Lee, Anna Liu, "An Empirical Study into Adaptive Resource Provisioning in the Cloud," in *International Conference on Utility and Cloud Computing*, Chennai, India, 2010.

[11] H. C. Lim, S. Babu, and J. S. Chase, "Automated control for elastic storage," presented at the Proceeding of the 7th international conference on Autonomic computing, Washington, DC, USA, 2010.

[12] A. Li, X. Yang, S. Kandula, and M. Zhang, "CloudCmp: comparing public cloud providers," presented at the Proceedings of the 10th annual conference on Internet measurement, Melbourne, Australia, 2010.

[13] N. Yigitbasi, A. Iosup, D. Epema, and S. Ostermann, "C-Meter: A Framework for Performance Analysis of Computing Clouds," presented at the Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009.

[14] I. Ari, B. Hong, E. L. Miller, S. A. Brandt, and D. D. E. Long, "Managing flash crowds on the Internet," in *International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems*, 2003.

[15] S. Islam, K. Lee, A. Fekete, and A. Liu, "How A Consumer Can Measure Elasticity for Cloud Platforms, Technical Report 680," The University of Sydney, 2011.

[16] D. A. Menascé, "TPC-W: A benchmark for e-commerce," *Internet Computing, IEEE,* vol. 6, pp. 83-87, 2002.

[17] D. Mosberger and T. Jin, "httperf—a tool for measuring web server performance," *ACM SIGMETRICS Performance Evaluation Review,* vol. 26, pp. 31-37, 1998.

[18] AWS. (2011, 2 May). *Auto Scaling*. Available: http://aws.amazon.com/autoscaling/

[19] W. Foundation. (2010, 14 March). *Wikipedia Visitor Statistics, raw hourly logfiles for December 2007*. Available: http://www.archive.org/details/wikipedia_visitor_stats_200712

[20] M. Arlitt and T. Jin, "A workload characterization study of the 1998 world cup web site," *Network, IEEE,* vol. 14, pp. 30-37, 2000.