Noname manuscript No.
(will be inserted by the editor)

Towards a biologically inspired soft switching approach for cloud resource provisioning

Amjad Ullah · Jingpeng Li · Amir Hussain · Erfu Yang

Received: date / Accepted: date

Abstract Cloud elasticity augments applications to dynamically adapt to changes in demand by acquiring or releasing computational resources on the fly. In the past, we developed a framework for cloud elasticity utilizing multiple feedback controllers simultaneously. Each controller determines the scaling action with different intensity, whereby the selection of a suitable controller is realized with a fuzzy inference system. In this paper, we aim to identify the similarities between cloud elasticity and action selection mechanism in animals. We treat each controller of our previous framework as an action and propose a novel bio-inspired, soft switching approach. This approach integrates basal ganglia computational model as an action selection mechanism. Initial experimental results are presented, which demonstrate that the basal ganglia based approach has higher potential to improve the overall system performance and stability.

A. Ullah
Divisions of Computer science and Maths, University of Stirling, UK
E-mail: aul@cs.stir.ac.uk

J. Li
Divisions of Computer science and Maths, University of Stirling, UK
E-mail: jli@cs.stir.ac.uk

A. Hussain
Divisions of Computer science and Maths, University of Stirling, UK
E-mail: ahu@cs.stir.ac.uk

E. Yang
Department of Design, Manufacture and Engineering Management, University of Strathclyde, UK
E-mail: erfu.yang@strath.ac.uk

Keywords Cloud horizontal elasticity · dynamic resource provisioning · fuzzy logic · Basal ganglia · soft switching · auto scaling · control theory

1 Introduction

The popularity of web applications such as social networking, wikis, news portals and e-commerce applications are posing new challenges to the management of underlying computational resources [1]. Such applications are subject to unpredictable workload conditions that vary from time to time. For example

i The higher workload on e-commerce website during festivals or promotional schemes than normal such as Amazon Christmas sale [2], recent China's 'singles day' sale [3] etc.
ii Facebook experienced a 10 time increase in their users within a span of three hours [4].
iii Web applications with diurnal pattern, where the workload arrival rate at day time is higher than night (e.g. Wikipedia trace [5]).

The performance of such applications is of utmost importance, as poor performance can result in the violation of Service Level Objectives (SLO). SLO violation has a direct consequence of losing customers and thus some business, e.g. every 100 ms of latency costs Amazon 1 percent in sales [6].

Cloud computing with attractive features of pay-as-you-go pricing model and elasticity, is a perfect match to host web applications that holds dynamically varying workloads. Cloud elasticity allows applications to dynamically adjust the underlying resources as closely as possible to the application demands, in response to changes observed in the environment such as workload

fluctuations. This enables cloud customers to pay only for the resources that are used [7]. The client has to provide an elastic policy that maintains the performance of a system at a desired level, as well as minimize the infrastructure running cost. However providing such an elastic policy that determines the right amount of cloud resources to meet system performance goals is a challenging task [8,9].

Control theory, therefore provides a systematic methodology to develop feedback controllers [10,11] to implement elasticity. Such methods are resilient to disturbances caused by workload and usually satisfy a constraint or guarantee to maintain the output of a system to a desired value [12]. An elastic feedback controller maintains the performance of systems close to a desired reference point by adjusting a manipulated variable, such as the number of running virtual machines [13]. The majority of existing proposals for elastic feedback controllers are designed with the use of one model that captures the system behaviour over an entire operating period. However, such approaches cannot perform well for systems that hold unpredictable workload conditions.

Considering the time-varying workload nature of cloud web applications, we have previously proposed an intelligent multi-controller based framework for cloud elasticity problems [14]. This framework distributes the system among three feedback controllers, where each controller can be designed for a particular operating region. The three controllers employed are named *Lazy, Moderate* and *Aggressive*. A switching mechanism is developed that determines the suitable controller at runtime. The results obtained using this method demonstrates higher potential in achieving system stated performance. However, such methods are subject to bumpy transitions that can lead systems to an unstable state [15,16].

Determining the optimal actions is an action selection problem and has been the focus of research in many fields [17,18]. There are evidences available which proves that the decision of 'what has to be done next' in animals is managed centrally using a switching mechanism in a brain nuclei called Basal Ganglia (BG) [19, 20]. Using this phenomenon, we aim to identify the opportunity to exploit the biological inspired approach of action selection for cloud elasticity. This enables us to treat the three controllers of our previous approach as actions thus enhancing our work to propose a bio-inspired soft-switching approach. This will allow the selection of right controllers in more natural biologically plausible method. Thus enhancing the possibility of smoother transitions that result in better system stability.

The contributions of this paper are comprised of.

i The formulation of cloud resource provisioning as an action selection problem to demonstrate the applicability of bio-inspired soft switching approach;
ii The integration of BG based computation model developed in [21,22];
iii The fuzzy logic based salience generation model;
iv The evaluation of proposed approach in comparison with existing elastic approaches using real workloads.

The rest of the paper is organized as follows. An overview of related work and relevant concepts are provided in Section 2 and 3 respectively. Section 4 introduces our previous approach, whereas Section 5 explains the proposed enhancements to the existing framework. Section 6 describes the experimentation and evaluation of the work, whereas Section 7 concludes the paper and briefly discusses the future work.

2 Related work

The existing literature on cloud elasticity is abundant. However, to the best of our knowledge, there is no such work that exploits bio-inspired action selection mechanism for cloud resource provisioning. Our motivation of this work comes from the use of bio-inspired approaches in complex systems for intelligent decision making in fields like autonomous vehicle systems and robotics [23, 18,16,24–28].

Focusing on elasticity literature, the resource provisioning proposals are versatile in nature as it highlights the use of different techniques such as control theoretic feedback controllers, threshold-based rules, machine learning, etc [13,29]. The use of threshold based rules is mostly common because of the commercially available solutions such as Amazon [30] and Rightscale [31]. Academic solutions are available as well e.g. [32, 33]. The appealing feature of rule based techniques is its simplistic nature. However, they require an in-depth knowledge of the underlying system to properly set-up the rules [13]. Secondly, they are unable to cope with sudden increase in workload [4].

Machine learning methods such as Reinforcement learning (RL) are also used to implement elasticity [6, 34,35]. However, such methods are often criticized for bad performance due to the long on-line training time and their inability to cope with sudden burst [13]. Other approaches include the use of elastic feedback controllers of various nature (e.g. fixed [11,36,37] or adaptive [10, 38]). Both kinds of approaches i.e. fixed and adaptive have their own merits and demerits. For example, fixed approaches are criticized for not suitable with dynamic

and unpredictable workload [39], while the adaptive controllers have been blamed for unable to cope with sudden burst in workload [13] and high computational cost because of on-line estimation [39]. The multi-model approach of [39, 40] is analogous to our approach. However, there are two main differences. Firstly, the selection of suitable controller is only based on prediction control error. Secondly, it is not clear that how the system must be partitioned among sub models. The approaches of [41–43] are different in the context, where each approach is applicable at the data centre level. Where, our approach advocates fine grained resource control over application level.

3 Action selection, Basal ganglia and Elastic controller

Action selection is referred to the process of selecting what to do next from a set of actions by an agent based on some knowledge of internal state and some provided sensory information of environmental context to best achieve its desired goal [44]. Over the period, researchers have learnt that in brain, the problem of action selection is handled through the use of a central switching mechanism [19, 20]. This mechanism is implemented by a group of subcortical nuclei collectively referred as Basal ganglia (BG).

Based on the functional anatomy of BG, the research work carried out in the past proposed functional models of BG [21, 22, 45, 46, 17, 47]. Focusing on computational model of [21, 22], competing actions are represented throughout the nervous system. The subsystems of brain send excitatory signals that represents the behavioural expression to the BG. The behavioural expression defines an action in BG and its strength is determined by the salience that represents the activity level of its neural representation. Where, outputs in BG, the actions are mediated through the release of inhibitory signals. Thus in every iteration, the functional model accepts a set of salience signals and produce a set of selected and unselected signals. The model can be run in one of three modes i.e. *Hard, Soft* or *Gate* mode. A maximum of one action can be selected in *Hard* mode, whereas multiple actions can be selected in *Soft* and *Gate* mode. However, in *Soft*, the selected actions are returned as an output, whereas in case of *Gate*, the model returns the proportion of each selected actions. For a detail functional anatomy of BG refers to [48].

The elasticity controller takes a scaling decision based on the current performance of system, available environmental information such as workload disturbances and internal state such as CPU utilization, memory consumption, etc. Analysing the description of elastic controllers and general definition of action selection problem, we can argue that an elastic controller is an autonomous agent and the problem of selecting the suitable controller by our previous approach can be mapped as an action selection problem. Therefore, we aim to integrate the BG computational model as an action selection mechanism. The problem can be defined as, how to select the right controllers, that results in an efficient readjustment of the underlying virtual machines as per the needs at that point of time.

4 Multi-controller based cloud resource provisioning

In [14], we proposed a multi-controller based approach to implement cloud elasticity. Considering the time-varying workload nature of cloud based web applications, this approach integrates multiple elastic feedback controllers simultaneously. Each controller can be designed specifically for different operating region. The existing research on the use of multiple controllers is still lacking of a standard approach that determines the partitioning of a system among sub controllers [49]. Therefore, this methodology uses the distribution of workload intensity into various categories such as low, medium and high by domain experts as a partitioning criterion to design multiple models. A switching methodology is developed that decides the suitable controller at runtime, based on current system behaviour. Figure 1 shows the architecture of this methodology whereas, the following subsections explains the various components of this framework.

4.1 Control policy

The three controllers employed as can be seen in Figure 1 are named *Lazy*, *Moderate* and *Aggressive*. They can be of any type. However, we have used the integral control law for each one of them because of its simplistic nature and the ability to remove the steady state errors [11]. Moreover, it has been also used for related problems [11, 36]. The average CPU utilization is used as a performance metric, whereas number of virtual machines is used as control input. This control methodology adjusts the number of virtual machines to keep the CPU utilization at a desired level. The integral control law can be defined as following:

$$u_{t+1} = u_t + K_i * (y_{ref} - y_t) \quad (1)$$

At each iteration, (u_{t+1}) represents the new number of virtual machines, where (u_t) represents the current

number of virtual machines. K_i is the integral gain parameter, which can be obtained off-line using standard procedure [15]. y_{ref} represents the desired CPU utilization where y_t is the measured CPU utilization obtained from system monitors.

4.2 System monitoring

Every cloud provider facilitates their customers with an Application Programming Interface (API) or monitoring service to get access to various system level performance metrics and log files, e.g. Cloudwatch by Amazon. The elastic scaling decision is dependent on these metrics as they represent the system behaviour at a particular time. Thus the system monitoring component of an elastic controller can makes use of system provided API to obtain up-to-date measurement of various performance metrics.

4.3 Switching mechanism

The switching mechanism selects suitable controller at each iteration based on the information obtained from *system monitoring* component. This mechanism is actually a Fuzzy Inference System (FIS), which is constructed using the following three standard steps: (1) specifying domain knowledge (2) defining membership function and (3) fuzzy rules. A brief description of each step is provided below.

- Domain knowledge: The knowledge base of the system consists of three parameters. These parameters are *Workload*, *ResponseTime* and *ControlError*. The *Workload* and *ResponseTime* are adapted from the work done in [4], where they are constructed using knowledge obtained from domain experts (i.e. architects and administrators). The *ControlError* represents the difference between desired and measured CPU utilization and can be represented as:

$$e_t = y_{ref} - y_t \tag{2}$$

The *ControlError* has been divided into three linguistic variables, i.e. *Positive*, *Normal* and *Negative*, which are obtained using trial and error method through experimentation. The *Positive* specifies that the measured CPU utilization is less than the desired, whereas the *Negative* represents that the measured CPU utilization is higher than the desired level. The *Normal* represents that either the error is 0 or within a margin of uncertainty due to noise or inaccuracy in the measurement. The full ranges of all three parameters can be seen from 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: This converts crisp input into corresponding fuzzy value. Introducing membership functions is the first step of fuzzification process [50], which defines the degree of crisp input against its linguistic variables in the range of 0 to 1. The FIS in our case contains three input and one output fuzzy variables and therefore, four membership functions in total one for each fuzzy variable. Figure 2 illustrates these membership functions.

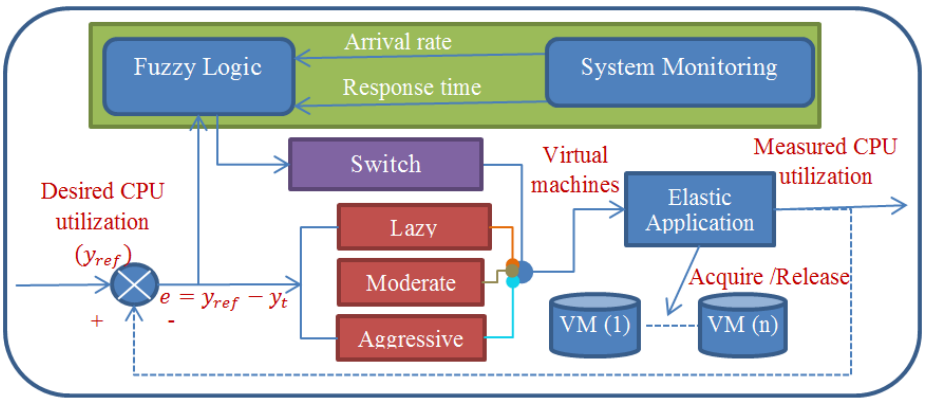


Fig. 1: Resource provisioning framework using multi-controller with fuzzy switching

– Fuzzy rules: The fuzzy rules describe the relationship between inputs and outputs of the FIS. *Workload (arrival rate)*, *Response time* and *Control error* are the inputs, whereas the output is *Controller*. Every elasticity decision consists of two ingredients i.e. the scaling actions and magnitude. The magnitude depends on selected controller, whereas the scaling actions can be determined by the value of *Control error*. There are three possible actions i.e. no scaling, scale up and scale down. A positive *Control error* means scale down, negative means scale up. Whereas normal means no scaling. Therefore, we have only rules where *ControlError* is either *Positive* or *Negative*. The following is one of the switching rules. In this case a scale down operation is performed using *Lazy* controller.

$$IF \quad \overbrace{arrivalRate\ IS\ high}^{\text{Possible values: high, middle or low}} \quad AND \quad \overbrace{responseTime\ IS\ instantaneous}^{\text{Possible values:instantaneous, medium or low}}$$

$$AND \quad \underbrace{error\ IS\ positive}_{\text{Possible values: Positive, Negative or Normal}} \quad THEN \quad \underbrace{controller\ IS\ lazy}_{\text{Possible values: Aggressive, Moderate or Lazy}}$$

Similarly, the following rule specifies a scale up operation using an *Aggressive* controller:

$$IF\ arrivalRate\ IS\ high\ AND\ responseTime\ IS\ slow$$

$$AND\ error\ IS\ negative\ THEN\ controller\ IS\ aggressive$$

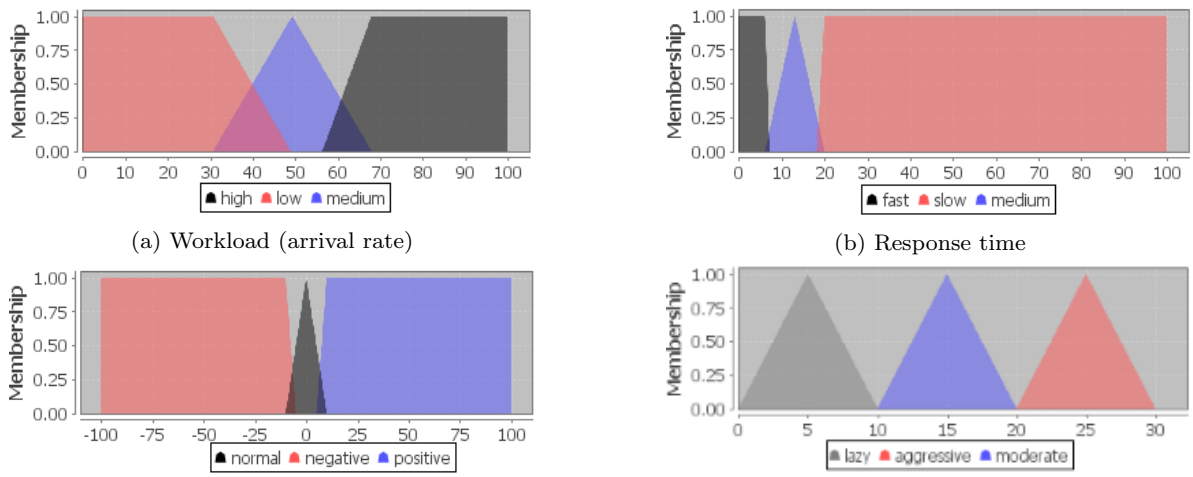
At each iteration, the overall process works as follows.

i The FIS obtain input values from *System Monitoring* component.
ii The input values are then fuzzified through the defined membership functions.
iii The FIS then evaluates the rules and identifies the output i.e. *Controller*.
iv The *Switch* component then only activates output of selected controller.
v The elastic application then add/remove virtual machines to/from the existing cluster based on the decision of selected controller.

5 Basal ganglia inspired cloud resource provisioning

The experimentation results obtained from our previous framework demonstrate that it has higher potential to improve system performance in comparison with typical single feedback controller approach of elasticity. However, the framework is based on hard switching mechanism, where the control methodology selects best controller at each iteration. Such a control methodology is subject to an undesirable phenomenon called bumpy transition occurred when switching between various operating regions. This phenomenon causes oscillation [15, 16] that leads system to an unstable state, where cloud resources can be acquire/release in a periodic way. The oscillation of resources can have deteriorating effects on system performance and running cost. It is, therefore desirable to improve the framework with possibility of smoother transition to avoid any oscillatory behaviour. Soft switching is an alternative approach used to avoid such undesired behaviour. In contrast to hard switching, the soft switching approach has the advantages of (1) avoidance the singularity and sensitivity problems, (2) improvement of robustness and stability aspects and (3) elimination of chattering issues [51].

Considering the advantages of soft switching approach, this research proposed a novel bio-inspired soft



(a) Workload (arrival rate)

(b) Response time

(c) Control error

(d) Controller

Fig. 2: Membership functions

switching approach for cloud resource provisioning problem. The new methodology integrates a BG based computational model [21,22] into our previous approach described in Section 4. The novelty of this work is at the system level as it combines various established methods including feedback controllers, fuzzy logic and BG based action selection mechanism in a novel way in order to exhibit their integrated effectiveness in a new problem domain. Whereas, the key aim of the BG integration is to demonstrate the effectiveness of bio-inspired action selection mechanism to underlying cloud resource provisioning problem. The BG based computational model has the advantages of biological plausibility and computational efficiency [23].

Our inspiration of exploiting BG based approach comes from research work carried out in the field of autonomous vehicle control (AVC) such as motion control of autonomous vehicle [23] and cognitive cruise control system [18]. In both approaches, the authors followed a modular approach by designing a set of controllers, where each controller can be optimized for a particular operating region or performance objective to achieve overall control objective by switching the suitable set of controllers at right time. Both of the approaches utilized the computational model of action selection proposed in [21,22].

Figure 3 presents the extended architecture of our previous work [14] presented in Figure 1. The extensions, as can be seen from figure, include (1) a modified version of the *Fuzzy Logic* component, (2) the integration of a new *Basal Ganglia* component and (3) derivation of the final output. Each of these extensions is further explained in the following sections.

5.1 Fuzzy logic

The integration of BG based computational model as an action selection mechanism requires salience signals as input. Thus, the first challenging issue that has to be dealt with is the generation of salience signals by making use of system internal state, various performance metrics and/or available sensory information [23].

In our previous work described in Section 4, we developed a FIS, which used as a switching mechanism. In this work, we extend the existing FIS to generate the salience signals required to provide as inputs to the BG based component. Thus, the switching mechanism of the previous work in its extended form becomes a fuzzy logic based salience generation model. The inputs to this model remain same, i.e. *Workload, ResponseTime* and *ControlError*, whereas the output is changed from one output (*Controller*) to three outputs. The outputs are salience strengths for each controller and can be read as *LazySalience, ModerateSalience* and *AggressiveSalience*. The following extension has been introduced to this part of the work:

- Membership function: As the inputs to model do not change, the corresponding membership functions remain the same as well. However, the output is changed. Therefore, the *Controller* membership function is replaced with three new functions, (i.e. one for each newly introduced output), which are the same and of basic triangular type as can be seen in Figure 5. All the membership functions used in our approach are either triangular or trapezoid because they have the advantage of being simple and efficient in comparison with others [52].
- Fuzzy Rules/Salience generation: The fuzzy rules are responsible to generate the salience signals that

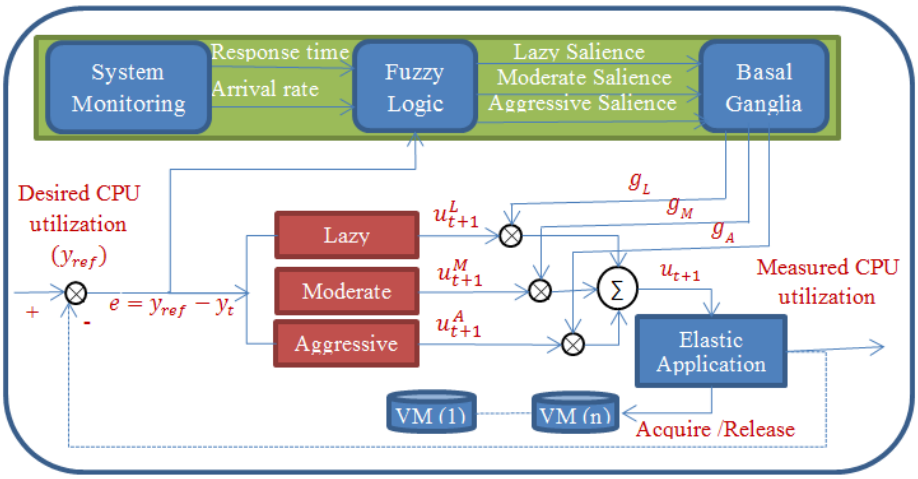


Fig. 3: Resource provisioning framework using BG based approach

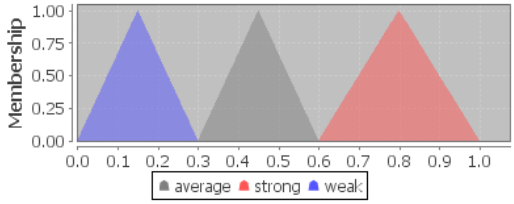


Fig. 5: Lazy/Moderate/Aggressive Salience

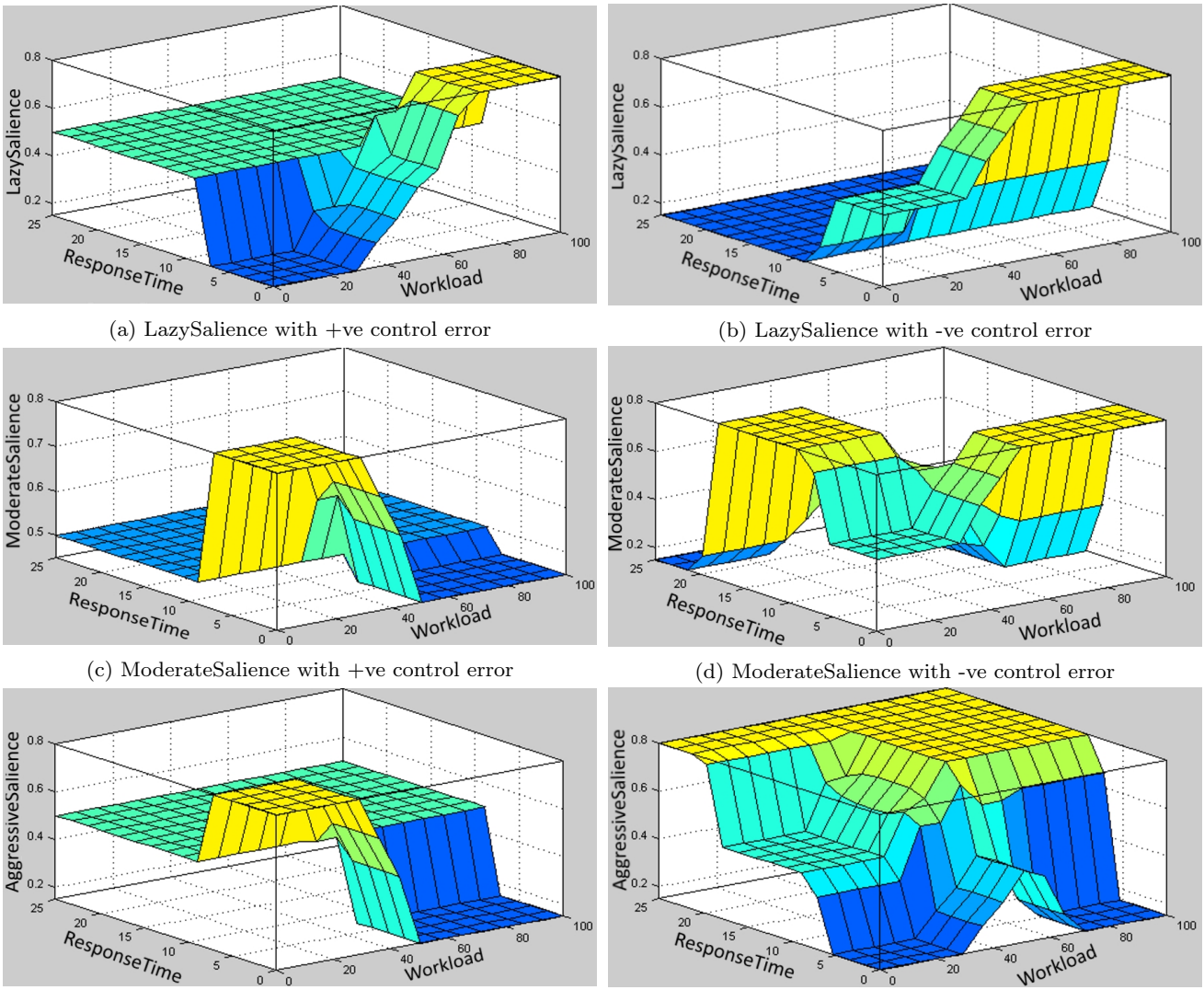
determine the strength of each controller. The fuzzy rules are now changed as previously every rule selects only one output, Whereas now, each rule has to determine the salience strength value for each controller. Thus the new rules look like the following,

IF arrivalRate IS high AND responseTime IS instantaneous AND error IS positive THEN (lazySalience IS strong), (moderateSalience IS average), (aggressiveSalience IS weak)

The possible value for each salience is *weak*, *average* and *strong*. There are 12 rules in total of the above format. The action surface of fuzzy salience generation model can be seen from Figure 4.

5.2 Basal Ganglia

The BG component integrates BG based computational model [21,22] of action selection described briefly in



(a) LazySalience with +ve control error

(b) LazySalience with -ve control error

(c) ModerateSalience with +ve control error

(d) ModerateSalience with -ve control error

(e) AggressiveSalience with +ve control error

(f) AggressiveSalience with -ve control error

Fig. 4: Action Surface

Section 3. The BG component accept three salience signals as inputs. These signals are named as *LazySalience, ModerateSalience* and *AggressiveSalience* respectively, which are obtained from the output of *Fuzzy logic* component as can be seen from Figure 3. These signals are then provided to BG based component, which produces gating signals that determines the proportion of each action.

5.3 Derivation of final output

The final output, i.e. u_{t+1} is calculated using the gating signals and the corresponding output of each controller as following:

$$u_{t+1} = \frac{(u^{L}_{t+1} * g_L) + (u^{M}_{t+1} * g_M) + (u^{A}_{t+1} * g_A)}{g} \quad (3)$$

The u_{t+1} represent the new final number of virtual machines, where u^{L}_{t+1}, u^{M}_{t+1} and u^{A}_{t+1} represents the output (new number of virtual machines) according to the individual controllers i.e. *Lazy, Moderate* and *Aggressive* respectively. The denominator g represents the number of gating signals, whose value is higher than zero as it is not always the case that more than one controller/action has to be selected at all time. This approach provides the calculation of final output in more naturally bio-inspired way, where it could provide the possibility to perform smoother transition between various switching decisions.

6 Experimentation and evaluation

6.1 Experimental set-up

We have extended CloudSim [53], a well-known simulator for cloud computing to implement a prototype of the proposed framework. JFuzzylogic [54] is also utilized to implement the fuzzy logic component. We have used two real workload traces to evaluate the performance of the proposed framework in comparison with existing approaches. Figure 6a represents the http requests made to 1998 world cup between (03/07/1998 08:01 to 04/07/1998 07:59). This data is obtained from [55]. Figure 6b represents the http requests made to Nasa website between (06/08/1995 00:01 to 07/08/1995 23:59) and is obtained from [56].

In CloudSim, we set-up a data centre in which the physical machines host virtual machines. The proposed framework manages a pool of virtual machines on behalf of web application. The CloudSim receives every http request of a workload as a job with a pre-defined length in a specific unit that determines the service time of that job. For this experimentation, we randomly assign service time to each job between (10 to 500 millisecond) based on the notion that some http requests are more time consuming than others such as mix read/write operations. The arrival time of each job is obtained from real time arrival of the http request in workload.

The various gain parameters of the controllers are obtained off-line using an experimental trial and error method. These are obtained by generating various synthetic random workloads based on a specific workload category, such as for *Lazy* gain where, 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 low number of SLO violation and small running time are selected from each category for the final experimentation. The gain parameters used for final experimentation can be seen from Table 2.

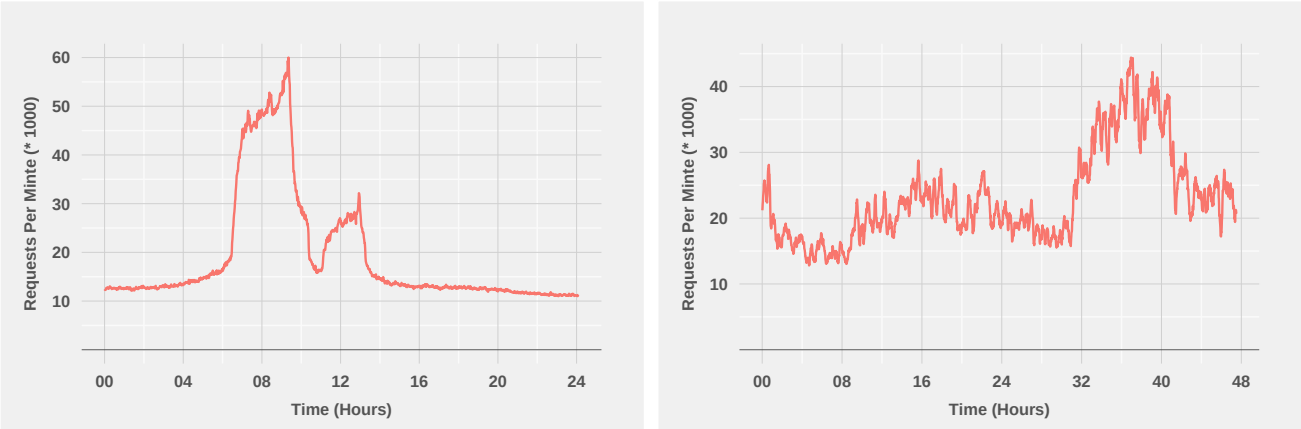
Controller	Gain
Lazy	-0.06
Moderate	-0.7
Aggressive	-1.1

Table 2: Integral gains used for experiments

6.2 Evaluation criteria

The evaluation of the proposed methodology is carried out in comparison with related cloud resource provisioning techniques. This include the conventional single model based feedback controllers, our previously proposed multi-controller based approach and Rightscale [31]. Rightscale is a well-known commercial elasticity mechanism developed using threshold based rules technique. Note that, we have not compared our selection of BG based computational model [21,22] as an action selection mechanism with other related approaches. This is because our aim is not to compare the performance of various action selection mechanisms but to demonstrate the effectiveness of a bio-inspired method in comparison with other state of the art cloud resource provisioning techniques. The evaluation criteria are comprised of the following:

- SLO Violation: SLO stands for Service Level Objectives, which is a measurable unit of Service Level Agreement (SLA). SLA defines an agreement between the provider and consumer of service. An SLO violation in our case is referred to the phenomenon, where a job request cannot complete its execution



(a) Worldcup workload trace

(b) Nasa workload trace

Fig. 6: Workloads used for experimentation

with in a desired response time (1 second for experimentation). The SLO violations can be treated as performance objective, where it is expected that each job must complete its execution within 1 second. This can be achieved, if the system maintains an average CPU utilization of 55 %. The relation i.e. maintaining average CPU utilization of 55 % can result in achieving response time less than 1 second, is obtained through off-line standard system identification experiments.

– Cost: The total running time of all virtual machines is recorded throughout the experiment. It includes the time when any virtual machine starts to the time it finishes execution either as a result of scale down operation or when the experiment finishes. The total time is calculated in minutes and partial hours are not considered as full hours. Moreover, an immediate start/stop of the virtual machine is considered to avoid any complexity in the implementation as well as to have a precise comparison of virtual machine running time because the experiments run for short time. The total running time of all virtual machines is then converted to hours for final calculation of hours. A rate of 0.013 $ per hour is applied to calculate the final cost based on the "t2.micro" machine pricing model of Amazon [57].

Apart from the above mentioned criteria, we also compare the results of average CPU utilization over the entire period of experiment for our previous work and BG based approach. In this regard, we recorded the measured CPU utilization for entire experiment, where each measurement represents the average CPU utilization of all virtual machine in last minute. These results shed light on stability perspective of the system with respect to BG usage.

6.3 Results

Figure 7 presents the aggregated results for both the experiments i.e. using Nasa and Worldcup workload traces. The *Lazy, Moderate* and *Aggressive* represent the typical single controller approaches, where each controller is designed to perform better in their respective regions, i.e. when the workload is low, medium or high. The *RS* represents Rightscale, whereas *MC* represents our previous approach described in Section 4 and *BG* represents the proposed work in this paper.

Considering the *Nasa* workload example, it can be seen from Figure 7b that overall, all approaches perform well in terms of performance except *Aggressive* approach. If we compare the percentile results of SLO violation, the *MC* approach has same number of SLO violation as *RS* (i.e. 0.21 %), where the *BG* has comparatively less number of SLO violation than all other approaches (i.e. 0.05 %). In terms of cost, there is not much difference in all approaches except *RS*. This means that *RS* has achieved better performance in this case but at a higher cost. In case of *Worldcup* workload example, it can be seen from Figure 7d that only *MC* and *BG* approach perform well in terms of achieving better performance with less number of SLO violations (i.e. 0.56 % and 0.29 % respectively). Moreover, they have achieved better performance at less cost than all other approaches.

The key objective of any elasticity mechanism is to improve the performance of underlying system by reducing the number of SLO violation to zero at a lowest cost possible. In both of the experiments, our proposed approaches (i.e. *MC* and *BG*) perform better in performance as well as in cost. However, other approaches like *RS* also showed good result in terms of performance

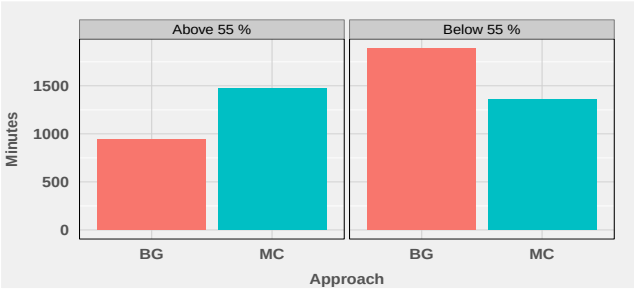
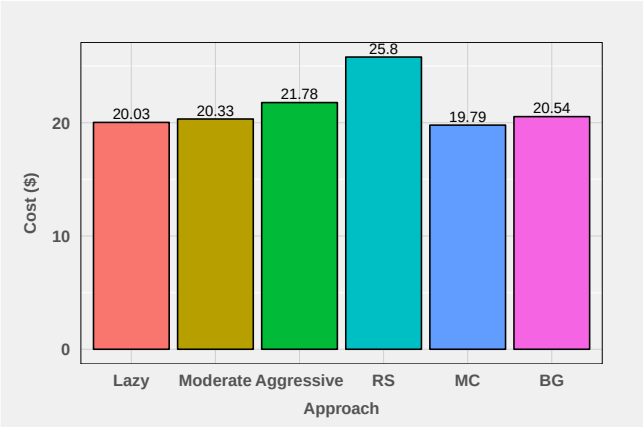


Fig. 8: Aggregated result of CPU Utilization highlighting, how many minutes an approach stayed below/above than reference point (55 %)

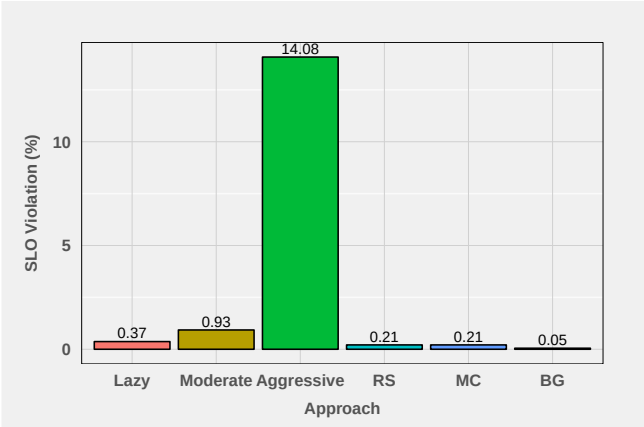
in the first case, but at a higher cost. Moreover, *Nasa* workload is comparatively less dynamic than *Worldcup* in terms of jumps in varying workload region. Comparing the results of *MC* and *BG*, we can observe that *BG* shows higher potential to achieve better performance with a bit higher but almost negligible cost than *MC*.

The above results demonstrate that adapting BG based action selection mechanism improves the overall results. However, another key aspect of adapting BG based approach is its ability of selecting the actions in a natural bio-inspired way, where it can improve the possibility of smoother transition between switching decision. In current experimentation, we do not provide a comprehensive quantitative measurements about how BG based approach improves the stability perspective of underlying application. However, the results in Figure 8 and 9 demonstrate some differences between *MC* and *BG* approaches with respect to the average CPU utilization recorded over the entire period of *Nasa* workload experiment that characterize the stability of system.

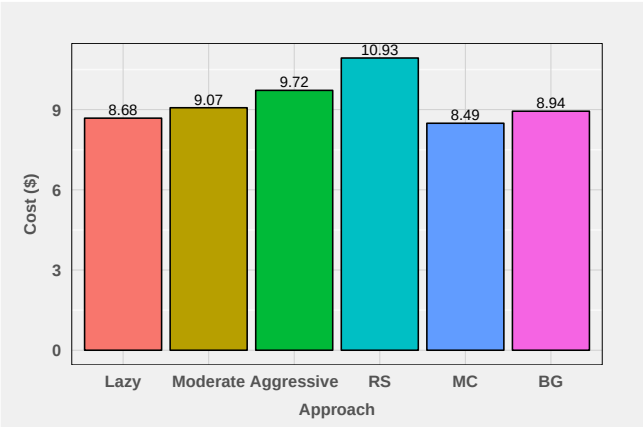
Note that the key objective of the control methodology is to maintain the CPU utilization close to the desired/reference point i.e. 55 % but under this range. The CPU utilization above the reference point mean that the performance of the system degrades. Figure 8 aggregates the count of the minutes for both approaches,



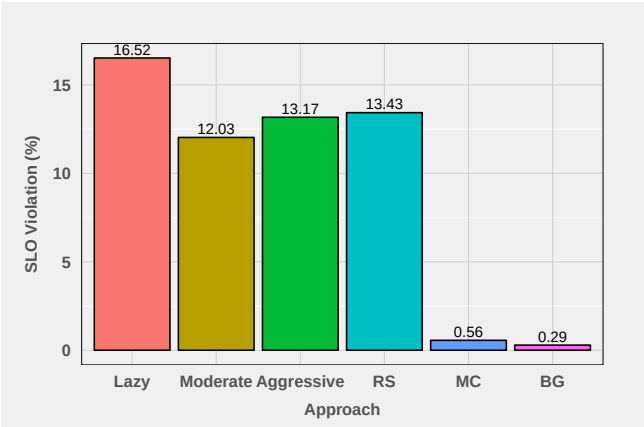
(a) Cost (Nasa)



(b) SLO (Nasa)



(c) Cost (Worldcup)



(d) SLO (Worldcup)

Fig. 7: Aggregated results of the experiments

when the cpu utilization is below and above the reference point. As can be seen from Figure 8, the *BG* approach maintained the CPU utilization below 55 %, maximum number of minutes (i.e.1892 to be exact) in comparison with *MC*, which is (1354 minutes). Where, the exact measurements for both the approaches when the CPU utilization is more than 55 % are 1476 and 938 for *MC* and *BG* respectively. This demonstrates that overall the *BG* approach maintains the CPU utilization closed under the reference point.

We further divide the measured CPU utilization for each approach into 12 hours, which is presented in Figure 9. This helps to visually demonstrate the difference between both of the approaches with respect to measured CPU utilization against the reference point. The 1st and 3rd rows belong to *MC* approach, whereas 2nd and 4th rows are from *BG* approach. The reference CPU utilization is represented with dark solid horizontal line in all graphs. The following points are observed with respect to the differences between both approaches.

- The overall average CPU utilization for the *BG* based approach is recorded as 52.58 %, whereas for the *MC* approach is 56 %. They can be seen in red colour dashed line in their respective graphs. Moreover *BG* reduces the likelihood of leading the system into overloaded status as some of the occurrences can be found in the case of *MC* approach, e.g. the sessions 08th to 12th hour, 20th to 24th hour etc.
- The CPU utilization in *BG* case never reaches to 70 % in the entire period of the experiment except at the start, which is same for both cases. Whereas, in case of *MC*, it has been crossed a number of times.

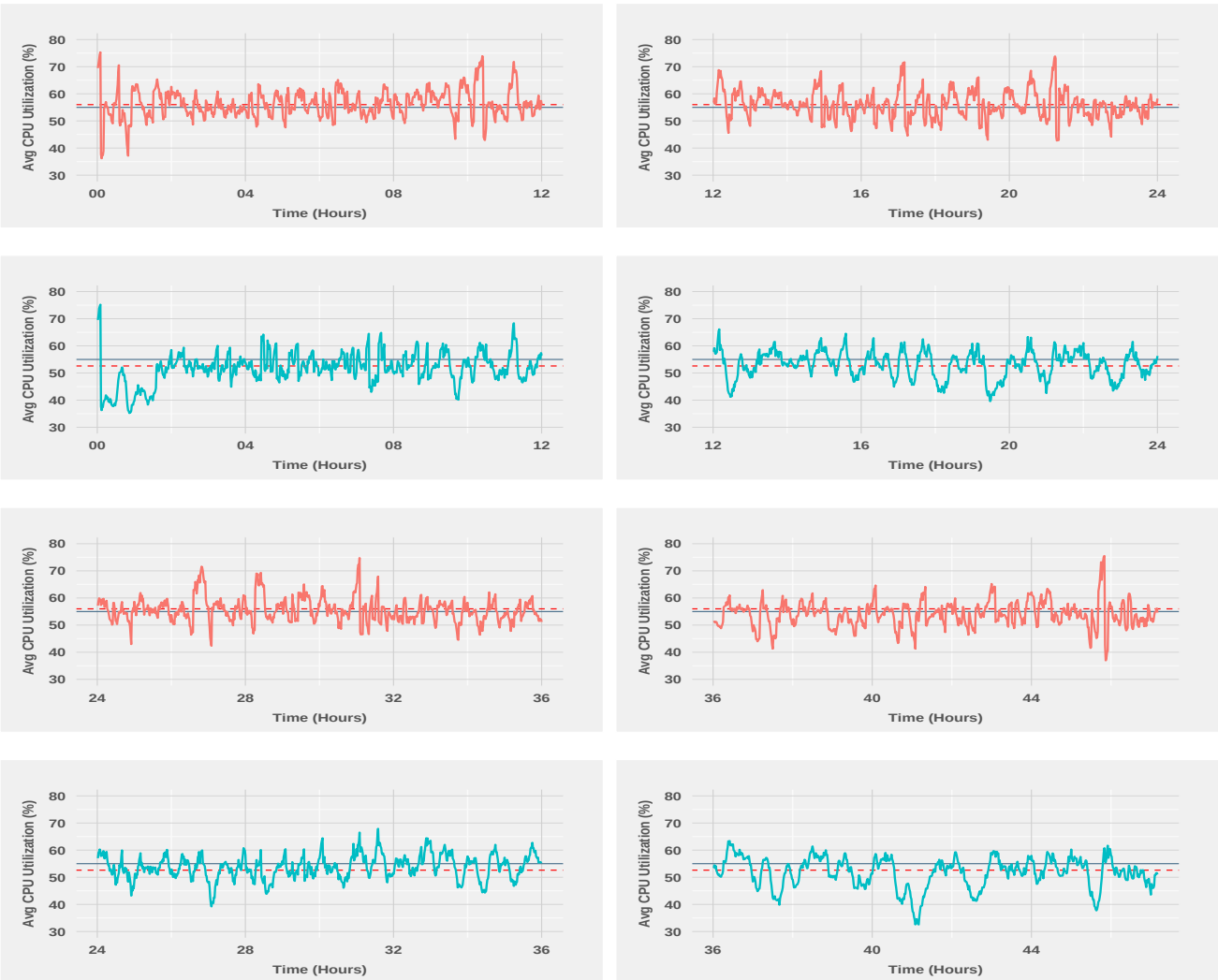


Fig. 9: Avg CPU Utilization of Nasa experiment with 12 hours period in each graph. 1st and 3rd rows belong to *MC*, while 2nd and 4th rows belongs to *BG*

- The CPU utilization in *BG* case almost remains lower than 65 % except only four times. Whereas in the case of *MC*, there were quite a few times, where it remained more than 65 % for some time such as the peaks in the 08th to 12th hour, 24th to 28th hour and 28th to 32th hour.
- Overall, the CPU utilization in case of *MC* has more abrupt transitions and peaks in comparison to *BG* approach, which can cause the oscillatory behaviour.

In light of the above discussion, we can argue that *BG* approach has the potential to reduce the likelihood of SLO violation by maintaining a desired CPU utilization, thus resulting in better system performance. Moreover, compared to the MC approach it shows smoother transition between switching decision that can reduce and/or avoid unwanted system oscillatory behaviour and will improve stability. Note that the work reported here is part of the preliminary study and therefore, we have not carried out theoretical stability analysis. However, an intuitive explanation is that, the mixing of all controllers is done (in Equation (3)) in a bio-inspired way augmented by the BG process that facilitate natural selection of actions that results in less 'bumping' at switching time [58]. Moreover, the computational model of [21,22] in particular is proved successfully to avoid oscillation and energy efficient in various action selection problems [17]. We aim, in future, to use the enhanced version of the BG model developed in [17], for which the formal stability proof can be establish using contraction theory of dynamical systems.

7 Conclusion and future work

We addressed the problem of cloud resource provisioning as an action selection problem. We proposed a biologically inspired soft switching approach to implement horizontal cloud elasticity. The proposed approach integrate a functional model of basal ganglia (BG), which augments the methodology to select the right set of controllers in a natural biologically plausible way. Thus reducing the likelihood of oscillation and increases the stability of underlying system. Moreover, a fuzzy inference system is introduced to generate the salience signals required to provide as inputs to BG model. We evaluated the proposed methodology by comparing with existing elasticity methods using CloudSim and two real workloads. The initial experimental results demonstrate that biological inspired method performs better in both evaluation aspects (i.e. performance and cost) than other approaches. Moreover, it also reduces the oscillation peaks in the measured CPU utilization observed in our previously proposed approach, thus having the potential to increase the stability of underlying system.

The work is still in its early stage, where we showed the suitability of biological inspired method of action selection in the context of cloud computing. The future work will address the key challenging issues related to the developed framework, which include the following: (1) A detailed theoretical convergence and stability analysis to formally evaluate the proposed methodology against other state of the art approaches, (2) Enhancement of fuzzy part using genetic algorithm to obtain optimal settings of fuzzy variable ranges, membership functions and fuzzy rules, (3) On-line learning capabilities of switching rules, and (4) The possibility to enhance the capability of framework by incorporating the vertical elasticity will be explored.

Compliance With Ethical Standards

Funding: 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 division of computer science and mathematics department, University of Stirling.

Ethical approval: This article does not contain any studies with human participants or animals performed by any authors.

References

1. A Al-Shishtawy and V Vlassov. ElastMan: Autonomic elasticity manager for cloud-based key-value stores. In *22nd ACM International Symposium on High-Performance Parallel and Distributed Computing, HPDC 2013*, pages 115–116, 2013.
2. Juliette Garside. Amazon's record $21bn Christmas sales push shares to new high, January 2013.
3. Theguardian. China's Alibaba records 'singles day' sales of $8bn in 10 hours, 2015.
4. Pooyan Jamshidi, Aakash Ahmad, and Claus Pahl. Autonomic resource provisioning for cloud-based software. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 95–104. ACM, 2014.
5. Guido Urdaneta, Guillaume Pierre, and Maarten van Steen. Wikipedia workload analysis for decentralized hosting. *Computer Networks*, 53:1830–1845, 2009.
6. Jinzhao Liu, Yaoxue Zhang, Yuezhi Zhou, Di Zhang, and Hao Liu. Aggressive Resource Provisioning for Ensuring QoS in Virtualized Environments. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2014.
7. Nikolas Roman Herbst, Samuel Kounev, and Ralf Reussner. Elasticity in Cloud Computing : What It Is , and What It Is Not. *Presented as part of the 10th International Conference on Autonomic Computing*, pages 23–27, 2013.
8. Lizhe Wang, Albert Y Zomaya, Dimitrios Georgakopoulos, Xian-He Sun, Guojun Wang, and Others. Recent

advances in autonomic provisioning of big data applications on clouds. *Cloud Computing, IEEE Transactions on*, 3(2):101–104, 2015.
9. Sukhpal Singh and Inderveer Chana. QoS-Aware Autonomic Resource Management in Cloud Computing: A Systematic Review. *ACM Computing Surveys (CSUR)*, 48(3):42, 2015.
10. Ahmed Ali-Eldin, Johan Tordsson, and Erik Elmroth. An adaptive hybrid elasticity controller for cloud infrastructures. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 204–212. IEEE, 2012.
11. Harold C Lim, Shivnath Babu, Jeffrey S Chase, and Sujay S Parekh. Automated control in cloud computing: challenges and opportunities. In *Proceedings of the 1st workshop on Automated control for datacenters and clouds*, pages 13–18. ACM, 2009.
12. Hamoun Ghanbari, Bradley Simmons, Marin Litoiu, and Gabriel Iszlai. Exploring alternative approaches to implement an elasticity policy. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 716–723. IEEE, 2011.
13. Tania Lorido-Botran, Jose Miguel-Alonso, and Jose A Lozano. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, 12(4):559–592, 2014.
14. Amjad Ullah, Jingpeng Li, and Amir Hussain. Towards Workload-Aware Cloud Resource Provisioning Using a Novel Multi-Controller Fuzzy Switching Approach. *International Journal of High Performance Computing and Networking*, 2015.
15. Joseph L Hellerstein, Yixin Diao, Sujay Parekh, and Dawn M Tilbury. *Feedback control of computing systems*. John Wiley & Sons, 2004.
16. Rudwan Abdullah, Amir Hussain, Kevin Warwick, and Ali Zayed. Autonomous intelligent cruise control using a novel multiple-controller framework incorporating fuzzy-logic-based switching and tuning. *Neurocomputing*, 71(13):2727–2741, 2008.
17. Benoît Girard, Nicolas Tabareau, Quang-Cuong Pham, Alain Berthoz, and J-J Slotine. Where neuroscience and dynamic system theory meet autonomous robotics: a contracting basal ganglia model for action selection. *Neural Networks*, 21(4):628–641, 2008.
18. Erfu Yang, Amir Hussain, and Kevin Gurney. A Brain-inspired Soft Switching Approach: Towards A Cognitive Cruise Control System. In *WIT Transactions on Engineering Sciences*, 2014.
19. P. Redgrave, T. J. Prescott, and K. Gurney. The basal ganglia: A vertebrate solution to the selection problem? *Neuroscience*, 89(4):1009–1023, 1999.
20. T. J. Prescott, P. Redgrave, and K. Gurney. Layered Control Architectures in Robots and Vertebrates. *Adaptive Behavior*, 7(1):99–127, 1999.
21. K Gurney, T J Prescott, and P Redgrave. A computational model of action selection in the basal ganglia. I. A new functional anatomy. *Biological cybernetics*, 84(6):401–410, 2001.
22. K Gurney, T J Prescott, and P Redgrave. A computational model of action selection in the basal ganglia. II. Analysis and simulation of behaviour. *Biological cybernetics*, 84(6):411–423, 2001.
23. Erfu Yang, Amir Hussain, and Kevin Gurney. A Basal Ganglia Inspired Soft Switching Approach to the Motion Control of a Car-Like Autonomous Vehicle. In *Advances in Brain Inspired Cognitive Systems*, volume 7888, pages 245–254. 2013.
24. Micha Czubenko, Zdzisaw Kowalczuk, and Andrew Ordys. Autonomous Driver Based on an Intelligent System of Decision-Making. *Cognitive Computation*, pages 1–13, 2015.
25. Tony J Prescott, Fernando M Montes González, Kevin Gurney, Mark D Humphries, and Peter Redgrave. A robot model of the basal ganglia: behavior and intrinsic processing. *Neural Networks*, 19(1):31–61, 2006.
26. José-Antonio Cervantes, Luis-Felipe Rodríguez, Sonia López, Félix Ramos, and Francisco Robles. Autonomous Agents and Ethical Decision-Making. *Cognitive Computation*, pages 1–19.
27. Benoît Girard, Vincent Cuzin, Agnès Guillot, Kevin N Gurney, and Tony J Prescott. A basal ganglia inspired model of action selection evaluated in a robotic survival task. *Journal of integrative neuroscience*, 2(2):179–200, 2003.
28. Philipp Rohlfshagen and Joanna J Bryson. Flexible latching: A biologically-inspired mechanism for improving the management of homeostatic goals. *Cognitive Computation*, 2(3):230–241, 2010.
29. Emanuel Ferreira Coutinho, Flávio Rubens de Carvalho Sousa, Paulo Antonio Leal Rego, Danielo Gonçalves Gomes, and José Neuman de Souza. Elasticity in cloud computing: a survey. *annals of telecommunications-annales des t{é}l{é}communications*, pages 1–21, 2015.
30. Amazong. Amazong auto scaling, 2015.
31. Rightscale. Set up Autoscaling using Alert Escalations, 2015.
32. Emiliano Casalicchio and Luca Silvestri. Autonomic Management of Cloud-Based Systems: The Service Provider Perspective. In *Computer and Information Sciences III*, pages 39–47. Springer, 2013.
33. Masum Z. Hasan, Edgar Magana, Alexander Clemm, Lew Tucker, and Sree Lakshmi D Gudreddi. Integrated and autonomic cloud resource scaling. *Proceedings of the 2012 IEEE Network Operations and Management Symposium, NOMS 2012*, pages 1327–1334, 2012.
34. Enda Barrett, Enda Howley, and Jim Duggan. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, 25(12):1656–1674, 2013.
35. Fouad Bahrpeyma, Ali Zakerolhoseini, and Hassan Haghighi. Using IDS fitted Q to develop a real-time adaptive controller for dynamic resource provisioning in Cloud's virtualized environment. *Applied Soft Computing*, 26:285–298, 2015.
36. Harold C Lim, Shivnath Babu, and Jeffrey S Chase. Automated control for elastic storage. In *Proceedings of the 7th international conference on Autonomic computing*, pages 1–10. ACM, 2010.
37. Ahmad Al-Shishtawy and Vladimir Vlassov. ElastMan: Elasticity Manager for Elastic Key-Value Stores in the Cloud. *Cloud and Autonomic Computing Conference (CAC '13)*, page 1, 2013.
38. Ahmed Ali-Eldin, Maria Kihl, Johan Tordsson, and Erik Elmroth. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In *Proceedings of the 3rd workshop on Scientific Cloud Computing Date*, pages 31–40. ACM, 2012.
39. Tharindu Patikirikorala, Alan Colman, Jun Han, and Liuping Wang. 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*, pages 218–227. ACM, 2011.

40. Tharindu Patikirikorala, Liuping Wang, Alan Colman, and Jun Han. HammersteinWiener nonlinear model based predictive control for relative QoS performance and resource management of software systems. *Control Engineering Practice*, 20(1):49–61, 2012.
41. Ahmed Ali-Eldin, Johan Tordsson, Erik Elmroth, and Maria Kihl. Workload classification for efficient autoscaling of cloud resources. *Department of Computer Science, Umea University, Umea, Sweden, Tech.Rep*, 2013.
42. Dan Xu, Xin Liu, and Athanasios V. Vasilakos. Traffic-Aware Resource Provisioning for Distributed Clouds. *IEEE Cloud Computing*, 2(1):30–39, 2015.
43. Qi Zhang, Student Member, and Mohamed Faten Zhani. Dynamic Heterogeneity-Aware Resource Provisioning in the Cloud. *Cloud Computing, IEEE Transactions on*, 2(1):14–28, 2014.
44. M. Tony Prescott. Action selection, 2008.
45. Jérémy Fix, Nicolas Rougier, and Frédéric Alexandre. A dynamic neural field approach to the covert and overt deployment of spatial attention. *Cognitive Computation*, 3(1):279–293, 2011.
46. Kevin N Gurney. Reverse engineering the vertebrate brain: methodological principles for a biologically grounded programme of cognitive modelling. *Cognitive Computation*, 1(1):29–41, 2009.
47. Alekhya Mandali, Maithreye Rengaswamy, V Srinivasa Chakravarthy, and Ahmed A Moustafa. A spiking Basal Ganglia model of synchrony, exploration and decision making. *Frontiers in Neuroscience*, 9:191, 2015.
48. Peter Redgrave. Basal ganglia, 2007.
49. Amir Hussain, Rudwan Abdullah, Erfu Yang, and Kevin Gurney. An intelligent multiple-controller framework for the integrated control of autonomous vehicles. In *Advances in Brain Inspired Cognitive Systems*, pages 92–101. Springer, 2012.
50. Ying Bai and Dali Wang. Fundamentals of Fuzzy Logic Control â Fuzzy Sets, Fuzzy Rules and Defuzzifications. In Ying Bai, Hanqi Zhuang, and Dali Wang, editors, *Advanced Fuzzy Logic Technologies in Industrial Applications*, Advances in Industrial Control, pages 17–36. Springer London, January 2006.
51. Sergey E Lyshevski. *Control systems theory with engineering applications*. Springer Science & Business Media, 2012.
52. Kevin M Passino, Stephen Yurkovich, and Michael Reinfrank. *Fuzzy control*, volume 42. Citeseer, 1998.
53. Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César A F De Rose, and Rajkumar Buyya. 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, 2011.
54. Pablo Cingolani and Jesus Alcala-Fdez. jFuzzyLogic: a robust and flexible Fuzzy-Logic inference system language implementation. In *FUZZ-IEEE*, pages 1–8. Citeseer, 2012.
55. Internet traffic Archive. Worldcup 1998 Web trace, 2015.
56. Network traffic Archive. Nasa-HTTP, 2015.
57. Amazon. Amazon EC2 pricing, 2015.
58. Erfu Yang, Amir Hussain, and Kevin Gurney. Neurobiologically-inspired soft switching control of autonomous vehicles. In *Advances in Brain Inspired Cognitive Systems*, pages 82–91. Springer, 2012.