Towards workload-aware cloud resource provisioning using a multi-controller fuzzy switching approach

Amjad Ullah

Division of computer science and Mathematics,
University of Stirling, Stirling, UK
E-mail: aul@cs.stir.ac.uk

Jingpeng Li

Division of computer science and Mathematics,
University of Stirling, Stirling, UK
E-mail: jli@cs.stir.ac.uk

Amir Hussain

Division of computer science and Mathematics,
University of Stirling, Stirling, UK
E-mail: aul@cs.stir.ac.uk

Abstract: Elasticity enables cloud customers to enrich their applications to dynamically adjust the underlying cloud resources as per their needs, in order to minimize the cost of infrastructure as well as to satisfy their performance goals. Over the past few years, a plethora of techniques have been introduced in order to implement elasticity. Control theory is one such technique that offers a systematic method to design feedback controllers to implement elasticity. A number of proposals based on feedback controller concepts have been introduced in the recent past in order to guarantee the QoS needs of a system deployed over a cloud. Many of these are based on the use of a single controller approach of various types, such as adaptive and fixed. However, for systems that operate in time-varying and unpredictable operating conditions, it becomes difficult for such approaches to perform effectively at all times, in order to comply with the system stated performance goals. The systems deployed over cloud are subject to unpredictable workload conditions that vary from time to time, e.g. an e-commerce website may face higher workloads than normal during festival or promotional schemes. This paper exploits the novel use of a recently developed multi-controller based approach, where each controller is specifically designed for one operating region. Moreover, the use of fuzzy logic is exploited to enable qualitative specification for the selection of the most suitable controller in runtime, based on system current behaviour. Initial experimental evaluation in comparison with the conventional single-controller approach demonstrates that our proposed method enhances the capability of an elastic application to comply with system performance goals.

Keywords: cloud elasticity; control theory; multi-controller; fuzzy logic; auto-scaling; dynamic resource provisioning.

Reference to this paper should be made as follows: Ullah, A., Li, J. and Hussain, A. (xxxx) 'Towards workload-aware cloud resource provisioning using a novel multi-controller fuzzy switching approach', *International Journal of High Performance and Networking*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Amjad Ullah received his MSc degree in Advance Distributed Systems from University of Leicester in 2011. He is currently doing a computing science PhD. His research interests include dynamic resource provisioning in cloud computing, fuzzy logic and feedback controllers.

Jingpeng Li is presently a Reader at the Division of Computer Science & Mathematics, University of Stirling, UK. He received the M.Sc. degree in computational mathematics from Huazhong University of Science and Technology, China, in 1998, and the Ph.D. in operational research from University of Leeds, U.K., in 2002. His research areas include intelligent transport scheduling, metaheuristic, multi-objective decision making, search methodologies, machine learning, stochastic process, fuzzy logic, and image process.

Amir Hussain obtained his BEng in 1992 and PhD in 1997 from the University of Strathclyde in Glasgow, UK. Following a Research Fellowship at the University of Paisley, UK (96-98), and a research Lectureship at the University of Dundee, UK (98-00), he joined the University of Stirling in 2000, where he is currently Professor of Computing Science, and founding Director of the COSIPRA Research Laboratory. He is an Associate Editor of the IEEE Transactions on Neural Networks and Learning Systems, founding Editor-in-Chief of Springers Cognitive Computation journal, Springer/BioMed Centrals Big Data Analytics journal, SpringerBriefs in Cognitive Computation and the Springer Book Series on Socio-Affective Computing.

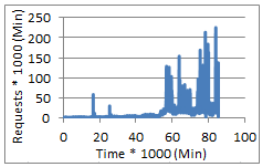
1 Introduction

In this modern world of web, Software as a service (SaaS) applications such as e-commerce websites, news portals and social networking websites are experiencing unpredictable workloads. These can occur due to various real happening events, time of festive or promotional offers, etc. Some examples of such scenarios are: (1) Facebook experienced an increase of 10 times in users within three days with an average 20,000 registrations per hours Jamshidi et al. (2014); (2) 2,500% increase on load was observed at Al-jazeeras news website during the fourth day of the Egyptian revolution in 2011 Kihl et al. (2013); (3) The workload of an online store can be of a seasonal pattern, where an increase in load can be observed in a particular season such as Christmas Garside (2013), etc. Such applications are very important and service providers do not want their applications to suffer from any performance related issues or service disruption. Moreover, they are also interested to use underlying computing resources as efficiently as possible, to save the operating cost.

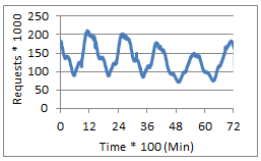
Cloud computing is the latest technology that enables the dynamic selection of services Wang et al. (2016), real time event management Kostantos et al. (2015), online monitoring of resources Lu et al. (2016) and more importantly, the dynamic readjustment of the resources to meet the application demands. Cloud elasticity is the key behind that enables the dynamic readjustment of resources. The changes in resources are usually require in response to environmental changes (E.g. due to workload fluctuations), such that the resources match to the demands as closely as possible. The core purposes of elasticity include Galante & De Bona (2012): (1) Performance: to avoid the degradation of system performance; (2) Infrastructure capacity: to increase the capacity of local resources; (3) Cost: to reduce the running operating cost; (4) Energy: to save the energy consumed. The cloud consumer has to provide an elastic system that implements some auto-scaling mechanism.

There are various auto-scaling techniques available. The most common approach usually adopted in public clouds is the threshold based rules. The rules follow the condition-action pattern. The conditions are based on the performance metrics such as CPU utilization, message queue length, etc, whereas a rule specifies the scaling action. For example, if the average CPU utilization is more than 70 % then add 2 virtual machines (VM). This approach is very easy to use but it suffers from two key issues (1) An in depth understanding of the system is required to quantitatively set the thresholds; (2) they are unable to cope with uncertainty due to unpredictable events Jamshidi et al. (2014).

Control theory is another technique that provides a systematic method to design feedback controllers for auto-scaling. Such feedback controllers are designed to be stable in order to avoid oscillation and settle quickly to the steady state by appropriately responding



(a) World cup 1998 trace log



(b) Wikipedia trace log

Figure 1: Real workload examples

to disturbances. They are better for achieving service level objectives, such as response time or throughput Abdelzaher et al. (2008). Over the last few years, feedback controllers have been used for cloud resource provisioning Lim et al. (2009), Lorido-Botran et al. (2014). In general practice of engineering and computing science, control theory is also exploited to achieve target performance objectives. Some examples amongst others include web servers Abdelzaher et al. (2002), Gandhi et al. (2002), database servers Parekh et al. (2004), cache storage systems Karlsson et al. (2005), etc. Feedback controllers enable the computing system to adapt to workload changes at runtime. However, a badly designed controller may result in oscillation and instability Zhu et al. (2009). Thus, careful attention is required while designing a closed-loop control system. Some of the recently identified challenges related to the use of feedback controllers for achieving performance objectives, include difficulty in constructing a system model and the non-linear behavior of the system Zhu et al. (2009). More specific challenges in the context of cloud computing includes uncertainty (such as difficulty in designing elasticity rules and inaccuracy of monitoring tools), unavailability (such as detailed knowledge about cloud environments and workloads) and the unpredictable workload Farokhi et al. (2015). Thus, new methodologies and techniques are required to be explored in order to tackle these challenges.

The majority of control theoretic approaches to resource provisioning are based on the use of one controller, either adaptive or fixed Lim et al. (2009), Lorido-Botran et al. (2014). Such approaches are based on the use of a single system model that captures the behavior of the system over the entire operating period. As mentioned earlier, a modern web can face varying workload at different time. Some examples of the real workload can be seen in Figure 1. Different workload patterns can be observed at different times, such as Figure 1a demonstrates the flash crowd effect with peak burst in workload traffic, whereas Figure 1b shows diurnal pattern where the workload in day time is high compared to night. While considering such scenarios with a highly unpredictable workload, is it possible to design a single system model that performs well all the time? A definite answer to this question is difficult. Therefore, alternative methods of control theory are required to achieve performance objectives for highly unpredictable computing systems, as existing research in this regard is still in its early stages, despite recent

progress reported in the use of control theory for self-adaptation Farokhi et al. (2015).

The Multi-Modal Switching and Tuning (MMST) Narendra et al. (2003) is an extended adaptive control theoretic technique. It is based on the idea of having multiple system models where the selection of most suitable model is realized at runtime. This approach is suitable for systems of multi behaviour and has been also exploited for QoS management Patikirikorala et al. (2011).

Here we report a similar idea that exploits the use of a multi-controller based approach for cloud resource provisioning in combination with fuzzy logic switching. In this case, the individual controllers are specifically designed for one operating region. Whereas, the selection of a suitable controller at runtime is achieved using a switching mechanism based on fuzzy logic. The key reason of using fuzzy-logic is to incorporate uncertainty and the qualitative selection of suitable controllers among the array of controllers.

Experimental results obtained using this methodology demonstrate better system performance in comparison with the use of a single controller, which cannot guarantee better performance in the presence of high variability in workload and on account of the non-linear nature of cloud applications.

The rest of the paper is organized as follows. The background study of the related concept is given in Section 2. In Section 3 we introduce the multi-controller methodology for resource provisioning that includes the framework, control policy and the switching mechanism. Section 4 describes the experimental evaluation of the introduced methodology in comparison with the conventional approach. Section 5 describes related work whereas Section 6 concludes the paper.

2 Background Study

2.1 Feedback Controller for Elasticity

A control theoretic approach to implementing elasticity uses a feedback loop model, where a controller monitors the inputs and outputs of the system to controls the output around some desired value. Generally, such control can be used to satisfy a constraint or guarantee an invariant on the outputs of the system Ghanbari et al. (2011). More specifically, the controller has to maintain the value of a controlled variable, e.g. CPU utilization or response time, close to a desired value by adjusting a manipulated variable e.g. the number of virtual machines (VM) Lorido-Botran et al. (2014). The input to the target system is the controlled input/manipulated variable, whereas the controlled variables are normally measured by sensors.

Figure 2 depicts the mechanism of the feedback controller, where it observes the system output to correct any deviation from the desired value. Any changes in the manipulated variables are applied using the target

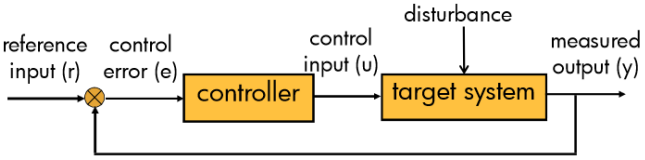


Figure 2: Block diagram of feedback control system adapted from Zhu et al. (2009)

system. The descriptions of various elements are the following.

The reference input such as CPU utilization is the desired value of the control variable where the error is the difference between the desired and measured value. The control input is the manipulated variable, such as number of virtual machines, which will change the behaviour of a system. The disturbances represent the various changes in the system e.g. the incoming workload where the measured output is the current measurement of the control variable retrieved from sensors. The design of a control system is composed of two steps, i.e. (1) the formal construction of a system model that determines the relationship between input and output; (2) design of the controller based on the model obtained. Whilst using the principles of control system design, various feedback controllers are introduced for resource provisioning in the cloud. These can be categorized as follows:

- Fixed gain controllers: The class of controllers, where the gains of the controller remains fixed over the entire operating time. The relationship model of the input-output are obtained off-line. This class of controller is very popular because of its simplistic nature Lorido-Botran et al. (2014). Proportional Integral Derivative (PID) is the most common controller from this category. The control law followed by PID controller is outlined below:

$$u_{k+1} = k_p e_k + K_i \sum_{j=1}^{k} e_j + K_d(e_k - e_{k-1}) \quad (1)$$

Consider the example of CPU utilization as control variable and the number of VM as manipulated variables of the system, the u_k is the new number of VM, e_k is the control error i.e. the difference between the desired CPU utilization and measured CPU utilization. Whereas K_p, K_i and K_d are the gain parameters. There are various well established methods available to derive these gain parameters. However, they can be derived off-line and remains fixed during the entire period of operation. Various available resource provisioning proposals include integral approaches Lim et al. (2009, 2010), Proportional Integral approach Park & Humphrey (2009) and PID approach Zhu & Agrawal (2010). Apart from its simplistic and easy to design feature, PID controller works well in achieving the desired performance for systems with little variability in workloads Patikirikorala & Colman

(2010). However, the performance suffers for systems with dynamic and unpredictable workload conditions Patikirikorala et al. (2011). Thus the use of PID cannot achieve better performance for application with mix-workload, where it changes over the lifetime of the application.

- Adaptive controllers: These types of controllers have the ability to dynamically adapt them by adjusting the parameters of the controller with respect to changes in the environment. Self-tuning PID controllers and self-regulators are common examples of such controllers Lorido-Botran et al. (2014). Some proposals based on adaptive controllers for resource provisioning include Ali-Eldin, Tordsson & Elmroth (2012), Ali-Eldin, Kihl, Tordsson & Elmroth (2012). The nature of adaptability in this type of controllers address some limitation of fixed gain controllers and are suitable for systems with slowly varying workload Patikirikorala & Colman (2010). However, it still suffers from certain problems such as being unable to cope with sudden burst in workload and system with highly changing conditions Lorido-Botran et al. (2014), Patikirikorala & Colman (2010). Moreover, it suffers from high computational cost because of the estimation of parameters on runtime Patikirikorala et al. (2011).

Model predictive controllers are another type of feedback controllers that incorporate predication features into the controller. These techniques follow a proactive approach of elasticity to enrich the underlying application to foresee future workloads for making elastic decisions. Such controllers (e.g. look ahead controller Roy et al. (2011)) solve the optimization problem by taking a cost function to maintain the output value of the control closer to a desired value.

Designing a single controller, which works well for all kinds of applications, is not feasible, as there is no optimal single controller Ali-Eldin et al. (2013). Therefore, existing methods can be extended in order to determine a better solution. One such approach is to look into the multi-controller based approach. Some of the available such proposals include Patikirikorala et al. (2011), Ali-Eldin et al. (2013). These proposals are further discussed in related work.

2.2 Fuzzy Logic

Fuzzy Logic refers to a computing approach, based on the notion of degree of truth rather than true/false. Fuzzy logic has also been used as a tool to deal with scenarios of decision making in the presence of uncertainty and imprecise information Zadeh (1996). It is one kind of many-valued logic that is based on approximate reasoning rather than fixed reasoning like Boolean logic. Compared to Boolean logic a variable can take a value in the range 0 to 1 rather than true or false. This value determines the degree of membership in the fuzzy set. An important use of the fuzzy logic theory is its employment into the qualitative decision making process by designing the rule based system. Such a system enables qualitative reasoning, because the rules can be made from meaningful words/labels that are easily understood by humans.

The important component of any rule based system is the knowledge base that represents the knowledge of the underlying problem, which is a collection of rules in the form of *if-then*, made using fuzzy logic statements. The *if* part of the rule is referred antecedent and the *then* part is called consequent. Applying fuzzy logic to any real world application requires three steps Bai & Wang (2006): (1) Fuzzification: the conversion of data often referred to classical or crisp into fuzzy membership functions, which are represented with linguistic variables; (2) Fuzzy inference: the combination of the various membership functions and the rules to obtain the fuzzy output; (3) Defuzzification: the method of converting the fuzzy output to crisp value is called defuzzification. The three commonly used techniques for this purpose are Centre of Gravity, Mean of Maximum and Height method Bai & Wang (2006).

3 Resource Provisioning Using a Multi-controller Approach With Fuzzy Switching

The multi-controller approach is used for complex dynamical systems, wherein an array of controllers is designed such that each controller is optimized to achieve better performance in a particular region e.g. autonomous vehicle control systems **?**. This research inspires from Abdullah et al. (2007), Hussain et al. (2012), where a multi-controller framework is developed to simultaneously control the various component of an autonomous vehicle such as throttle and brake to achieve a desired speed and track.

The key idea behind this proposed control method is the expert oriented distribution of the application workload/arrival rate into various categories. This categorization then enables the design of individual controllers for each category. The fuzzy logic switching enriches the framework to automatically select the most suitable controller intelligently, keeping in view the current state of the system. Thus, overall, it provides more fine grained control over resource provisioning from time to time. Figure 3 represents the structure of this framework where the details of each component is given below,

3.1 System Monitoring

The monitoring in an elastic controller makes use of the cloud provider Application Programming Interface (API) to get access to various system performance metrics such as CPU utilization, request queue length,

etc. The scaling decisions are dependent on these performance metrics, because they determine system behaviour at that point of time. This component of the framework is responsible to keep track of all system measurements. These measurement describe the current system behaviour, which is then utilized by *Fuzzy Logic* component to make decision suitable at that point of time.

3.2 Control Policy

The control approach followed in this work is similar to other control theoretic approaches for related problems such as Lim et al. (2009, 2010), Padala et al. (2007), except where we have used multiple controllers and a switching methodology. An integral control is utilized, because it removes the steady state errors Lim et al. (2009). The performance metric used is CPU utilization. The control law will adjust the number of VM in order to keep the overall CPU utilization at a desired level to achieve the performance goals. Equation 2 defined an integral control,

$$u_{t+1} = u_t + K_i * (Y_{ref} - Y_t) \tag{2}$$

The output of the controller (u_{t+1}) at each iteration represents the new number of VM that determines the scaling action (i.e. up or down) and is calculated on the basis of current number of VM (u_t). The parameter K_i is the integral gain parameter that can be estimated off-line either by using methods like Ziegler Nichols, root locus or an empirical method Parekh et al. (2002). The desired CPU utilization is represented by y_{ref} where y_t is the measured CPU utilization, which is obtained from system monitors and their difference is called control error.

The three controllers employed are termed *Lazy*, *Moderate* and *Aggressive*. In this case, each of these controllers uses the same control law. However, they are designed to be operated in different operating regions.

These operating regions are basically the segregation of the workload/arrival rate into three levels i.e. Low, medium and high realized using the expert based categorization adapted from Jamshidi et al. (2014). However, the selection of the controller is not only dependent on arrival rate but other parameters too. The

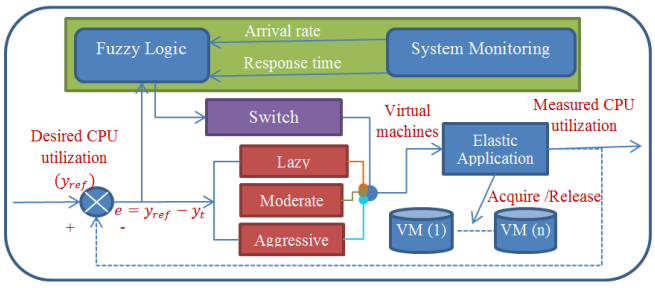


Figure 3: Resource provisioning framework using multi-controller with fuzzy switching

gain parameter for each controller can then be obtained empirically by using workload of that level. The ranges for each region are categorized using the fuzzy set and adapted from the study carried out in Jamshidi et al. (2014). The details of workload ranges and controller selection mechanism are provided in the next section.

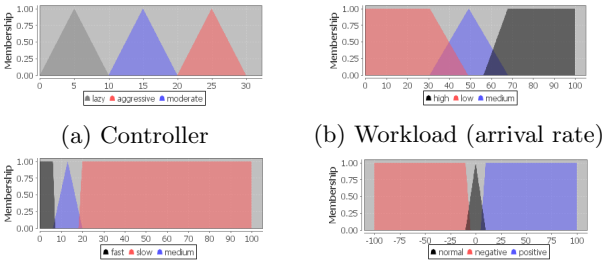
3.3 Controller Switching

The switching decision determines the selection of a suitable controller at each instant of time on the basis of the current system behaviour. The switching part is constructed using a fuzzy logic-based system. The construction of any fuzzy logic system consists of three steps, i.e. extracting domain knowledge, defining membership functions, and rules. For this research work, we adapted the domain knowledge and corresponding membership functions from the research carried out in Jamshidi et al. (2014), where a fuzzy elastic controller was constructed that performs resource provisioning decisions based on the workload and system response time.

- Domain (Elasticity) knowledge: In Jamshidi et al. (2014), the knowledge base is constructed using domain experts, i.e. architects and administrators. Our adaptation of this work is only the extraction of the knowledge with respect to workload and response time. They have constructed a fuzzy set of five members for each fuzzy variable i.e. workload and response time. The set for workload are *very low*, *low*, *medium*, *high* and *very high*. Similarly for response time they have *instantaneous*, *fast*, *medium*, *slow* and *very slow*. The ranges for these linguistic variables are defined from a range [0, 100]. For further details refer to Jamshidi et al. (2014). Here in this research work, we reduce the levels from five to three levels by merging (1) In workload - (*very low*, *low*) into *low* and (*fast*, *very fast*) into *high* where (2) In response time (*fast*) is changed into *medium* where (*medium*, *low* and *very low*) is changed to *low*. These changes are made to reduce the number of switching rules as well as to characterize the workload into three levels where each control can be tune with respect to that kind of workload (arrival rate). Apart from these two linguistic variables, we also consider control error which is the difference between desired CPU utilization and measured CPU utilization as following,

$$e_t = Y_{ref} - Y_t \tag{3}$$

The control error is divided into three linguistic variables, i.e. *positive*, *normal* and *negative*. The *positive* means that the current CPU utilization is less than the desired, the *negative* specifies that it is more than the desired level, and the *normal* specifies that either the error is 0 or within a margin of uncertainty due to noise or inaccuracy in



(a) Controller (b) Workload (arrival rate)

(c) Response time (d) Control error

Figure 4: Membership functions

the measurement. The full error range is [-100,100]. The final ranges for all variables are given in Table 1.

Fuzzy variable	Set member	Range
Workload(Arrival Rate)	Low	0 — 48.9
	Medium	30.7 — 67.94
	High	56.41 — 100
Response time	Instantaneous	0 — 7.2
	Medium	6.1 — 20
	Low	18.2 — 100
Control error	Negative	-5 — -100
	Normal	-10 — +10
	Positive	+5 — +100

Table 1 Ranges for fuzzy variables

- Membership functions: The next step is to define a method that determines how the crisp input is converted into fuzzy values. This is achieved through the membership function. The membership function defines the degree of the crisp input against its linguistic variables in the range of 0 to 1. Defining membership function for both input and output variables are the first step of fuzzification Bai & Wang (2006). In this research the membership function is adapted from Jamshidi et al. (2014) with a modification to comply with the changes made in the domain knowledge. Figure 4 represents these membership functions.

- Fuzzy rules: The fuzzy rules describe the relationship between inputs and outputs. In this case, the inputs are arrival rate, response time and control error, whereas the output is one of the controllers. In order to obtain the fuzzy rules for switching mechanism, we initially listed every possible combination of inputs and outputs. We then performed the following steps in order to obtain the final switching rules,

  1. Removed all those rules where control error is normal. The normal control error means that no elastic decision is required at this instant of time because the current CPU Utilization is within acceptable reference range.
  2. Removed all those rules where control error is positive and response time is either medium or slow. The positive control error means that a scale down operation is required and such case can only occur when the existing performance is instantaneous. Thus only 9 possible rules are left in this category, from which those 3 are selected that produce better results.
  3. For the scale up operation , we considered rules based on the following two guidelines

     i if current performance is poor then select those rules that react quickly.

     ii if current performance is instantaneous then consider saving cost.

The final rules obtained are given in ***Fuzzy rules*** box where Figure 5 represents their illustration of how the controller are selected in response to the input combinations. The range here for the *Response time* variable is reduced to **25** here for display purposes as the output from that point onward is always *Aggressive*.

Fuzzy rules Switching mechanism

1: IF arrivalRate IS high AND responseTime IS instantaneous AND error IS positive THEN controller IS lazy;
2: IF arrivalRate IS medium AND responseTime IS instantaneous AND error IS positive THEN controller IS moderate;
3: IF arrivalRate IS low AND responseTime IS instantaneous AND error IS positive THEN controller IS aggressive;
4: IF arrivalRate IS high AND responseTime IS instantaneous AND error IS negative THEN controller IS moderate;
5: IF arrivalRate IS high AND responseTime IS medium AND error IS negative THEN controller IS aggressive;
6: IF arrivalRate IS high AND responseTime IS slow AND error IS negative THEN controller IS aggressive;
7: IF arrivalRate IS medium AND responseTime IS instantaneous AND error IS negative THEN controller IS moderate;
8: IF arrivalRate IS medium AND responseTime IS medium AND error IS negative THEN controller IS aggressive;
9: IF arrivalRate IS medium AND responseTime IS slow AND error IS negative THEN controller IS aggressive;
10: IF arrivalRate IS low AND responseTime IS instantaneous AND error IS negative THEN controller IS lazy;
11: IF arrivalRate IS low AND responseTime IS medium AND error IS negative THEN controller IS moderate;
12: IF arrivalRate IS low AND responseTime IS slow AND error IS negative THEN controller IS aggressive.

The complete process of the fuzzy part works as follows. The *System Monitoring* component measures the input values, which is then fuzzified using the membership function. The fuzzy inference engine will determine the fuzzy output from the rules using defuzzification process to identify the suitable controller at that point of time. The *Switch* then enables the

suitable controller as per the decision made by *Fuzzy Logic* component, where the elastic application executes the scaling decision as per the output of the controller.

The defuzzification process ensure the final output even in case of multiple rules are active at a time. E.g. Consider the scenario in Figure 6, where for a given input two rules are active i.e. *Rule 1* and *Rule 2*. The output of both rules conflict with each other i.e. *Rule 1* suggests *Lazy* controller where *Rule 2* suggests *Moderate* controller. We have used the *Centre of Gravity* method as the defuzzification method because of its popularity and usage in real applications Bai & Wang (2006). This method calculate the area of membership functions for all the active rules against the range of fuzzy output variable, which in this case is controller and then calculate the centre of area using Equation 4.

$$CoG = \frac{\int_a^b f(x).xdx}{\int_a^b f(x)dx} \tag{4}$$

Solving Equation 4 for the normalized input values of Figure 6 i.e. when (arrivalRate=63, responseTime=3 and error is positive) gives the output value **8.6**, which represents *Lazy* controller.

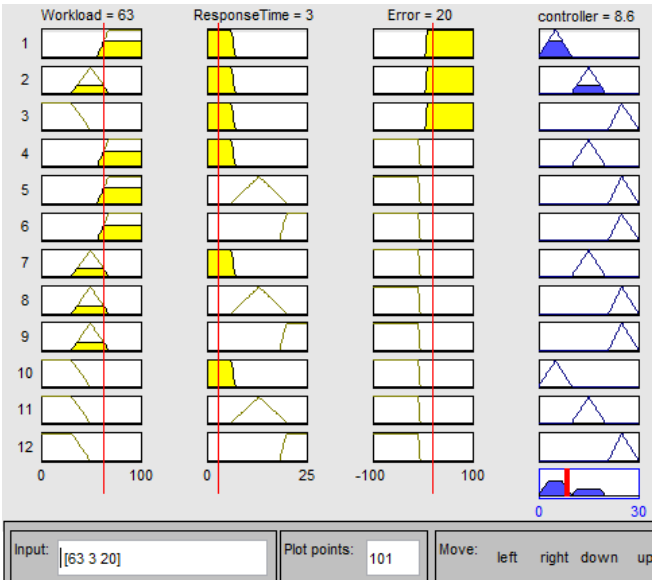


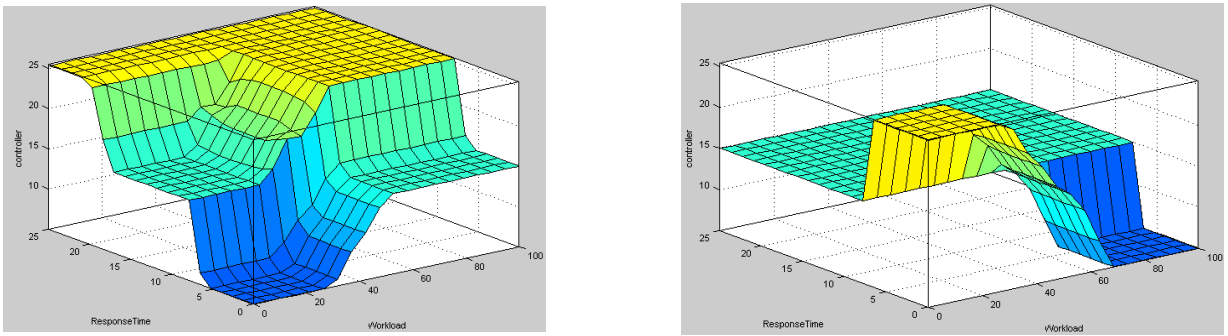
Figure 6: Fuzzy switching rule viewer

4 Experimental Evaluation

Here, we present an experimental evaluation of the suitability of the proposed methodology in comparison with a conventional integral controller. For this purpose we have extended the CloudSim Calheiros et al. (2011) framework to perform the necessary experimentation. Moreover, JFuzzyLogic Cingolani & Alcala-Fdez (2012) library is exploited to implement the switching logic. A self-written workload generator for producing various synthetic random workloads is also utilized. The following two criteria are considered for the evaluation:

- SLO violation: Service Level Objectives (SLO) are the measurable elements of a Service Level Agreement (SLA). An SLA is an agreement between the provider of the service and the consumer. An SLO violation will be considered, if a service request does not complete its execution with in defined response time.
- Running time: This is referred to as the total running time of all virtual machines till the completion of an experiment. Normally cost is considered for such evaluation. However, the reason of selecting running time in minutes is that, the experiments are performed on workloads that consist of few hours time span. Therefore, in such a situation, where partial hours are considered as full hours, the calculation of costs per hour basis cannot provide a good estimate for comparison.

In CloudSim, the requests/jobs of a workload are submitted with the length of the job that determines the service time of that job. We consider a fixed service time of 400 milliseconds for each job, where the desired response time is considered as 1.6 seconds. Thus an *SLO violation* will be considered, if a job takes more than 1.6 seconds. The *Running Time* of a virtual machine is the time since it is created until destroyed, either as a result of a scaling decision or when the execution



(a) When the control error is negative(scale up)

(b) When the control error is positive (scale down)

Figure 5: Action surface for controllers fuzzy switching mechanism

finishes completely. In all experiments, virtual machine boot up time are not considered and a 5-minute cool down period is applied between two scaling decisions. Two scenarios are evaluated here, one using synthetic workloads, whereas the other is based on real workload extracted from Wikipedia traces Wikibench (2009).

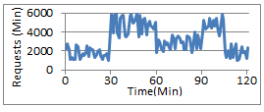
The various workloads used for evaluation contains arrival rate of up to 6000 job requests per minute, as can be seen in Figure 7. Each workload is categorized into three levels as per the percentage ranges defined for arrival rate in the switching logic section. The gain parameters of the controllers are obtained off-line using an experimental trial and error method by generating various synthetic random workloads based on a specific workload category, such as for *Lazy* gain, the workloads with low arrival rate are utilized. Different experiments are then performed using these random synthetic workloads with various gain values. The gain with best results i.e. with a low number of SLO violation and small running time are selected for final experimentation from each category. Whereas gain for conventional controller has obtained with workload from minimum possible requests per minute to maximum requests per minute. The summarized gain parameters used for final experimentation can be seen from Table 2.

Controller	Gain
Lazy	-0.08
Moderate	-0.1
High	-0.8
Conventional	-1.1

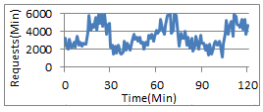
Table 2 Integral gains used for experiment

4.1 Synthetic

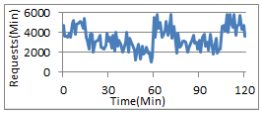
Three synthetic workloads are generated. Each workload is generated for two hours span with 8 chunks, whereas each chunk is from a different category that is selected randomly, except the 1st chunk. The 1st chunk contains arrival rates from low, moderate and high categories respectively as shown in Figure 7a – 7c. For each minute



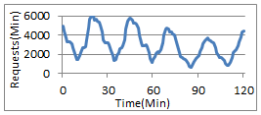
(a) Synthetic with low level start (WL_Low)



(b) Synthetic with middle level start (WL_Middle)

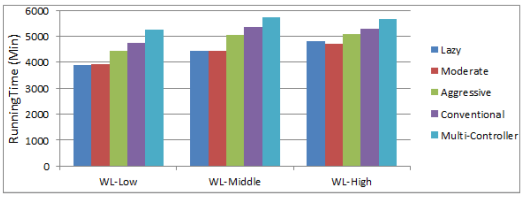


(c) Synthetic with high level start (WL_High)

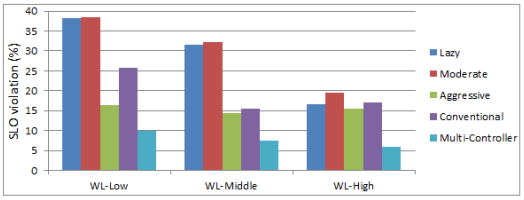


(d) Wikipedia workload (WL_Wiki)

Figure 7: Various workloads used for experimentation



(a) Aggregated results for running time



(b) Aggregated results for SLO violation (%)

Figure 8: Aggregated results for synthetic workloads

the number of requests is generated randomly based on the respective category range.

Figure 8 presents the aggregated results obtained. As can be seen, in comparison with single integral controllers, the multi-controller approach with respect to *Running Time* is comparatively expensive but in smaller magnitude. E.g. In case of *WL_High*, the *RunningTime* of multi-controller is higher (5678 vs 5279) than conventional. However, the multi-controller approach in terms of the performance parameter is much better, i.e. the percentile time of *SLO violation* occurred in experiment. E.g. in case of *WL_High*, The number of *SLO violation* is 5.92 % using our multi-controller approach, where for the same experiment; it is 16.79 % using the conventional approach. This shows that the multi-controller approach in comparison with other single controller based approaches has significantly higher potential to guarantee system stated performance.

4.2 Real Data: Wikipedia Trace Logs

The Wikipedia trace logs presented of five days recorded from 18th September, 2007 to 23rd September, 2007 presented in Figure 1b. This trace is scaled down to 2 hours horizontally and to 6000 requests per minute vertically, as shown in Figure 7d for the experimentation. The results of the experiment itself are shown in Figure 9. These results are approximately identical to the results described above in synthetic workload scenarios. The multi-controller approach in terms of Running Time is comparatively expensive but in smaller magnitude (5482 vs 4723) than the conventional approach, whereas in terms of SLO violation, it performs significantly better (6.06 % vs 10.14%).

The key objective of an elastic controller is to maintain better performance by minimizing the number of SLO violation close to zero. In both of the scenarios i.e. synthetic and real, the multi-controller approach although remains slightly expensive. However it performs significantly better in terms of maintaining system performance by reducing the number of SLO violation

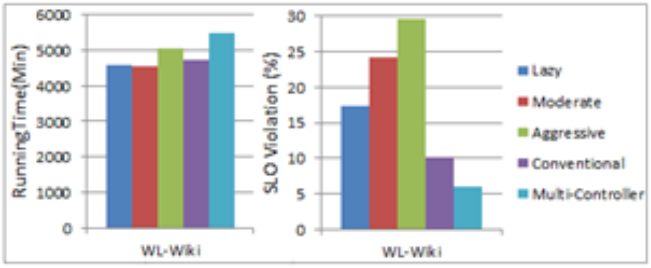


Figure 9: Aggregated results for Wikipedia trace

from an average of 19 % to 8% in synthetic case and similarly from 10% to 6% in the case of real workload example. The reason of being expensive is the design of the rules i.e. in scale down, if arrival rate is higher then prioritize performance otherwise saving cost as can be seen from Section 3.3. Thus depending on the situation, in comparison with single controller approach, the proposed approach adapt to suitable controller to achieve better performance.

Passino et al. (1998) suggests the consideration of two important factors i.e. minimizing the amount of memory used and computation time while implementing the fuzzy based control mechanism. Thus considering both of the issues, the switching mechanism is designed with minimum number of rules i.e. 12 in total. The small set of rules avoid memory usage by computing directly at each instant of time rather than from any persistent storage. Moreover, in order to avoid the computational overhead each input and output is defined using a combination of triangular and trapezoid functions only and with the use of just three membership functions. The triangular and trapezoid functions has the advantage of being simple and efficient in comparison with other Passino et al. (1998).

5 Related Work

Control theory is one of the available techniques employed for dynamic resource provisioning in cloud computing. There are various kinds of control theoretic approaches available in the literature. However, to the best of our knowledge, there is no prior work that exploits the use of multi-controller based approaches in combination with fuzzy logic to make informed decision based on current system behaviours. Some of the approaches exploited in similar scenarios are briefly summarized in this section.

A multi-model framework is proposed using MMST for QoS management in Patikirikorala et al. (2011). This framework is based on the use of multi-controllers; where each controller is designed for a different operation region. A particular controller is selected using model prediction error only. Moreover, details of how the different operating regions shall be realized are not available. Contrary to this approach, the proposed approach used not only control error but the current state of the system to select the most suitable controller. Moreover, the operating regions are realized using experts based categorization of workloads.

Ali-Eldin et al. (2013) proposed an analysis technique for workload classification at the data center level. This classification is then used to detect a suitable controller which results in better resource provisioning. This approach classifies one workload from another by determining the pattern of a workload. Based on the pattern, a workload is assigned to a specific controller that performs better for that kind of workload. Our approach differs in the context, where we classify the workload of an elastic application into various categories and then use fuzzy switching to select the most suitable controller in runtime.

A Proportional threshold controller by Lim et al. (2009) is based on a dynamic range of reference point rather than a fixed set point. A similar technique is also used for elastic storage Lim et al. (2010). Both of these approaches are however based on the use of a single controller. Other approaches include the use of adaptive techniques that dynamically adapt to changes at runtime. Such as adaptive hybrid elasticity controller by Ali-Eldin, Tordsson & Elmroth (2012), adaptive controller for bursty workloads Ali-Eldin, Kihl, Tordsson & Elmroth (2012) and a self-tuning controller for predictable eScience Park & Humphrey (2009). All these are based on the dynamic readjustment of the gain parameters. On the contrary, our approach provides an adaptive methodology using multi-controller and fixed gains that do not require any on-line estimation or learning technique.

There are various fuzzy based controllers used to implement elasticity in cloud Jamshidi et al. (2014), Xu et al. (2007), Wang et al. (2011). The fuzzy rule engine in such proposals drives the elastic system that determines how to scale. Such approaches in general are different from control theoretic approaches where the controller used satisfies a constraint or guarantees an invariant on the outputs of the system Ghanbari et al. (2011). However, this research work partially builds on the expert based categorization for workload and response times for the autonomic fuzzy based elastic controller Jamshidi et al. (2014).

6 Conclusion

We solve the problem of horizontal elasticity of cloud computing using a novel multi-controller approach with fuzzy switching. The introduced control methodology is used to dynamically alter the number of allocated virtual machines in order to keep the CPU utilization close to a desired level. We compare the introduced methodology with the conventional single controller based approach used for similar problems. Initial experimental results performed on various synthetic and real workloads demonstrate that the use of a multi-controller with intelligent switching mechanism achieves much better

performance than its conventional counterparts. Thus it enhances the capability of an elastic application to guarantee the system stated performance.

Ongoing and future work will continue to further address the main challenges associated with multiple switching control systems. This includes the following:

i SASO properties Analysis: SASO stands for Stability, Accuracy, Short settling, and Overshoot, which are the most important properties that have to be considered for a control system when evaluating. The analysis of these properties will help in the formal evaluation of the proposed control methodology in comparison with the state of the art approaches.

ii Chattering: One of the main issue for many switching control methodologies, e.g. sliding mode control, that results in high oscillation in the output of the system. It is consider more serious than transients encountered during the controllers' switching when the scheme is not 'bumpless'.

iii Performance: In future, the use of additional real case studies and a full range of cloud workload patterns will be explored for a more detailed comparative performance analysis and benchmarking against other state-of-the-art approaches.

Acknowledgement

The research work carried out in this paper is funded through a PhD scholarship program provided jointly by SICSA (http://www.sicsa.ac.uk) and the Division of Computer Science and Mathematics, University of Stirling. The work is also supported by Natural Science Foundation of China (under grants 71571076 and 71171087). A. Hussain was supported by the Royal Society of Edinburgh (RSE) and NNSFC joint project grant no. 61411130162, and the UK Engineering and Physical Science Research Council (EPSRC) grant no. EP/M026981/1. We also wish to thank the anonymous reviewers who helped improve the quality of the paper.

References

Abdelzaher, T., Diao, Y., Hellerstein, J. L., Lu, C. & Zhu, X. (2008), 'Introduction to control theory and its application to computing systems', *Performance Modeling and Engineering* pp. 185–215.

Abdelzaher, T. F., Shin, K. G. & Bhatti, N. (2002), 'Performance guarantees for web server end-systems: A control-theoretical approach', *Parallel and Distributed Systems, IEEE Transactions on* **13**(1), 80–96.

Abdullah, R., Hussain, A. & Polycarpou, M. M. (2007), Fuzzy logic based switching and tuning supervisor for a multi-variable multiple controller, *in* 'Fuzzy Systems Conference, 2007. FUZZ-IEEE 2007. IEEE International', IEEE, pp. 1–6.

Ali-Eldin, A., Kihl, M., Tordsson, J. & Elmroth, E. (2012), Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control, *in* 'Proceedings of the 3rd workshop on Scientific Cloud Computing Date', ACM, pp. 31–40.

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

Ali-Eldin, A., Tordsson, J., Elmroth, E. & Kihl, M. (2013), 'Workload classification for efficient auto-scaling of cloud resources', *Department of Computer Science, Umea University, Umea, Sweden, Tech.Rep* .

Bai, Y. & Wang, D. (2006), Fundamentals of fuzzy logic Control - fuzzy sets, fuzzy rules and defuzzifications, *in* Y. Bai, H. Zhuang & D. Wang, eds, 'Advanced Fuzzy Logic Technologies in Industrial Applications', Advances in Industrial Control, Springer London, pp. 17–36.
URL: *http://dx.doi.org/10.1007/978-1-84628-469-4_2*

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F. & Buyya, R. (2011), 'CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms', *Software: Practice and Experience* **41**(1), 23–50.

Cingolani, P. & Alcala-Fdez, J. (2012), jFuzzyLogic: a robust and flexible fuzzy-Logic inference system language implementation., *in* 'FUZZ-IEEE', Citeseer, pp. 1–8.

Farokhi, S., Jamshidi, P., Brandic, I. & Elmroth, E. (2015), Self-adaptation Challenges for Cloud-based Applications: A Control Theoretic Perspective.

Galante, G. & De Bona, L. C. E. (2012), A survey on cloud computing elasticity, *in* 'Proceedings - 2012 IEEE/ACM 5th International Conference on Utility and Cloud Computing, UCC 2012', pp. 263–270.

Gandhi, N., Tilbury, D. M., Diao, Y., Hellerstein, J. & Parekh, S. (2002), Mimo control of an apache web server: Modeling and controller design, *in* 'American Control Conference, 2002. Proceedings of the 2002', Vol. 6, IEEE, pp. 4922–4927.

Garside, J. (2013), 'Amazon's record $21bn Christmas sales push shares to new high'.
URL: *http://www.theguardian.com/technology/2013/jan/30/a*
christmas-record-sales-profits
http://tinyurl.com/qjo7g6v

Ghanbari, H., Simmons, B., Litoiu, M. & Iszlai, G. (2011), Exploring alternative approaches to implement an elasticity policy, *in* 'Cloud Computing (CLOUD), 2011 IEEE International Conference on', IEEE, pp. 716–723.

Hussain, A., Abdullah, R., Yang, E. & Gurney, K. (2012), An intelligent multiple-controller framework for the integrated control of autonomous vehicles, *in* 'Advances in Brain Inspired Cognitive Systems', Springer, pp. 92–101.

Jamshidi, P., Ahmad, A. & Pahl, C. (2014), Autonomic resource provisioning for cloud-based software, *in* 'Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems', ACM, pp. 95–104.

Karlsson, M., Karamanolis, C. & Zhu, X. (2005), 'Triage: Performance differentiation for storage systems using adaptive control', *ACM Transactions on Storage (TOS)* **1**(4), 457–480.

Kihl, M., Elmroth, E., Tordsson, J., Årzén, K.-E. & Robertsson, A. (2013), The challenge of cloud control, *in* '8th International Workshop on Feedback Computing'.

Kostantos, K., Kyriazis, D. & Themistocleous, M. (2015), 'Real-time event management in cloud environments', *International Journal of High Performance Computing and Networking* **8**(3), 212–221.

Lim, H. C., Babu, S. & Chase, J. S. (2010), Automated control for elastic storage, *in* 'Proceedings of the 7th international conference on Autonomic computing', ACM, pp. 1–10.

Lim, H. C., Babu, S., Chase, J. S. & Parekh, S. S. (2009), Automated control in cloud computing: challenges and opportunities, *in* 'Proceedings of the 1st workshop on Automated control for datacenters and clouds', ACM, pp. 13–18.

Lorido-Botran, T., Miguel-Alonso, J. & Lozano, J. A. (2014), 'A review of auto-scaling techniques for elastic applications in cloud environments', *Journal of Grid Computing* **12**(4), 559–592.

Lu, Z., Wu, J., Bao, J. & Hung, P. C. K. (2016), 'OCReM: OpenStack-based cloud datacentre resource monitoring and management scheme', *International Journal of High Performance Computing and Networking* **9**(1-2), 31–44.

Narendra, K. S., Driollet, O. A., Feiler, M. & George, K. (2003), 'Adaptive control using multiple models, switching and tuning', *International Journal of Adaptive Control and Signal Processing* **17**(2), 87–102.

Padala, P., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A. & Salem, K. (2007), Adaptive control of virtualized resources in utility computing environments, *in* 'ACM SIGOPS Operating Systems Review', Vol. 41, ACM, pp. 289–302.

Parekh, S., Gandhi, N., Hellerstein, J., Tilbury, D., Jayram, T. & Bigus, J. (2002), 'Using control theory to achieve service level objectives in performance management', *Real-Time Systems* **23**(1-2), 127–141.

Parekh, S., Rose, K., Diao, Y., Chang, V., Hellerstein, J., Lightstone, S. & Huras, M. (2004), Throttling utilities in the IBM DB2 universal database server, *in* 'American Control Conference, 2004. Proceedings of the 2004', Vol. 3, IEEE, pp. 1986–1991.

Park, S.-M. & Humphrey, M. (2009), Self-tuning virtual machines for predictable escience, *in* 'Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid', IEEE Computer Society, pp. 356–363.

Passino, K. M., Yurkovich, S. & Reinfrank, M. (1998), *Fuzzy control*, Vol. 42, Addison-wesley, Menlo Park, CA.

Patikirikorala, T. & Colman, A. (2010), Feedback controllers in the cloud, *in* 'Proceedings of APSEC'.

Patikirikorala, T., Colman, A., Han, J. & Wang, L. (2011), A multi-model framework to implement self-managing control systems for QoS management, *in* 'Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems', ACM, pp. 218–227.

Roy, N., Dubey, A. & Gokhale, A. (2011), Efficient autoscaling in the cloud using predictive models for workload forecasting, *in* 'Cloud Computing (CLOUD), 2011 IEEE International Conference on', IEEE, pp. 500–507.

Wang, L., Xu, J., Zhao, M., Tu, Y. & Fortes, J. A. B. (2011), Fuzzy modeling based resource management for virtualized database systems, *in* 'Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on', IEEE, pp. 32–42.

Wang, X., Cao, J. & Wang, J. (2016), 'A dynamic cloud service selection strategy using adaptive learning agents', *International Journal of High Performance Computing and Networking* **9**(1-2), 70–81.

Wikibench (2009), 'Wikipedia access traces'.
URL: *http://www.wikibench.eu/?page_id=60*

Xu, J., Zhao, M., Fortes, J., Carpenter, R. & Yousif, M. (2007), On the use of fuzzy modeling in virtualized data center management, *in* 'Autonomic Computing, 2007. ICAC'07. Fourth International Conference on', IEEE, p. 25.

Zadeh, L. A. (1996), 'Fuzzy logic= computing with words', *Fuzzy Systems, IEEE Transactions on* **4**(2), 103–111.

Zhu, Q. & Agrawal, G. (2010), Resource provisioning with budget constraints for adaptive applications in cloud environments, *in* 'Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing', ACM, pp. 304–307.

Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A., Padala, P. & Shin, K. (2009), 'What does control theory bring to systems research?', *ACM SIGOPS Operating Systems Review* **43**(1), 62–69.