Cogn Comput (2016) 8:992–1005
DOI 10.1007/s12559-016-9391-y

Towards a Biologically Inspired Soft Switching Approach for Cloud Resource Provisioning

Amjad Ullah[1] · Jingpeng Li[1] · Amir Hussain[1] · Erfu Yang[2]

Received: 23 November 2015 / Accepted: 19 February 2016 / Published online: 7 March 2016

Abstract Cloud elasticity augments applications to dynamically adapt to changes in demand by acquiring or releasing computational resources on the fly. Recently, we developed a framework for cloud elasticity utilizing multiple feedback controllers simultaneously, wherein, each controller determines the scaling action with different intensity, and the selection of an appropriate 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 the animal brain. We treat each controller in our previous framework as an action, and propose a novel bioinspired, soft switching approach. The proposed methodology integrates a basal ganglia computational model as an action selection mechanism. Initial experimental results demonstrate the improved potential of the basal ganglia-based approach by enhancing the overall system performance and stability.

Keywords Cloud elasticity · Dynamic resource provisioning · Fuzzy logic · Basal ganglia · Soft switching · Auto-scaling · Elastic feedback controller

✉ Amjad Ullah
aul@cs.stir.ac.uk

Jingpeng Li
jli@cs.stir.ac.uk

Amir Hussain
ahu@cs.stir.ac.uk

Erfu Yang
erfu.yang@strath.ac.uk

[1] Division of Computer Science and Mathematics, University of Stirling, Stirling, UK

[2] Department of Design, Manufacture and Engineering Management, University of Strathclyde, Glasgow, UK

Introduction

The popularity of web applications such as social networking, wikis, news portals and e-commerce applications is 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] and recent China's singles day' sale [3].
ii A 10-time increase that Facebook experienced 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 % 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 hold dynamically varying workloads. Cloud elasticity allows applications to dynamically adjust the underlying resources as closely as possible to the application demands, in response to the 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 the 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 was developed to determine the suitable controller at runtime. The results obtained using this method demonstrate a 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 on many fields [17, 18]. There are evidences available which prove that the decision of "what has to be done next" in animal's brain 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 a biologically inspired approach of action selection for cloud elasticity. This enables us to treat the three controllers in our previous approach as actions thus enhancing our work to propose a bioinspired soft switching approach. The selection of right controllers in more biologically plausible method will increase the possibility of smoother transitions that result in better system stability.

The contributions of this paper comprise the following:

1. Formulation of cloud resource provisioning as an action selection problem to demonstrate the applicability of bioinspired soft switching approach;
2. Integration of the BG-based computation model developed in [21, 22];
3. Fuzzy logic-based salience generation model;
4. Evaluation of the proposed approach in comparison with some existing elastic approaches using real workloads.

The rest of the paper is organized as follows. Following an overview of related work and relevant concepts, we introduce our previous approach and new basal ganglia-inspired cloud resource provisioning methodology. This is followed by the description of prototypical implementation, comparative simulation results and finally, some concluding remarks and future work are provided.

Related Work

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

Focusing on elasticity literature, the resource provisioning proposal is versatile in nature as it highlights the use of different techniques such as control theoretical feedback controllers, threshold-based rules and machine learning [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 are also used to implement elasticity [6, 34, 35]. However, such methods are often criticized for bad performance due to long online 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 the fixed and adaptive approaches have their own merits and drawbacks. For example, the fixed approaches are criticized for unsuitable 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 online estimation [39]. The multi-model approach in [39, 40] is analogous to our approach, but with the following two main differences: firstly, their selection of suitable controller is only based on the prediction of control

error; secondly, it is not clear how the system can be partitioned into submodels. The approaches from [41–43] are different in the context, where each of the approaches is applicable at the data centre level, while our approach advocates fine-grained resource control over the application level.

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 animal's brain, the problem of action selection is handled through the use of a central switching mechanism [19, 20], which is implemented by a group of subcortical nuclei collectively referred as basal ganglia (BG).

Based on the functional anatomy of BG, various functional models of BG have been proposed [17, 21, 22, 45–47]. Focusing on the computational model [21, 22], competing actions are represented throughout the nervous system. The brain subsystems send excitatory signals that represent the behavioural expressions to the BG. Each behavioural expression defines an action in BG, and its strength is determined by the salience that represents the activity level of its neural representation. These actions are mediated through the release of inhibitory signals. Thus in every iteration, the functional model accepts a set of salience signals and produces 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* modes. However, in *Soft* mode, the selected actions are returned as an output, whereas in the case of *Gate*, the model returns the proportion of each selected action. For a detailed functional anatomy of BG refer to [48].

The elasticity controller takes a scaling decision based on the current system performance, the available environmental information such as workload disturbances and internal state such as CPU utilization, and memory consumption. Analysing the description of elastic controllers and the 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 controller, which results in an efficient readjustment of the underlying virtual machines as per the needs at that point of time.

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 the cloud-based web applications, this approach integrates multiple elastic feedback controllers simultaneously. Each controller can be designed specifically for different operating region. Existing research on the use of multiple controllers still lacks a standard approach that determines the partitioning of a system among subcontrollers [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 to decide the suitable controller at runtime, based on current system behaviour. Figure 1 shows the architecture of this framework, whereas the following subsections explain the various components of the framework.

Control Policy

The three controllers employed as can be seen in Fig. 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 some similar problems [11, 36]. The average CPU utilization is used as a performance metric, whereas the 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 follows:

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

At each iteration, u_{t+1} represents the new number of virtual machines, while u_t denotes the current number of virtual machines. K_i is the integral gain parameter, which can be obtained offline using a standard procedure [15]. y_{ref} represents the desired CPU utilization, and y_t is the measured CPU utilization obtained from system monitors.

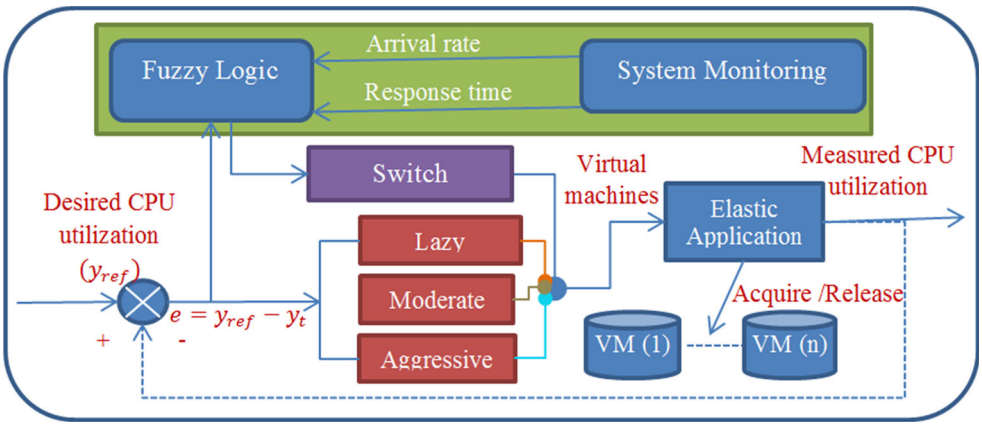

Fig. 1 Resource provisioning framework using multi-controller with fuzzy switching

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 make use of system provided API to obtain up-to-date measurement of various performance metrics.

Switching Mechanism

The switching mechanism selects a 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 functions 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: *Workload*, *ResponseTime* and *ControlError*. The *Workload* and *ResponseTime* are adapted from the work done in [4], where they are constructed using the knowledge obtained from domain experts (i.e. architects and administrators). The *ControlError* represents the difference between the desired and measured CPU utilization which is represented as:

$$e_t = y_{ref} - y_t \quad (2)$$

The *ControlError* has been divided into three linguistic variables (i.e. *Positive*, *Normal* and *Negative*) which are obtained using the 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.

– 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 [0,1]. The FIS in our case contains three inputs and one output fuzzy variables and therefore, four membership functions in total. Figure 2 illustrates these membership functions.

– Fuzzy rules: The fuzzy rules describe the relationship between the 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 the 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

Table 1 Ranges for fuzzy variables

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

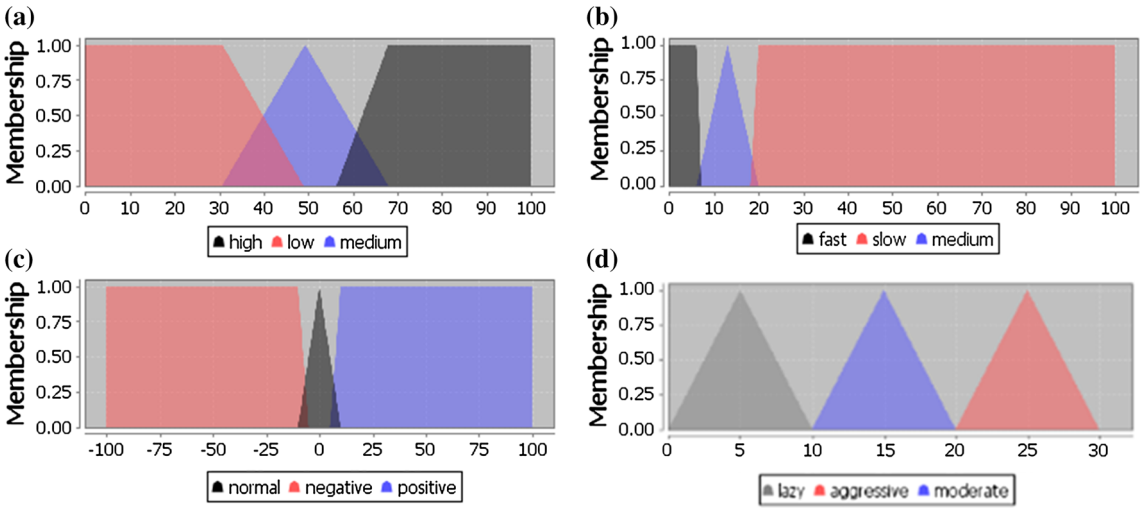

Fig. 2 Membership functions. **a** Workload (arrival rate). **b** Response time. **c** Control error. **d** Controller

down. A positive *Control error* means scale down, negative means scale up, and 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.

4. The *Switch* component then only activates the output of selected controller.
5. The elastic application then adds/removes virtual machines to/from the existing cluster based on the decision of the selected controller.

Possible values : high, middle or low — Possible values : instantaneous, medium or low

IF arrivalRate IS high AND responseTime IS instantaneous

AND error IS positive THEN controller IS lazy

Possible values : Positive, Negative or Normal — 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.

1. The FIS obtains input values from the *system monitoring* component.
2. The input values are then fuzzified through the defined membership functions.
3. The FIS then evaluates the rules and identifies the output, i.e. *Controller*.

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 a typical single feedback controller approach of elasticity. However,

the framework is based on the hard switching mechanism, where the control methodology selects the best controller at each iteration. Such a control methodology is subject to an undesirable phenomenon called bumpy transition occurred when the switching among various operating regions. This phenomenon causes oscillation [15, 16] that leads the system to an unstable state, where cloud resources can be acquired/released in a periodic way. The oscillation of resources may have deteriorating effects on system performance and running cost. It is therefore desirable to improve the framework with the 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) avoiding 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 bioinspired soft switching approach for cloud resource provisioning problem. The new approach integrates a BG-based computational model [21, 22] into our previous approach described in "Multi-controller-Based Cloud Resource Provisioning" section. 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 the bioinspired action selection mechanism to the underlying cloud resource provisioning problem. The BG-based computational model has the advantages of both biological plausibility and computational efficiency [23].

Our inspiration of exploiting BG-based approach comes from the 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 the 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 Fig. 1. The extensions, as can be seen from figure, include (1) a modified version of the *fuzzy logic* component, (2) an integration of the new *basal ganglia* component and (3) a derivation of the final output. Each of these extensions is further explained in the following sections.

Fuzzy Logic

The integration of BG-based computational model as an action selection mechanism requires salience signals as inputs. 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 earlier in this paper, 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

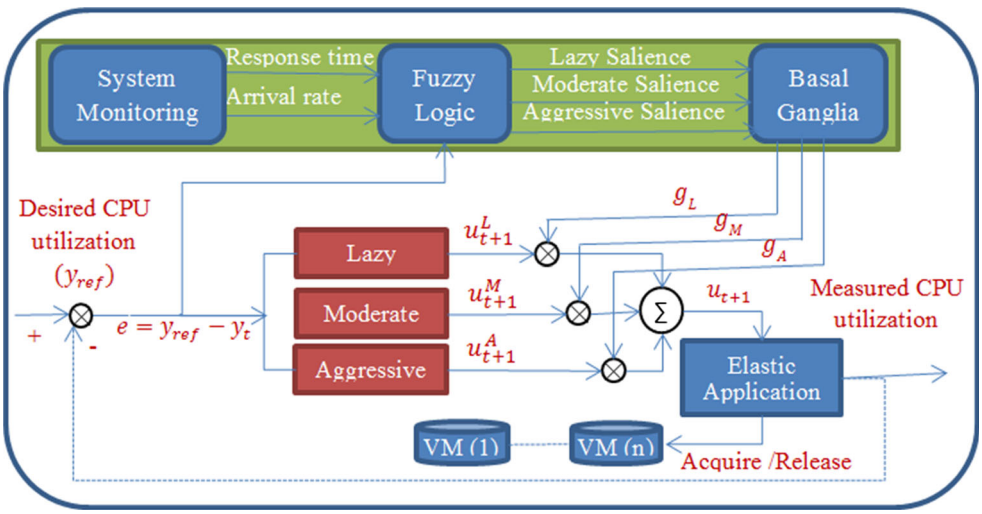

Fig. 3 Resource provisioning framework using BG-based approach

the 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 Fig. 4. 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 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 in the above format. The action surface of fuzzy salience generation model can be seen from Fig. 5.

Basal Ganglia

The BG component integrates the BG-based computational model [21, 22] of action selection described briefly earlier in this paper. The BG component accepts three salience signals (i.e. *LazySalience, ModerateSalience* and *AggressiveSalience*) as the inputs, which are obtained from the output of *fuzzy logic* component as can be seen from Fig. 3. These signals are then provided to the BG-based component to produce gating signals that determine the proportion of each action.

Derivation of the Final Output

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

$$u_{t+1} = \frac{\left(u^{L}_{t+1} * g_{L}\right) + \left(u^{M}_{t+1} * g_{M}\right) + \left(u^{A}_{t+1} * g_{A}\right)}{g} \quad (3)$$

The u_{t+1} represents 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. Similarly, the g_L, g_M and g_A are the gating signals that represents the proportion of each controller, i.e. *Lazy, Moderate* and *Aggressive*, respecively, where the denominator g represents the count of gating signals, when their value is higher than zero as it is not always the case that more than one controller/action has to be selected at every time. This approach provides the calculation of the final output in a more naturally bioinspired way, where it could provide the possibility to perform a smoother transition between various switching decisions.

Experimentation and Evaluation

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 the existing approaches. Figure 6a represents the http requests made to 1998 world cup between 03/07/1998 08:01 and 04/07/1998 07:59. These data are obtained from [55]. Figure 6b represents the http requests made to NASA website between 06/08/1995 00:01 and 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 and 500 ms based on the notion that some http requests are more time-consuming than others such as mixed read/write operations. The arrival time of each job is obtained from real-time arrival of the http request in workload.

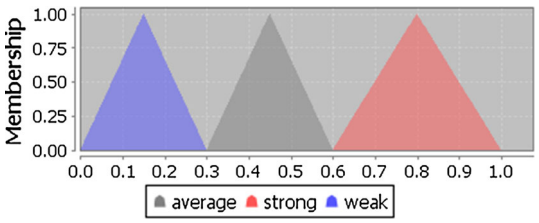

Fig. 4 Lazy/Moderate/Aggressive Salience

The various gain parameters of the controllers are obtained offline 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 the low number

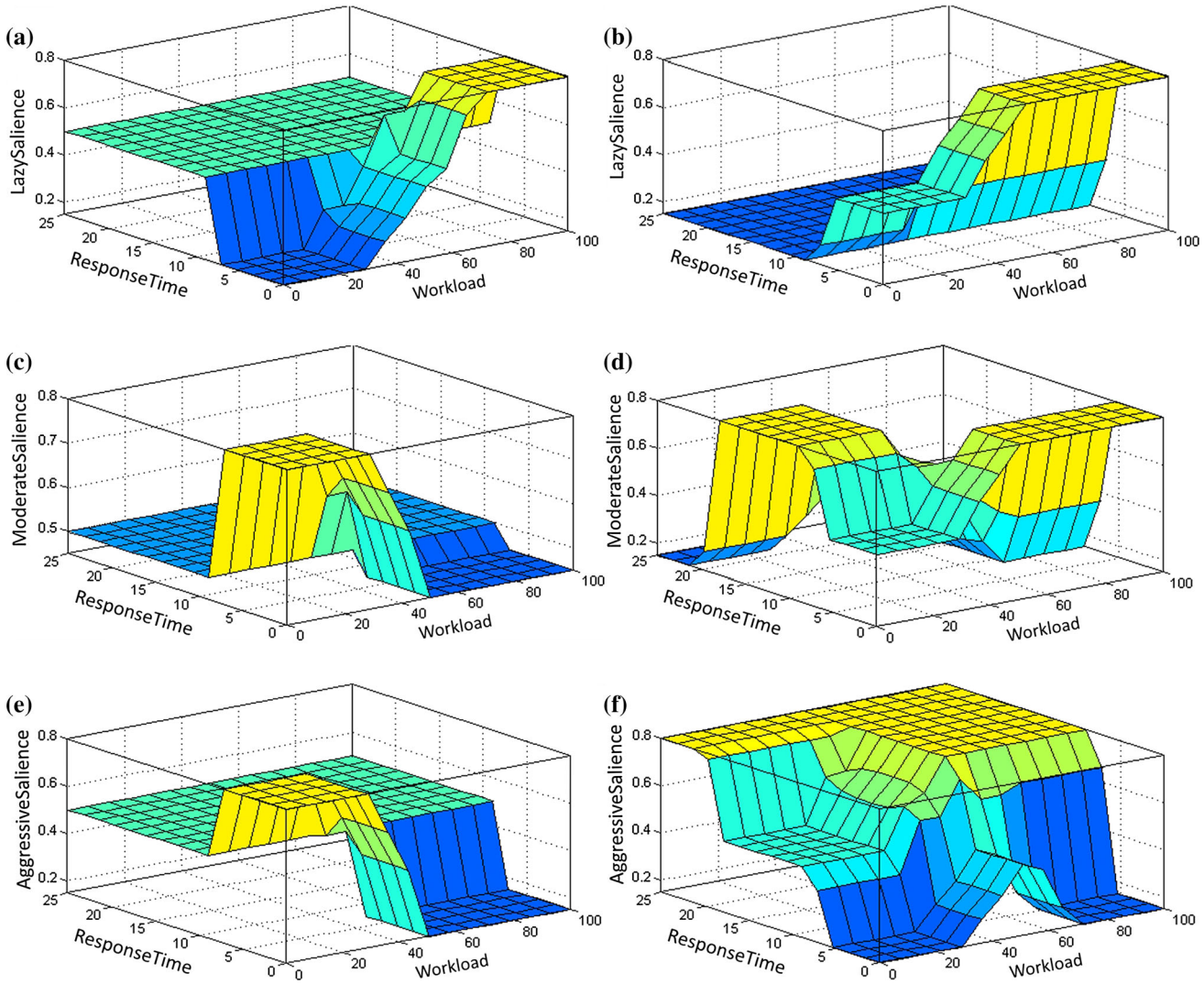

Fig. 5 Action surface. **a** LazySalience with positive control error. **b** LazySalience with negative control error. **c** ModerateSalience with positive control error. **d** ModerateSalience with negative control error. **e** AggressiveSalience with positive control error. **f** AggressiveSalience with negative control error

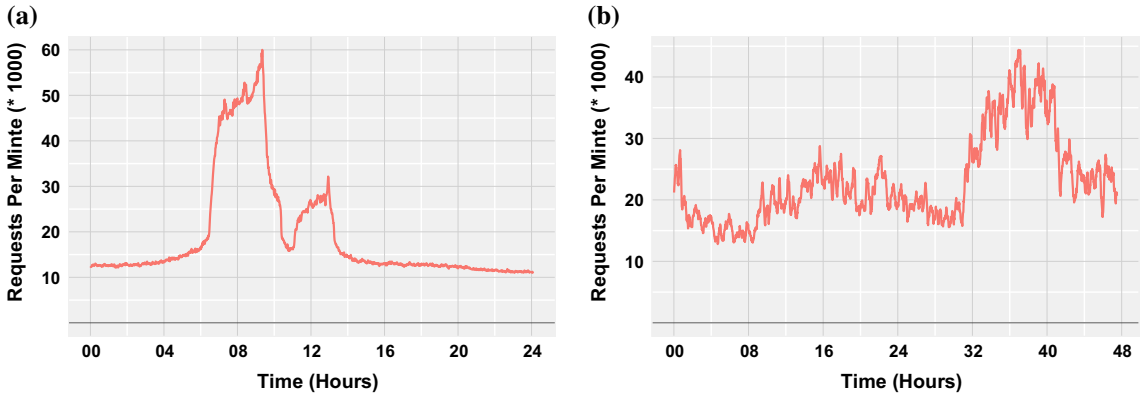

Fig. 6 Workloads used for experimentation. **a** Worldcup workload trace. **b** NASA workload trace

Table 2 Integral gains used for experiments

Controller	Gain
Lazy	−0.06
Moderate	−0.7
Aggressive	−1.1

of SLO violation and small running time, is selected from each category for the final experimentation. The gain parameters used for the final experimentation can be seen from Table 2.

Evaluation Criteria

The evaluation of the proposed methodology is carried out in comparison with the related cloud resource provisioning techniques. This includes 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 the 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 bioinspired 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 a service. An SLO violation in our case is referred to the phenomenon, where a job request cannot complete its execution within a desired response time (1 s for experimentation). The SLO violations can be treated as performance objective, where it is expected that each job must complete its execution within 1 s. This can be achieved, if the system maintains an average CPU utilization of 55 %. The relation between 55 % average CPU utilization and 1-s response time is obtained through offline 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 the average CPU utilization over the entire period of experiment for our previous work and the BG-based approach. In this regard, we record the measured CPU utilization for the entire experiment, where each measurement represents the average CPU utilization of all virtual machines in the last minute. These results shed light on the stability perspective of the system with respect to the BG usage.

Results

Figure 7 presents the aggregated results for both the experiments, i.e. using the NASA and Worldcup workload traces. The *Lazy, Moderate* and *Aggressive* represent the

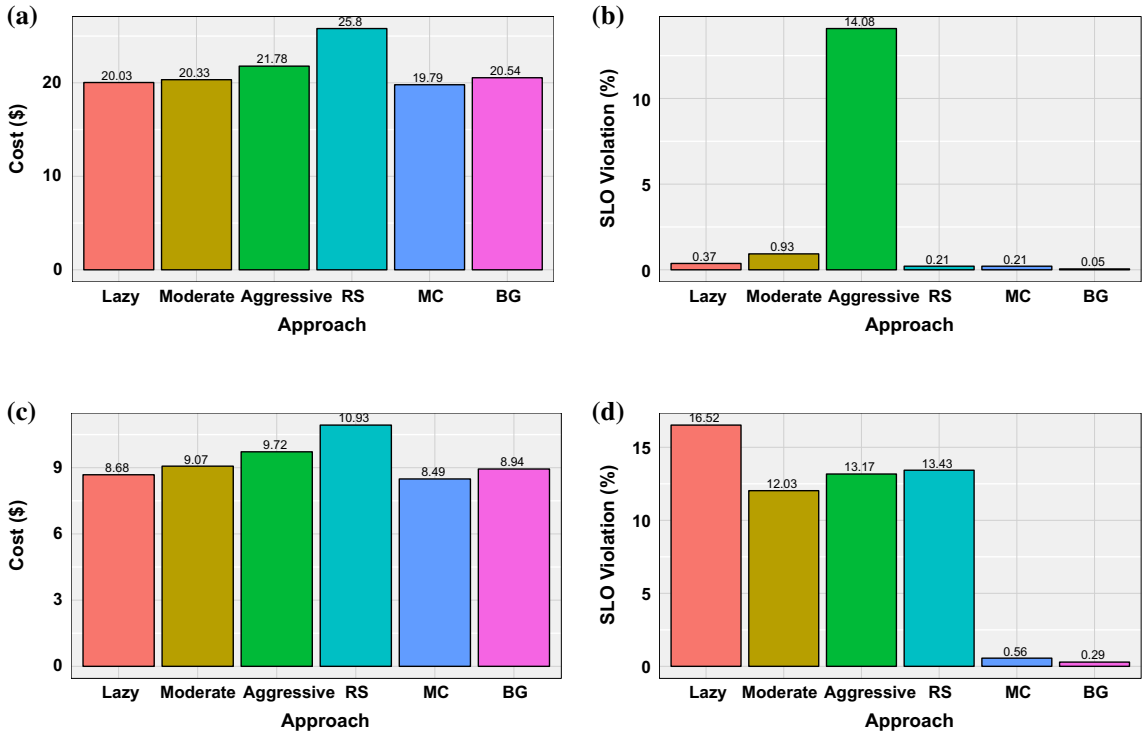

Fig. 7 Aggregated results of the experiments. **a** Cost (NASA). **b** SLO (NASA). **c** Cost (Worldcup). **d** SLO (Worldcup)

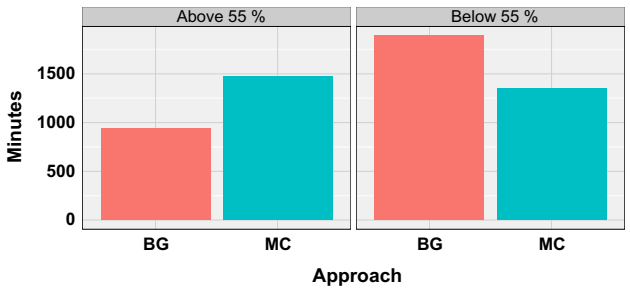

Fig. 8 Aggregated result of the CPU Utilization highlighting the minutes an approach stays below/above the reference point (55 %)

typical single controller approaches, where each controller is designed to perform better in their respective regions when the workload is low, medium and high, respectively. The *RS* represents Rightscale, *MC* represents our previous approach described in "Multi-controller-Based Cloud Resource Provisioning" section, and *BG* represents the proposed work in this paper.

Considering the *NASA* workload example, it can be seen from Fig. 7b that overall, all approaches performed well in terms of performance except *Aggressive* approach. If we compare the percentile results of the SLO violation, the *MC* approach has the same number of the violation as that of *RS* (i.e. 0.21 %), where the *BG* has comparatively less number of the SLO violation than all other approaches (i.e. 0.05 %). In terms of the 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 the *Worldcup* workload example, it can be seen from Fig. 7d that only *MC* and *BG* approach performed well in terms of achieving the better performance with less number of SLO violations (i.e. 0.56 and 0.29 %, respectively). Moreover, they have achieved the better performance at less cost than all the other approaches.

The key objective of any elasticity mechanism is to improve the performance of the 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*) performed better in performance as well as in cost. However, other approaches like *RS* also showed a good result in terms of performance in the first case, but at a higher cost. Moreover, the *NASA* workload is comparatively less dynamic than *Worldcup* in terms of jumps in varying workload regions. Comparing the results of *MC* and *BG*, we can observe that the *BG* shows a higher potential to achieve better performance with a bit higher but almost negligible cost than *MC*.

The above results demonstrate that adapting the BG-based action selection mechanism improves the overall results. However, another key aspect of adapting the BG-based approach is its ability of selecting the actions in a natural, bioinspired way, where it can improve the possibility of a smoother transition between different decisions.

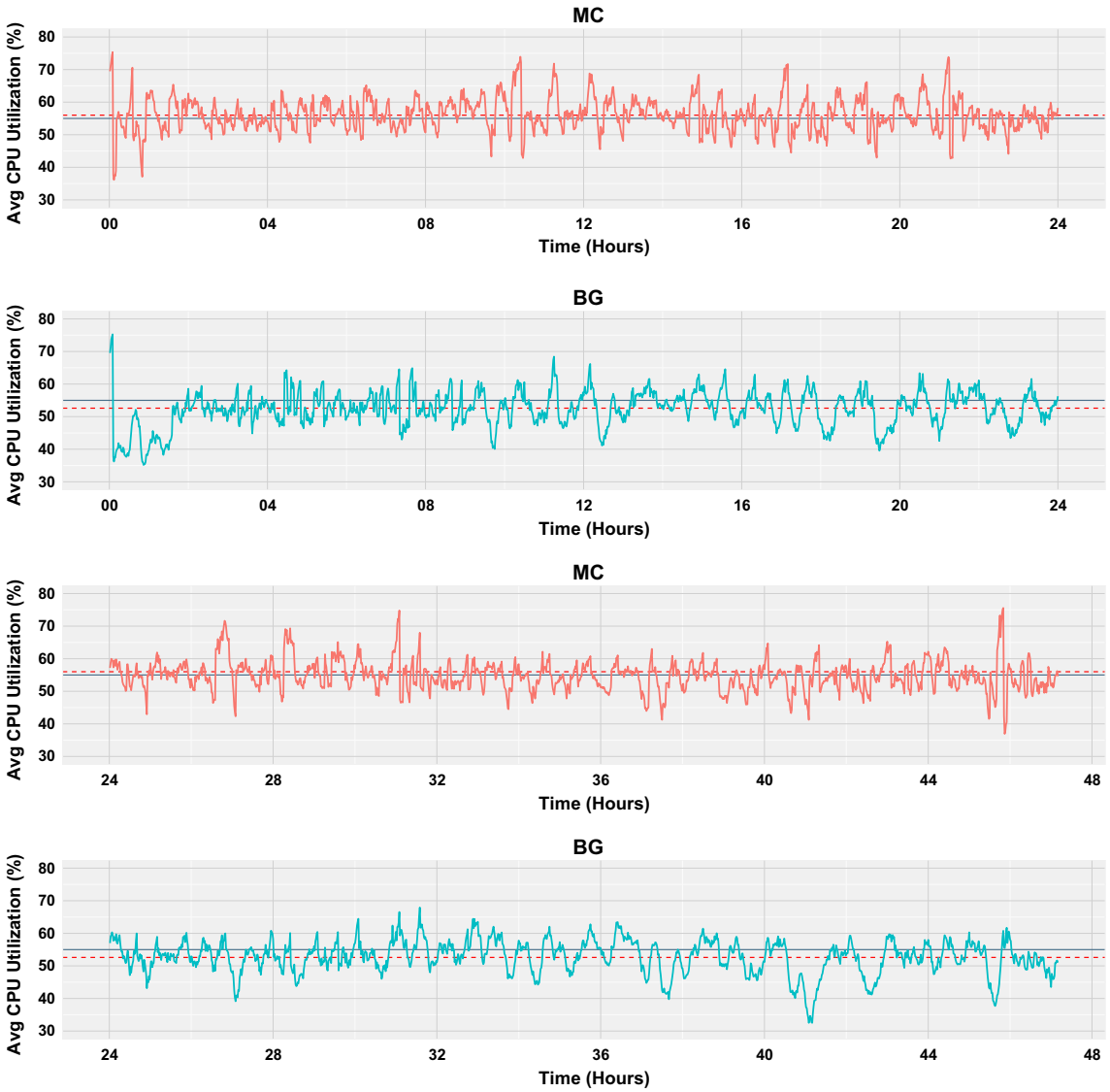

Fig. 9 Average CPU utilization of NASA experiment with 12 hours period in each graph. First and third rows belong to *MC*, while the second and fourth rows belong to *BG*

In current experimentation, we do not provide comprehensive quantitative measurements about how the BG-based approach improves the stability perspective of the underlying application. However, the results in Figs. 8 and 9 demonstrate some differences between *MC* and *BG* approaches with respect to the average CPU utilization recorded over the entire period of the *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 means that the performance of the system degrades. Figure 8 aggregates the count of the minutes for both approaches, when the CPU utilization is below and above the reference point. As can be seen from Fig. 8, it aggregates the count of the minutes for both approaches, when the CPU utilization is below and above the reference point. As can be seen from Fig. 8, during the total period of 2830 min, the *BG* approach maintains much longer time (i.e. 1892 min to be exact) for the CPU utilization to stay below 55 % in comparison with *MC* (which is 1354 min). This demonstrates that overall the *BG* approach maintained the CPU utilization closer under the reference point.

We further divide the measured CPU utilization for each approach into 24 h, which is presented in Fig. 9. This helps to visually demonstrate the difference between the approaches with respect to the measured CPU utilization against the reference point. The first and third rows belong to the *MC* approach, whereas the second and fourth rows

belong to the *BG* approach. The reference CPU utilization is represented with a dark solid horizontal line in all graphs. The following points are observed with respect to the differences between two approaches.

- The overall average CPU utilization for the *BG*-based approach is recorded as 52.58 %, whereas for the *MC* approach it is 56 %. They can be seen in red colour dashed lines in their respective graphs. Moreover, the *BG* reduces the likelihood of leading the system into an overloaded status as some of such occurrences can be found in the case of *MC* approach, e.g. the sessions 08th to 12th hour, and 20th to 24th hour.
- The CPU utilization in the *BG* case never reaches to 70 % in the entire period of the experiment except at the start, which is the same for both cases, whereas in the case of *MC*, it has been crossed a number of times.
- The CPU utilization in the *BG* case almost remains lower than 65 % except only four times. In the case of *MC*, there are quite a few times, where it remains 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 the case of *MC* has more abrupt transitions and peaks in comparison with the *BG* approach, which can cause the oscillatory behaviour.

In light of the above discussion, we can argue that the *BG* approach has the potential to reduce the likelihood of SLO violation by maintaining a desired CPU utilization, thus resulting in a better system performance. Moreover, compared with the MC approach, it shows smoother transitions between switching decision, which 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 thus we have not carried out a further theoretical stability analysis. However, an intuitive explanation is that the mixture of all controllers is done [in Eq. (3)] in a bioinspired way augmented by the BG process, which facilitates a natural selection of actions that results in less "bumping" at the switching time [58]. Moreover, the computational model of [21, 22] in particular is proved to successfully avoid the oscillation and keep the energy efficiency in various action selection problems [17]. In future, we aim to use the enhanced version of the BG model developed in [17], for which the formal stability proof can be established using the contraction theory of dynamical systems.

Conclusion and Future Work

We address the problem of cloud resource provisioning as an action selection problem. We propose a biologically inspired soft switching approach to implement horizontal cloud elasticity. The proposed approach integrates 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 increasing 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 evaluate 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 show the suitability of the biologically inspired method of action selection in the context of cloud computing. Our 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 approach 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) online learning capabilities of switching rules and (4) the possibility to enhance the capability of the framework by incorporating the vertical elasticity will be explored.

Acknowledgments The research work carried out in this paper is funded through a PhD scholarship programme 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) and the recent EPSRC grant (Ref. EP/I009310/1). Finally, the EPSRC funded ARCHIE-WeSt High Performance Computer (http://www.archie-west.ac.uk, under EPSRC grant no. EP/K000586/1) was used to obtain the simulation results reported in this paper.

Compliance with Ethical Standards

Conflict of Interest Amjad Ullah, Jingpeng Li, Amir Hussain, and Erfu Yang declare that they have no conflict of interest.

Informed Consent All procedures followed were in accordance with the ethical standards of the responsible committee on human experimentation (institutional and national) and with the Helsinki Declaration of 1975, as revised in 2008 (5). Additional informed consent was obtained from all patients for which identifying information is included in this article.

Human and Animal Rights This article does not contain any studies with human participants or animals performed by any authors.

References

1. Al-Shishtawy A, Vladimir, V. ElastMan: autonomic elasticity manager for cloud-based key-value stores. In: 22nd ACM international symposium on high-performance parallel and distributed computing, HPDC 2013, p. 115–116.
2. Garside J. Amazon's record $21bn Christmas sales push shares to new high, 2013.
3. Theguardian. China's Alibaba records 'singles day' sales of $8bn in 10 h, 2015.
4. Jamshidi P, Ahmad A, Pahl C. Autonomic resource provisioning for cloud-based software. In: Proceedings of the 9th international symposium on software engineering for adaptive and self-managing systems, 2014. p. 95–104.
5. Urdaneta G, Pierre G, van Steen M. Wikipedia workload analysis for decentralized hosting. Comput. Netw. 2009;53:1830–45.
6. Liu J, Zhang Y, Zhou Y, Zhang D, Liu H. Aggressive resource provisioning for ensuring QoS in virtualized environments. IEEE Trans Cloud Comput. 2014;2(3):119–31.
7. Herbst NR, Kounev S, Reussner R. Elasticity in cloud computing : what it is, and what it is not. In: 10th International conference on autonomic computing; 2013. p. 23–27.
8. Ranjan R, Wang L, Zomaya AY, Georgakopoulos D, Sun X-H, Wang G. Recent advances in autonomic provisioning of big data applications on clouds. IEEE Trans Cloud Comput. 2015;3(2):101–4.
9. Singh S, Chana I. QoS-aware autonomic resource management in cloud computing: a systematic review. ACM Comput Surv (CSUR). 2015;48(3):42.
10. Ali-Eldin A, Tordsson J, Elmroth E. An adaptive hybrid elasticity controller for cloud infrastructures. In: Network operations and management symposium (NOMS), 2012. p. 204–212.
11. Lim HC, Babu S, Chase JS, Parekh SS. Automated control in cloud computing: challenges and opportunities. In Proceedings of the 1st workshop on automated control for datacenters and clouds; 2009. p. 13–18.
12. Ghanbari H, Simmons B, Litoiu M, Iszlai G. Exploring alternative approaches to implement an elasticity policy. In: 2011 IEEE International conference on cloud computing (CLOUD); 2011. p. 716–723.
13. Lorido-Botran T, Miguel-Alonso J, Lozano JA. A review of auto-scaling techniques for elastic applications in cloud environments. J Grid Comput. 2014;12(4):559–92.
14. Ullah A, Li J, Hussain A. Towards workload-aware cloud resource provisioning using a novel multi-controller fuzzy switching approach. Int J High Perform Comput Netw. 2015. (in press).
15. Hellerstein JL, Diao Y, Parekh S, Tilbury DM. Feedback control of computing systems. Hoboken: Wiley; 2004.
16. Abdullah R, Hussain A, Warwick K, Zayed A. Autonomous intelligent cruise control using a novel multiple-controller framework incorporating fuzzy-logic-based switching and tuning. Neurocomputing. 2008;71(13):2727–41.
17. Girard B, Tabareau N, Pham Q-C, Berthoz A, Slotine. J-J. Where neuroscience and dynamic system theory meet autonomous robotics: a contracting basal ganglia model for action selection. Neural Netw. 2008;21(4):628–41.
18. Yang E, Hussain A, Gurney K. A brain-inspired soft switching approach: towards a cognitive cruise control system. In: WIT transactions on engineering sciences, 2014.
19. Redgrave P, Prescott TJ, Gurney K. The basal ganglia: a vertebrate solution to the selection problem? Neuroscience. 1999;89(4):1009–23.
20. Prescott TJ, Redgrave P, Gurney K. Layered control architectures in robots and vertebrates. Adapt Behav. 1999;7(1):99–127.
21. Gurney K, Prescott TJ, Redgrave P. A computational model of action selection in the basal ganglia. I. A new functional anatomy. Biol Cybern. 2001;84(6):401–10.
22. Gurney K, Prescott TJ, Redgrave P. A computational model of action selection in the basal ganglia. II. Analysis and simulation of behaviour. Biol Cybern. 2001;84(6):411–23.
23. Yang E, Hussain A, Gurney K. A basal ganglia inspired soft switching approach to the motion control of a car-like autonomous vehicle. Adv Brain Inspir Cogn Syst. 2013;7888:245–54.
24. Czubenko M, Kowalczuk Z, Ordys A. Autonomous driver based on an intelligent system of decision-making. Cogn Comput. 2015;7:1–13.
25. Prescott TJ, González FMM, Gurney K, Humphries MD, Redgrave P. A robot model of the basal ganglia: behavior and intrinsic processing. Neural Netw. 2006;19(1):31–61.
26. Cervantes J-A, Rodríguez L-F, López S, Ramos F, Robles F. Autonomous agents and ethical decision-making. Cogn Comput. 2015. doi:10.1007/s12559-015-9362-8.
27. Girard B, Cuzin V, Guillot A, Gurney KN. A basal ganglia inspired model of action selection evaluated in a robotic survival task. J Integr Neurosci. 2003;2(2):179–200.
28. Rohlfshagen P, Bryson JJ. Flexible latching: a biologically-inspired mechanism for improving the management of homeostatic goals. Cogn Comput. 2010;2(3):230–41.
29. Coutinho EF, de Carvalho Sousa FR, Rego PAL, Gomes DG, de Souza JN. Elasticity in cloud computing: a survey. Ann Telecomm-Annales Des Télécommunications. 2015;70:1–21.
30. Amazong. Amazong auto scaling, 2015.
31. Rightscale. Set up autoscaling using alert escalations, 2015.
32. Casalicchio E, Silvestri L. Autonomic management of cloud-based systems: the service provider perspective. In: Computer and information sciences III: 27th international symposium on computer and information sciences. London: Springer; 2013. p. 39–47.
33. Hasan MZ, Magana E, Clemm A, Tucker L, Gudreddi SLD. Integrated and autonomic cloud resource scaling. Proceedings of the 2012 IEEE network operations and management symposium, NOMS 2012, p. 1327–1334.
34. Barrett E, Howley E, Duggan J. Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. Concurr Comput: Pract Exp. 2013;25(12):1656–74.
35. Bahrpeyma F, Zakerolhoseini A, Haghighi H. Using IDS fitted Q to develop a real-time adaptive controller for dynamic resource provisioning in cloud's virtualized environment. Appl Soft Comput. 2015;26:285–98.
36. Lim HC, Babu S, Chase JS. Automated control for elastic storage. In: Proceedings of the 7th international conference on autonomic computing; 2010. p. 1–10.
37. Al-Shishtawy A, Vlassov V. ElastMan: elasticity manager for elastic Key-Value stores in the cloud. In: Cloud and autonomic computing conference (CAC '13); 2013. p. 1.
38. Ali-Eldin A, Kihl M, Tordsson J, Elmroth E. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In: Proceedings of the 3rd workshop on scientific cloud computing date; 2012. p. 31–40.
39. Patikirikorala T, Colman A, Han J, Wang L. 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; 2011. p. 218–227.
40. Patikirikorala T, Wang L, Colman A, Han J. HammersteinWiener nonlinear model based predictive control for relative QoS performance and resource management of software systems. Control Eng Pract. 2012;20(1):49–61.
41. Ali-Eldin A, Tordsson J, Elmroth E, Kihl M. Workload classification for efficient auto-scaling of cloud resources. Department of

Computer Science, Umea University, Umea, Technical Report, 2013.
42. Dan X, Liu X, Vasilakos AV. Traffic-aware resource provisioning for distributed clouds. IEEE Cloud Comput. 2015;2(1):30–9.
43. Zhang Q, Member S, Zhani MF. Dynamic heterogeneity-aware resource provisioning in the cloud. IEEE Trans Cloud Comput. 2014;2(1):14–28.
44. Tony Prescott M. Action selection, 2008.
45. Fix J, Rougier N, Alexandre F. A dynamic neural field approach to the covert and overt deployment of spatial attention. Cogn Comput. 2011;3(1):279–93.
46. Gurney KN. Reverse engineering the vertebrate brain: methodological principles for a biologically grounded programme of cognitive modelling. Cogn Comput. 2009;1(1):29–41.
47. Mandali A, Rengaswamy M, Srinivasa Chakravarthy V, Moustafa AA. A spiking basal ganglia model of synchrony, exploration and decision making. Front Neurosci. 2015;9:191.
48. Redgrave P. Basal ganglia, 2007.
49. Hussain A, Abdullah R, Yang E, Gurney K. An intelligent multiple-controller framework for the integrated control of autonomous vehicles. In: Advances in brain inspired cognitive systems: 5th international conference, BICS 2012, Shenyang, China, July 11–14, 2012. Berlin, Heidelberg: Springer; 2012. p. 92–101.
50. Bai Y, Wang D. Fundamentals of fuzzy logic control—fuzzy sets, fuzzy rules and defuzzifications. In: Bai Y, Zhuang H, Wang D, editors. Advanced fuzzy logic technologies in industrial applications, advances in industrial control. London: Springer; 2006. p. 17–36.
51. Lyshevski SE. Control systems theory with engineering applications. New York: Springer; 2012.
52. Passino KM, Yurkovich S, Reinfrank M. Fuzzy control, vol. 42. CA: Addison-wesley; 1998.
53. Calheiros RN, Ranjan R, Beloglazov A, De Rose CAF, Buyya R. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw: Pract Exp. 2011;41(1):23–50.
54. Cingolani P, Alcala-Fdez J. jFuzzyLogic: a robust and flexible fuzzy-Logic inference system language implementation. In FUZZ-IEEE, Citeseer, 2012. p. 1–8.
55. Internet traffic Archive. Worldcup 1998 Web trace, 2015.
56. Network traffic Archive. Nasa-HTTP, 2015.
57. Amazon. Amazon EC2 pricing, 2015.
58. Yang E, Hussain A, Gurney K. Neurobiologically-inspired soft switching control of autonomous vehicles. In: Advances in brain inspired cognitive systems: 5th international conference, BICS 2012, Shenyang, China, July 11–14, 2012. Berlin, Heidelberg: Springer; 2012. p. 82–91.