Cluster Computing (2018) 21:1735–1764
https://doi.org/10.1007/s10586-018-2807-6

A control theoretical view of cloud elasticity: taxonomy, survey and challenges

Amjad Ullah[1] · Jingpeng Li[1] · Yindong Shen[2] · Amir Hussain[1]

Received: 10 August 2017 / Revised: 27 February 2018 / Accepted: 26 April 2018 / Published online: 7 May 2018

Abstract
The lucrative features of cloud computing such as pay-as-you-go pricing model and dynamic resource provisioning (elasticity) attract clients to host their applications over the cloud to save up-front capital expenditure and to reduce the operational cost of the system. However, the efficient management of hired computational resources is a challenging task. Over the last decade, researchers and practitioners made use of various techniques to propose new methods to address cloud elasticity. Amongst many such techniques, control theory emerges as one of the popular methods to implement elasticity. A plethora of research has been undertaken on cloud elasticity including several review papers that summarise various aspects of elasticity. However, the scope of the existing review articles is broad and focused mostly on the high-level view of the overall research works rather than on the specific details of a particular implementation technique. While considering the importance, suitability and abundance of control theoretical approaches, this paper is a step forward towards a stand-alone review of control theoretic aspects of cloud elasticity. This paper provides a detailed taxonomy comprising of relevant attributes defining the following two perspectives, i.e., control-theory as an implementation technique as well as cloud elasticity as a target application domain. We carry out an exhaustive review of the literature by classifying the existing elasticity solutions using the attributes of control theoretic perspective. The summarized results are further presented by clustering them with respect to the type of control solutions, thus helping in comparison of the related control solutions. In last, a discussion summarizing the pros and cons of each type of control solutions are presented. This discussion is followed by the detail description of various open research challenges in the field.

Keywords Cloud elasticity · Elastic feedback controllers · Control theory · Dynamic cloud resource provisioning · Cloud resource management

1 Introduction

Elasticity is the most promising feature of cloud computing, which enables the readjustment of the underlying computational resources at runtime to meet application demands. This helps to avoid the degradation of system performance, to reduce operational cost and to minimize the energy consumption of the system [1]. An elastic policy is usually required to exploit the elastic architecture of cloud computing. This policy is responsible for maintaining the performance of the system at an acceptable level with the lowest cost possible. However, providing such an efficient policy is a challenging task.

Over the years with the rise in popularity of Internet based applications, the notion of providing better elasticity management has increased. This has proportional effects

✉ Amjad Ullah
aul@cs.stir.ac.uk

Jingpeng Li
jli@cs.stir.ac.uk

Yindong Shen
yindong@hust.edu.cn

Amir Hussain
ahu@cs.stir.ac.uk

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

[2] School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China

on cloud elasticity literature, where researchers and practitioners made use of various techniques ranging from simple "if-then" kind of rules to complex machine learning based algorithms. Control theory is one of such techniques that provides a systematic method to design feedback controllers to implement cloud elasticity. Such feedback controllers are designed to be stable in order to avoid oscillation and settle quickly to the steady state by appropriately responding to disturbances. They are better for achieving service level objectives, such as response time or throughput [2]. The use of control theory is not limited to the advent of cloud computing or elasticity. In the past, control theory has been a well-recognized approach to achieve the desired QoS needs of computing systems. For example, feedback controllers are exploited to achieve the target performance objectives of computing systems like Webservers [3, 4], database servers [5], cache servers [6], etc.

Many research undertakings are carried out on cloud elasticity, and its various aspects are explored. There are also several survey papers available that provide a concise review of different aspects of cloud elasticity. Lorido-Botran et al. [7] classified the overall elasticity proposals based on the underlying implementation techniques, whereas Naskos et al. [8] distributed the literature based on the decision-making mechanism, and Coutinho et al. [9] on the other hand provided a systematic review of utilised performance metrics, measurements tools and evaluation. The scopes of all these papers have been broad where they mainly focused on the high-level view of overall elasticity research rather than the specific details on one implementation technique. In this work, considering the importance, suitability and abundance of control theoretical approaches in the context of cloud elasticity, a standalone review paper is targeted, focusing only on control theoretical methods of cloud elasticity.

The main contributions of this paper include (1) the proposition of a taxonomy that includes characteristics from both (i.e., elasticity and control theory) perspectives, (2) an exhaustive, up-to-date survey of the literature in accordance with the taxonomy, and (3) an overview of the open issues and research challenges. This paper however, does not cover the following aspects of elasticity domain including the commonly used experimental platforms, real workloads' data, monitoring tools, application benchmarks. The key reason behind is that these important aspects are already fully covered in various related survey papers such as [7, 10].

This paper will help researchers of the target area to understand the various related concepts of cloud elasticity including control theoretic approaches, different possible types of control solutions, how these are used to implement cloud elasticity, their pros and cons and the open research challenges. This paper consolidates the available research works using a large set of attributes highlighting the important aspects from both, i.e., the application domain (elasticity) and implementation technique (control-theoretic) perspectives. The inclusion of these large set of attributes help to better analyse and compare the related approaches. This paper clusters the existing approaches based on the type of feedback controllers. Such a classification will help the researchers in analysing and benchmarking the control solutions of a particular type.

The rest of the paper is organized as follows. Section 2 describes the related surveys and how this article is distinct from them. Section 3 explains the proposed taxonomy that has been developed to conduct the literature review. Section 4 examines the elasticity literature, whereas Sect. 5 provides a discussion and presents open issues and research challenges of the field. Finally, Sect. 6 concludes the paper.

2 Related surveys

This section briefly describes the current survey papers as related work and provides a brief explanation of how the review conducted in this paper is different. For this purpose, we classify the relevant review papers into the following three categories based on their primary strengths.

2.1 Cloud resource management

The review articles in this category mainly cover an extensive range of cloud resource management related problems such as provisioning, allocation, scheduling, mapping, adaptation, discovery and brokering. Amongst these problems, cloud elasticity (or dynamic cloud resource provisioning) approaches are covered either partially or in a limited capacity. For example, Singh and Chana [11] focused on autonomic computing with a particular emphasis on QoS-aware management of resources; Jennings and Stadler [12] used resource management functions as a classification method; Mustafa et al. [13] reviewed the literature based on the metrics used and discussed the underlying research problems. Manvi and Shyam [14] classified the literature into problem specific categories such as resource provisioning and allocation; whereas Singh and Chana [15] targeted resource provisioning in general, wherein elasticity is considered as a trait of resource provisioning mechanism.

2.2 Adaptability using control theory

The review papers in this category are related as their primary focus is on the use of control theory in similar context, e.g., QoS management or adaptation in general.

However, none of them has considered control solutions that implement elasticity. Yfoulis and Gounaris [16] briefly investigated the control theoretical perspective in cloud computing context with a focus on SLA management. More relevant however, brief discussions on the use and suitability of feedback controllers for performance management in larger context of cloud computing domain is carried out in [17]. Gambi et al. [18] focused on the assurance and adaptability perspective of cloud controllers. However, their scope is wider and also includes other techniques such as rule-based and machine learning. Patikirikorala et al. [19] carried out a systematic survey of the design of self-adaptive systems using control solutions. They presented a quantitative review based on a taxonomy consisting of attributes such as target system, control system and validation mechanism. The scope of their research, however, is much wider on general adaptive systems rather than cloud elasticity. Moreover, they only presented a quantitative analysis of the existing research works rather than a detailed review.

2.3 Cloud elasticity

A comprehensive survey on cloud elasticity is carried out in [7], where the authors classified the overall elasticity literature based on the underlying implementation techniques. Galante and De Bona [1] classified them into infrastructure and application level, and a taxonomy consisting of features like scope, purpose, decision-making mechanism, action type and evaluation is proposed in [8]. A similar taxonomy is also provided in [20] with a focus on the application provider perspective. The authors of [21] focused on strategy, action type and architecture perspective. Whereas, an adaptability view of computational resources with a larger scope including concepts like node adaptation and virtual machine (VM) migration is provided in [22]. They, however, used adaptation techniques as one of the dimensions to review the literature. At last, elasticity functions such as reactive migration, resizing and proactive replication are used as a means of classification in [23].

3 Taxonomy

The primary focus of the survey articles reviewed in Sects. 2.1 and 2.2 are not cloud elasticity. However, they represent the set of problems (and systems) amongst which cloud auto-scaling is a subset. In contrast, the survey papers reviewed in Sect. 2.3 are particularly focused on cloud elasticity and therefore closely related to this survey paper. All the survey papers reviewed are very innovative and mostly overlapped regarding the essential elasticity features, e.g., elasticity type (Reactive/Proactive), trigger (Horizontal/Vertical), scope [Cloud provider (CP)/Service provider (SP)], etc. However, their scope is wide, i.e., overall cloud elasticity research and apart from [7], they lack details on the underlying implementation techniques of the proposed solutions. Majority of the existing such review papers utilise the various cloud elasticity features to classify the overall cloud elasticity literature. In such a classification, the underlying implementation technique is consider as one attribute, (e.g., in [7, 9]) and therefore no classification of the existing literature is performed using the attributes of a particular implementation technique. This result in the lack of conducting an exhaustive review of the proposals of each implementation technique.

In contrast to the existing survey papers of cloud elasticity, the key aim of this review paper is to propose a technique specific (control theory in this case), an up-to-date and exhaustive review of cloud elasticity solutions. A closely related survey paper in this regard is the research work conducted by Patikirikorala et al. [19], where they carried out a systematic survey of the design of self-adaptive systems using control solutions. However, the scope of this paper is much wider, i.e., self-adaptive systems, where cloud auto-scaling is a small part of it. Secondly, in their paper, no discussion is made about elasticity perspective nor any elasticity attributes are considered. Furthermore, they conduct a quantitative review rather than an analysis of the existing approaches in the context of cloud elasticity. In contrast, we aim to focus on the implementation perspective of cloud elasticity only (rather than larger adaptive systems) using control theoretical approaches.

The implementation of cloud elasticity using a control theoretical approach commonly uses a feedback loop model, where a controller maintains the output of the system around some desired value by monitoring the inputs and outputs of the system. Generally such a control system can be used to satisfy a constraint or guarantee an invariant on the outputs of the system [25]. Figure 1 depicts the general mechanism of such a feedback model where it observes the system output to correct any deviation from the desired value. The basic elements of the control systems can be seen from Fig. 1. The details of these elements describe the implementation aspects of a control systems. Therefore, these elements become the attributes of the taxonomy and will helps us to analyse the implementation

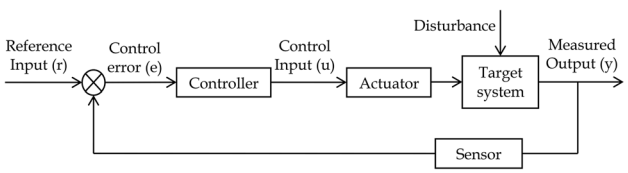

Fig. 1 Block diagram of feedback control system adapted from [24]

perspective of a proposed solution. However, these attributes do not tell anything about the elasticity characteristics of the proposed system. Therefore, we also included the basic characteristics of the elasticity domain, which are commonly used in the related survey papers such as [9]. Thus to summarise, this section introduce the taxonomy (shown in Fig. 2) and briefly explains its various characteristics. This taxonomy consists of characteristics from control theoretical point of view (i.e., as an implementation technique) as well as from cloud elasticity perspective (i.e., as an application domain). In terms of the structure, our taxonomy complements the taxonomies proposed in [9, 26]. The following subsections explain all the characteristics of the taxonomy in detail.

3.1 Control solution view

The various essential elements of a control system can be seen from Fig. 1. The brief explanations of all these elements are provided below.

(1) *Control objective* refers to the main intended purpose for which a control system is developed, e.g., to maintain an overall average response time of less than t seconds.

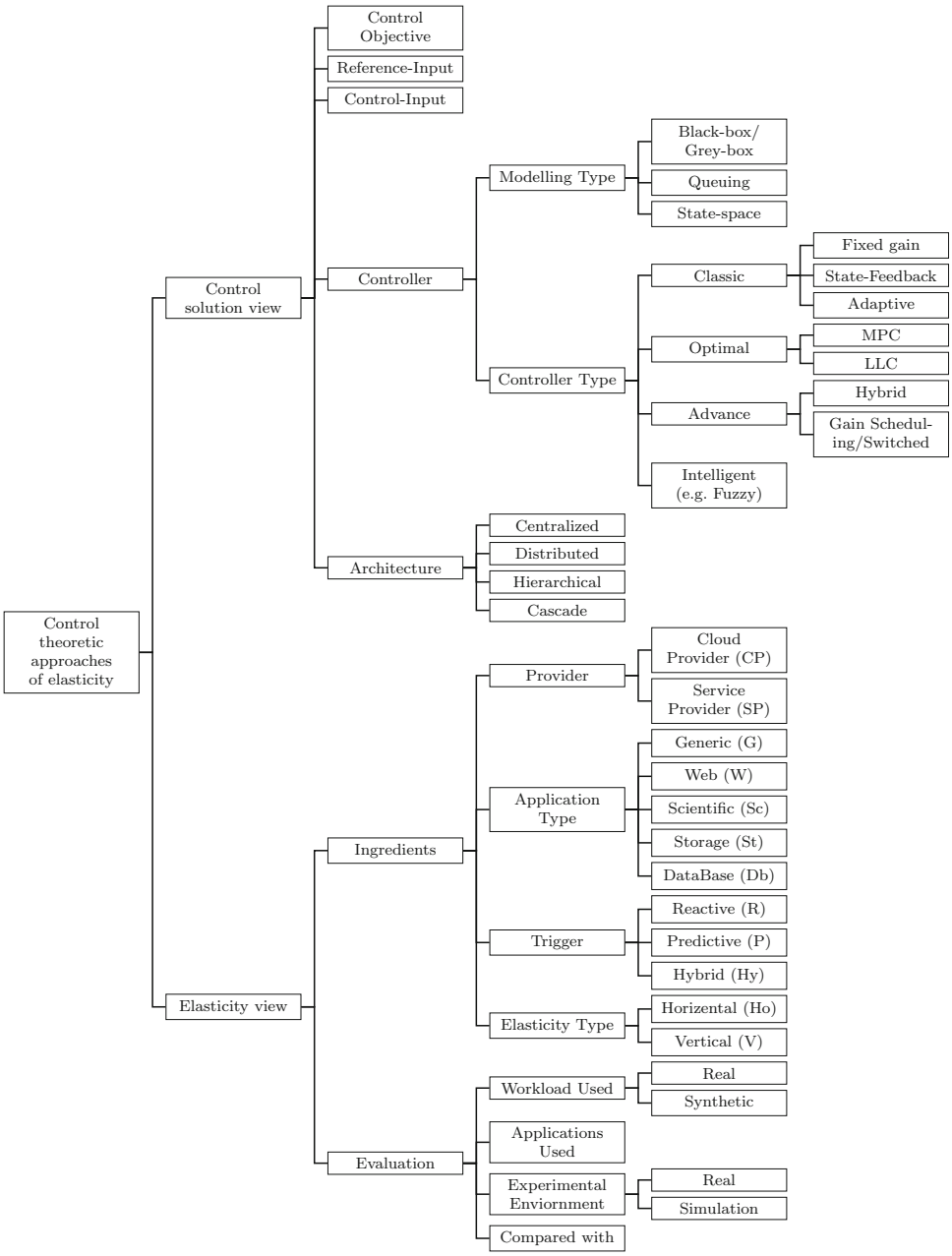

Fig. 2 Taxonomy of control theoretic elasticity

(2) *Reference input* refers to the desired value of the system output that the controller is required to maintain. For example, the overall average CPU utilisation of all acquired VMs (cluster) must be 60%.
(3) *Control error* refers to the difference between desired reference input and the measured value of system's output.
(4) *Control input* refers to the dynamic parameter computed by the controller that affects the behaviour of the target system to achieve the desired reference input, e.g., the number of VMs.
(5) *Actuator* is a component that executes the decision made by the controller.
(6) *Sensor* measures the values of metrics needed by the controller for making the next scaling decision. For example, to measure the CPU utilisation of VMs.
(7) *Controller* is the mechanism that computes the values for the *Control input* required to achieve the desired objective value, e.g., *Reference input* by taking into account various measurements. The systematic design of a feedback controller consists of the following two steps: (1) the formal construction of a system model, and (2) the implementation of a control mechanism. Based on this description, the *Controller* is divided into the following two subcategories:
    (i) *Modelling type* the model captures the behaviour of a target system, which represents the corresponding time-varying relationship between system inputs (e.g., number of VMs) and outputs, (e.g., Response time) [27]. In literature, there are different types of modelling techniques used for the design of elastic control solutions. These types can be seen from Fig. 2, whereas their brief explanation is provided in Sect. 4.4.
    (ii) *Controller type* there are various types of controller used for the implementation of elasticity. We have clustered them into four groups adapted from the controller types used in [19]. These types can be seen from Fig. 2, whereas their brief explanation and the review of the control solutions belong to these types are provided in Sect. 4.4.
(8) *Architecture* of a control system refers to the pattern of how a particular control methodology is implemented. The most common patterns observed in cloud elasticity research include *Centralised* and *Decentralised* (also known as *Distributed*). However, there are also few cases, where *Cascade* and *Hierarchical* patterns are used as well. The brief description of each of these patterns and the overview of the control solutions following these patterns are further provided in Sect. 4.5.

The *Disturbance* in Fig. 1 refers to the workload, whereas the *Measured output* is the latest measurement of the system output. All of the above mentioned elements of a control system except *Control error*, *Actuator* and *Sensor* are part of the taxonomy.

3.2 Elasticity view

This section covers the attributes from the cloud elasticity perspective that defines different aspects of an autoscaling approach. These attributes are already addressed in existing review papers [1, 7, 8, 20, 23], however, mostly as classification factors. In contrast, we use the attributes of control solution view as classification attributes. The elasticity attributes are included to highlight the elasticity perspective of the reviewed proposals. The brief description of the attributes considered is as follow:

(1) *Provider* an elasticity proposal targets the aims of a particular stakeholder. This attribute can be divided into two categories, i.e., *CP* and *SP*. The *CP* specifies that a proposed elasticity method is implemented by the infrastructure provider, whereas the *SP* indicates that the elasticity mechanism is implemented by the user of the cloud, who deploy their applications/services over the cloud infrastructure. The *CP* aim of performing dynamic resource provisioning is to increase the efficiency of their under-utilised computational nodes by shutting down some servers and shifting their load to others, hence minimising the energy consumption to reduce electricity costs as well as CO_2 emission. Alternatively, *CPs* could also oversubscribe their resources, hence maximising their revenue. In contrast, the *SPs* are concerned with the efficient use of their rented computational resources, so that they can release any under-utilised or unused VMs to reduce their service operating costs. This attribute of the taxonomy specifies the type of stakeholders, which demonstrates the aim and purpose of a given elasticity proposal.
(2) *Application type* the auto-scaling approaches are proposed for different kind of applications. This attribute refers to the nature of the application for which the elasticity method is proposed. Possible types can be seen from Fig. 2.
(3) *Trigger* this represents the triggering behaviour of an elasticity method. The possible types include *Reactive*, *Predictive* or *Hybrid*. In the case of *Reactive*, the auto-scaling system performs the scaling

decision in response to changes in the behaviour of the system. The *Predictive* anticipates future behaviour of the system and performs the scaling decision in advance, whereas, the *Hybrid* combines both the *Reactive* and *Predictive* mechanisms.

(4) *Elasticity type* refers to the type of resource scaling that can be either *Horizontal* or *Vertical*. The *Horizontal* elasticity enables the increase or decrease in the number of VMs, whereas, the *Vertical* elasticity allows changes in the specification of existing VMs, e.g., the increase or decrease in CPU and/or memory capacity of one or a set of VMs.

(5) *Evaluation* this attribute of taxonomy highlights how the assessment of a particular approach is carried out. This consists of the following specifications:

   (i) *Workload used* for the evaluation can be either real or synthetically generated. This attribute represents the nature of the workload and its brief description.
   (ii) *Applications used* includes the details of any applications used either for the generation of workload or experimentation purposes.
   (iii) *Environment* includes the particulars of the experimental set-up.
   (iv) *Compared with* specifies the approaches or scenarios used for comparison purposes.

4 Review of existing control theoretical approaches of elasticity

This section provides the details of the existing cloud elasticity approaches that are implemented using control theory. The review is carried with respect to each attribute of the control solution view of the taxonomy. The summarised results are clustered by the *Controller Type* attribute of the taxonomy and presented in Tables 1, 2, 3, 4, 5, 6 and 7.

4.1 Control objective

In general, there are three different types of *Control objective*. The brief descriptions of these types are the following:

– *Regulatory purpose* a feedback controller developed for regulatory purposes maintains system output close to the desired reference value. For example, the average CPU utilisation of the *Cluster* must be 60%.
– *Optimisation* the controller is responsible to obtain the best settings for the system output in the presence of certain constraints. For example, minimisation of system's response time with the lowest possible cost.
– *Disturbance rejection* such a controller is used to manage and adjust the level of disturbances, e.g., *Admission control system*. It only allows enough workload that does not affect the performance of the system.

In cloud elasticity domain, the majority of existing solutions belong to the category of either regulatory control (e.g., [28–33]) or optimisation (e.g., [34–39]). The control solutions having the objective of disturbance rejection often assist another control solution (e.g., [40–43]). Irrespective of these types, the key objective of any elastic control solution is to improve the utilisation of computational resources whilst maintaining acceptable level of performance of the system and reducing its operational cost. This objective however, can be viewed differently by CPs and SPs.

The CPs perspective of better resource utilisation is to improve performance of the system and to reduce the operational cost of data centre, e.g., decrease in electricity consumption, increase in revenue generations using oversubscription, and reduction in the CO_2 emissions. From the SPs perspective, it is to reduce the operational cost of the services consumed whilst simultaneously maintaining performance and reliability of their deployed services. These different points of views dictate the design of control solution for various purposes. The control solutions reviewed in this paper carries one or multiple of the following purposes including maintaining an acceptable level of performance, reducing operating cost, minimising energy consumption and maintaining capacity level.

It is evident from the analysis of Tables 1, 2, 3, 4, 5, 6 and 7 that the objective of the majority of the control solutions is to improve system performance. These objectives are either in the form of the regulation of different performance metrics (e.g., the control solutions proposed in [31, 41, 44–47] aiming to maintain the response time of system less than certain threshold) or in the form of optimisation of the system (e.g., obtaining the best value for the number of VMs to avoid under-utilised and over-utilised behaviour [48]). In such control solutions, the operating cost factor is indirectly considered, but the primary objective is to maintain the desired level of performance. Alternatively, the control solutions proposed in [28, 35–37, 49] directly consider cost as one of the objective by the system.

Table 1 Fixed gain controllers

	[72]	[60]	[44]	[43]	[48]	[53]	[50]	[61]
Type	PID	PID	PID	PI	PD	Fixed gain	Integral	Integral
Model	State-space	Queuing	–	Grey-box	–	Black-box	Black-box	Black-box
Architecture	Centralized	Distributed	Centralized	Centralized	Centralized	Centralized	Centralized	Centralized
Control objectives	99th % read operation latency	Application SLO (response time) at a pre-defined level	Maintain a desired response time	Ensure service time constraints	Optimal number of VMs avoiding under/over utilized scenarios	Desired memory utilization	Desired CPU utilization	Maintain a desired response time
Reference inputs	Service time	CPU utilization	CPU utilization	Service time	Server load and memory utilization	Memory utilization	CPU utilization	CPU utilization
Control input	Number of Voldmart nodes	Number of VMs	Number of VMs	Number of map-reduce nodes, number of clients	Number of VMs	Memory allocation	Number of VMs	Number of VMs
Monitoring metrics	Read latency without round-trip time	Mean CPU utilization of tier's VMs	Mean CPU utilization of cluster	Service time and number of clients	Server load and memory utilization	CPU and memory usage, page fault rates	CPU load, arrival rate	CPU load, arrival rate
Ingredients	SP St R Ho	SP W R Ho	SP W R Ho	– D Hy Ho	CP W R Ho	– W R V	SP W R Ho	SP St R Ho
Workloads	Synthetic: generated using YCSB	Real: FIFA	Synthetic	Synthetic	Synthetic	Real	Synthetic	Synthetic
Applications used	YCSB	–	Hogna Framework	MapReduce Benchmark Suite	–	httperf, MemAccess	–	Cloudstone
Environment	Real: Voldmart	Real: Amazon	Real: Amazon and SAVI [117]	Real: Grid5000 [118]	Real: Amazon	Real: Custom (4 HP Proliant servers) + Xen 2.6.16	Real: ORCA [119] + Xen as Hypervisor	Real: ORCA [119] + Xen as Hypervisor
Compared with	Not provided	Threshold based, proportional controller	Not provided	Not provided	Compared with [81]	Not provided	Static threshold, integral control	Static provisioning

The second popular category of control objective after performance management is the maintenance of resource level capacity. The capacity level in most cases has an indirect relation with the performance of the system. However, in terms of implementation, the control solution is responsible for maintaining the utilisation (or allocation) level of system resources. In such cases, we found the following two possibilities: (1) in the case of horizontal elasticity, the common objective is to maintain the CPU utilisation level of overall cluster (e.g., [50–52]). (2) Whereas, in the case of vertical elasticity, the objective depends on the nature of reconfigurable resources. For example, there are control solutions, where the control objective is the readjustment of the memory allocation (such as [53, 54]) or the readjustment of the CPU allocation (such as [49, 55–58]).

In last, the following control solutions [34, 39, 45, 59] have considered the energy efficiency perspective of cloud elasticity, where the control objective is to reduce the power consumption behaviour of the system in association with the performance goal.

Table 2 State-space feedback and adaptive approaches

	[28]	[29]	[30]	[31]	[32]	[33]	[86]	[68]
Type	SSF + FC	SSF	SSF (adaptive)	Adaptive	Adaptive + hill climbing	Adaptive	Adaptive	Adaptive (integral)
Models	State-space + LQR	State-space	Black-box + LQR	Black-box	Queuing	Queuing	Queuing	Black-box
Architecture	Centralized and cascade	Centralized	Distributed	–	Centralized	Centralized	Centralized	Hierarchical
Control objectives	To meet SLO (CPU load, response time and bandwidth)	To meet performance (CPU load, response time and throughput)	To achieve applications' SLO (throughput and response time)	To guarantee target performance (response time)	To achieve application performance (latency)	To guarantee SLA (max number of requests handled per time unit)	Macro level analysis to evaluate controller performance	To achieve application level QoS (response time, throughput)
Reference input	CPU load, response time, cost	CPU load, response time, throughput	Response time, throughput	Response time	Latency	Un-handled service request in past interval	Over- and under-provisioned systems	CPU utilization
Control input	Number of storage nodes	Number of VMs	CPU, memory and I/O allocation	Memory size	Virtual CPUs	Number of VMs	Number of VMs	CPU entitlement
Monitoring metrics	CPU load, response time, bandwidth, total cost	CPU usage, response time, throughput	CPU and memory usage, I/O consumption	Response time	Arrival rate, performance, CPU usage, queue length	Arrival requests, service rate	Service capacity, arrived requests, buffer size, processed requests	CPU usage, workload arrivals
Ingredients	SP St R Ho	– G R Ho	SP G R V	SP W R V	SP G R V	CP G Hy Ho	SP/CP Sc Hy Ho	CP W R V
Workloads	Synthetic	Synthetic	Synthetic	Real: FIFA, Wikipedia	Real: FIFA	Real: FIFA	Real: FIFA, Google traces	Synthetic
Applications used	–	httperf	httperf, RUBiS, TPC-W	RUBBoS	Zimbra load generator	–	–	RUBiS , TPC-W
Environment	Simulation: EstoreSim	Real: Eucalyptus private cloud	Real: Linux machines + KVM	Real: Linux machine + KVM	Real: Linux machines + Zimbra [120]	Simulation: discrete event simulator	Simulation: discrete event simulator [121]	Real: 5 HP Proliant Servers + Xen
Compared with	Scenario without any auto-scaling	Not provided	Scenario without any controller	Non-adaptive (details not provided)	Threshold-based utilization controller	Regression based controller [122]	Reactive controller [123]	None provided

Table 3 Adaptive approaches

	[62]	[63]	[73]	[75]	[49]	[55]	[54]
Types	Adaptive (optimal)	Adaptive PI + pole placement	Adaptive PI	Nested adaptive (integral)	MIMO adaptive PI + RL	SISO, MIMO and adaptive MIMO	Adaptive
Model	Black-box (ARMA 2nd order)	Black-box (1st order AR model)	Least square regression	–	ARMAX (2nd order) + SVM [124]	Kalman filter	Linear regression (1st order)
Architecture	Distributed	–	Distributed	Cascade and distributed	Centralized	Centralized	Centralized (node level)
Control objectives	To achieve application level QoS (response time)	To achieve application level QoS (response time)	To obtain target job progress to meet deadline	To achieve target QoS goal (response time)	To maximize application benefit (QoS) within time and budget constraints	To maintain CPU allocation right above the CPU utilization	Adjustment of memory size to achieve desire application response time
Reference input	Per application response time	Response time	Target job progress	Response time, CPU utilization	Benefit function, execution time, resource cost	CPU utilization	Response time
Control input	CPU entitlement and IO allocation	CPU entitlement	CPU share	CPU allocation	Adaptive parameters	CPU allocation	Memory size allocation
Monitoring metrics	CPU usage, response time, disk usage	Response time	Job progress, milestone	CPU utilization, measured response time	Adaptive parameters, CPU and memory usage	CPU utilization	Measured response time, memory utilization
Ingredients	CP W P V	– W R V	SP Sc R V	– G R V	SP/CP G P V	– W P V	SP W R V
Workloads	Synthetic	Synthetic	Synthetic	R: SPECweb99 [125]	–	Synthetic	R: FIFA, Wikipedia
Application used	RUBiS, TPC-W, custom: secure media server	httperf	ADCIRC, OpenLB, WRF, BLAST and Montage	httperf	Great Lake nowcasting and forecasting, volume rendering	RUBiS	RUBBoS, httpmon
Environment	Real: two test-beds (HP C-class blades and Emulab [126])	Real: two machines, i.e., HP9000-R server and Pentium III	Real:8-core AMD server + Hyper-V and Xen	Real: two machines, i.e., HP9000-L server and HP LPr Netserver	Real: two private clusters (each with 64 nodes)	R: three machines with Xen 3.0.2 Hypervisor	R: 32 cores and 56 GB memory based machine + Xen hypervisor
Compared with	Two cases: work-conserving and static allocation	Fixed PI controller	Feedback approach of [127]	Single loop QoS controller, utilization controller	Work conserving, static scheduling	Not provided	Capacity based [128] and performance based [46] memory controllers

4.2 Reference input

The reference input in most of the cases reflects the corresponding control objective of the control systems. However, this is not always the case, e.g., the objective of the control solutions proposed in [44, 60, 61] is to maintain the desired level of system performance but the reference input in each case is the CPU utilisation. Therefore, we analyse the reference input independently from control objective attribute.

The analysis of the reference input for each reviewed solution presented in Tables 1, 2, 3, 4, 5, 6 and 7 hints on the use of three types of metrics. These include *Performance-based*, *Capacity-based* and the combination of both.

Table 4 Optimal approaches

	[45]	[37]	[38]	[35] and [36]	[39]	[34]	[59]
Types	Distributed MIMO MPC	Stochastic MPC	MPC + event triggering	[35]: MPC, [36] MPC + k-means clustering	LLC	LLC	LLC + search methods
Models	Fuzzy logic + ANN	Queuing + Kalman filter	G	ARIMA [129]	ARIMA + Kalman filter	Kalman filter	ARIMA + Kalman filter
Architecture	Distributed	–	Centralized	Centralized	Centralized	Hierarchical	Centralized
Control objectives	To achieve end-to-end desired response time of co-located web applications	To minimize application response time and cost	To reduce cluster reconfiguration decisions	To obtain optimal trade-off between energy saving and reconfiguration cost	To maximize profit and minimize power consumption	To maximize profit and minimize power consumption	To obtain a desired response time while reducing power consumption
Reference input	Per application response time	–	–	SLA (average task scheduling delay)	Response time of each client class	Response time of each client class	Response time
Control input	CPU and memory allocation	Number of VMs	Map-reduce cluster size	Number of machines	Cluster size, operating frequencies	Cluster size, CPU share, active host machines, workload share of VM	Operating frequency
Monitoring metrics	CPU and memory usage of local and neighbour VMs, energy usage, response time	Resource usage, response time, cost, service rate of cluster, queue length, arrival rate	Number of clients, service time, availability	Queue length, SLA, CPU and memory usage, arrival rate	Arrival rate, response time	Queue length, arrival rate, processing rate, response time	Arrival rate, queue length, power consumption
Ingredients	CP W P V	SP G P Ho	– St Hy Ho	CP G P Ho	CP G P Ho/V	CP G P Ho/V	CP G P V
Workloads	Real: FIFA	Real: FIFA	Real: Taobao [130]	Real: Google [131]	Real: FIFA	Real: FIFA	Real: http traces [132]
Application used	RUBiS and custom web service	Convex optimization solver	Map-reduce framework	–	–	IBM's trade6 benchmark, httperf	–
Environment	Real: Dell PowerEdge R610 servers + VMWare enviornment	Simulation: custom simulator	Simulation: Matlab Simulink	Simulation: trace driven simulation used in [35], Matlab in [36]	Simulation: Matlab simulation	R: Dell PowerEdge servers + VMWare ESX	Simulation: Matlab
Compared with	Perfume [98], PAC [133]	None provided	Time based control mechanism, error based event mechanism	[35]: Not provided, [36]: heuristic based provisioning	A system without LLC	A system without LLC	Comparison among various search methods

Table 5 Hybrid approaches

	[42]	[74]	[64]	[56]	[57]	[46]	[69]
Types	Adaptive integral + fixed gain + heuristic	Feedback (PI) + feed-forward (MPC)	Feedback + feed-forward + integral	Feedback (PI) + feed-forward (P)	Optimal + feedback (PI)	Adaptive + fuzzy controller	Multiple fuzzy controllers
Model	Black-box	Black-box	Queuing + transaction mix [134]	Queuing	Queuing + ARMA	Fuzzy models + black-box	Fuzzy models
Architecture	Distributed	Centralized	Hierarchical and distributed	Centralized	Centralized	Distributed	Centralized
Control objectives	Desired application performance and reduction of SLO violations	To regulate target performance of key-value store	Desired response time of multiple multi-tier applications	To minimize CPU capacity allocation per application, and maintain desired response time	To obtain optimal partitioning scheme that reduce round trip time	To maintain a target response time	Adaptive allocation of multi-objective resources and service differentiation
Reference input	Response time	99th percentile of read latency	Response times of applications	Response time	Round trip time	Response time	Response time, throughput, play rate
Control input	CPU capacity, memory allocation, MaxClient value	Average throughput per storage server	CPU capacity	Shared CPU capacity	CPU budget	CPU and memory allocation	CPU, memory, disk bandwidth
Monitoring metrics	CPU and memory usage, response time, throughput, number of Apache processes	R99, average throughput per server	Transaction mix, measured response time, CPU utilization	Arrival rate, response time, CPU usage	Arrival rate, average round trip time, CPU usage, average resident time	Average CPU and memory usage, mean response time	Response time, throughput, play rate
Ingredients	SP W P V	SP St Hy Ho	CP W Hy V	– W Hy V	CP W R V	SP W R V	– G P V
Workloads	Synthetic	Synthetic	Real: VDR traces [135]	Real: news portal traces	Real: FIFA	Synthetic	Synthetic
Applications used	httperf, autobensh tool	Voldmart	Customized RUBiS	CRIS tool	RUBiS	RUBiS, RUBBoS, Olio and httpmon	TPC-W, key-value store, video streaming
Environment	Real: Intel Quad Core i7 + Xen 3.3	Real: OpenStack (cluster of 11 nodes)	Real: HP ProLiant servers + Xen 2.6	Java simulation and real set-up on Linux server	Real: Linux machines + xen 3.0.3	Real: one machine with 32 cores + 56 GB memory + Xen	Real: two Dell servers + Xen
Compared with	Static scenario without reconfiguring resources	Not provided	Static, utilization only, nested control, feed-forward + utilization	Modified versions of [136, 137]	Schemes like equal shares and equal utilization	Not provided	Kalman filter, adaptive PI, ARMA

Table 6 Gain scheduling/switched approaches

	[58]	[40]	[52]	[65]	[51]	[41]	[66]
Types	GS (PID)	GS (SSF + LQR)	GS ($LPV - H_\infty$ + LQR)	Multi-model PI	Integral + fuzzy controller	SSF + pole placement	LPV controller
Models	–	Grey-box + least square regression	ARX + LPV-ARX	Black-box	Black-box + fuzzy model	State-space + grey-box	State-space LPV models
Architecture	Centralized	Distributed	–	Centralized	Centralized	Distributed	Centralized
Control objectives	To maintain a desired CPU load	To maintain web server performance (response time and throughput)	To dynamically control the aggregate CPU frequency to maintain target response time	To optimize share of CPU capacity among all VMs to meet desired response time	To maintain a desired CPU utilization of overall cluster	To maintain a desired response time	To maintain target response time for each application
Reference input	CPU load	Response time	Response time	Per application/VM average response time	CPU utilization	Response time	Response time
Control input	Number of VMs	Number of VMs, admission control	Aggregate CPU frequency	CPU capacity	Number of VMs	Number of VMs, admission control	Per VM CPU capacity share
Monitoring metric	CPU utilization, number of VMs	Utilization of VMs, throughput, measured response time, service rate	Number of jobs, queue length, throughput, mean response time	Per application response time	Response time, arrival rate, CPU utilization	Utilization of VMs, throughput, response time	Per application (arrival rate, effective service time, response time)
Ingredients	SP G R V	– W R Ho	CP G R Ho	CP G R V	SP W R Ho	– W R Ho	CP W R V
Workloads	Synthetic + real	Synthetic	Real: web traces [138]	–	Real: Nasa + FIFA	Synthetic	Real: web traces
Applications used	Httpmon	httperf	–	RUBiS	–	httperf	Customized Apach Jmeter, Micro-benchmarking web service
Environment	Real: Amazon	Real: Eucalyptus + KVM hypervisor	Simulation: customize CSIM simulation	Real: custom (three machines) + Xen 2.6	Simulation: CloudSim	Real: Eucalyptus + KVM hypervisor	Real: custom (eight VMs) + Xen 3.0
Compared with	Fixed gain PID	None provided	Linear models, queuing theory	Single model controllers	Fixed integral controller, Rightscale	None provided	None provided

Table 7 Intelligent approaches

	[47]	[76]	[77]	[97]	[70]	[71]	[67]
Types	Fuzzy type-2 controller	Fuzzy controller	Fuzzy controller	Fuzzy + optimization	Fuzzy controller	Fuzzy controllers	Fuzzy self-learning controller
Models	Fuzzy model	Zero-order Sugeno-type fuzzy model + clustering	Fuzzy model + classification and clustering	Fuzzy model + queuing	Fuzzy model + stochastic approximation algorithm [139]	Fuzzy model + t-digest [140]	Fuzzy + Q-learning
Architecture	Centralized	Distributed	Distributed	Centralized	Distributed	Distributed and centralized	Centralized
Control objectives	To maintain a target response time	To adjust per VM CPU capacity to meet performance goals	To adjust per VM CPU and disk IO to meet performance goals	To guarantee 90th percentile end-to-end delay in multi-tier systems	To maintain performance SLOs of an application	To maintain performance SLOs of an application	To maintain a desired response time
Reference input	95th percentile of response time	–	–	90th Percentile end-to-end delay	Response time, throughput	Response time and throughput	Response time
Control input	Number of VMs	CPU capacity	CPU capacity, disk IO bandwidth	Number of servers	CPU capacity allocation for each tier	CPU and memory allocation for each tier	Number of VMs
Monitoring metrics	Measured response time, arrival rate	CPU usage, throughput (reply rate), workload	CPU and disk IO usage, average query throughput, measured response time	End-to-end delay	Mean CPU usage per tier/VM, measured SLO	Mean CPU and memory usage per tier/VM, measured SLO	Arrival rate, measured response time, throughput
Ingredients	SP G R Ho	SP G P V	– DB P V	– W R Ho	CP W/D P V	CP W/DB/St P V	SP G R Ho
Workloads	Synthetic: adapted from [138]	Synthetic + real: FIFA	Synthetic + real: FIFA	Synthetic	Synthetic	Synthetic	Synthetic
Applications used	Jmeter	Java Pet store, httperf	TPC-H, RUBiS	Three tier application	RUBiS, Olio, Cassendra, RAIN, YCSB	RUBiS, RUBBoS, Olio, Cassendra, Redis	ElasticBench
Environment	Real: Microsoft Azure	Real: custom (16-CPU IBM x 336 based cluster) + VMWare ESX 3.0	Real: custom (Quad-core Intel Q6600) + Xen 3.3.1	Simulation: simulation (no information provided)	Real: custom (two Fujitsu Servers) + Xen 4.2	Real: custom (two Fujitsu Servers) + Xen 4.4	Real: Azure and OpenStack
Compared with	Over-/under-provisioning scenarios	Not provided	Peak load based allocation scheme	With/without optimization scenarios	DynaQoS [69], AutoControl [62]	APPLEware [45], FMPC [92]	Fuzzy controller without learning, Azure rule-based

The *Performance-based* as their name specifies refers to the category that include metrics related to the performance of the system, e.g., Response time, Throughput, etc. On the other hand, the *Capacity-based* relies on system provided metrics such as CPU utilisation, memory consumption, etc. The key benefits of using system provided metrics are the following: (1) they are directly obtained from monitoring API provided by CPs. Hence it does not require application level monitoring or efforts. (2) No runtime relation identification between application metric, e.g., *Response time*, is required. Hence it does not involve additional overhead at runtime. However, control solutions that rely only on system provided metrics assume that the specified utilisation threshold will always meet the performance expectation. On the other hand, control solutions that rely on performance metrics require an additional burden of monitoring system's performance. Furthermore, they do not consider system resource utilisation level.

The commonly used performance based metric includes the use of *Response time* as *Reference input*. This include the control solutions proposed in [31, 39–42, 45, 46, 52, 54, 59, 62–67]. The subset [39, 45, 59, 62, 64–66] of these control solutions targets the *CPs* perspective, where the reference input is actually the *Response time for each application* (*or each client*). In some other control solutions, the use of *Response time* is coupled with *Throughput* such as in [30, 68–71]. Apart from *Response time* and *Throughput*, the use of some other performance metrics is also observed. This include *Service time* for data oriented applications [43, 72], *Job progress* for scientific application [73] and *Read operation latency* for storage application [32, 74].

On the other hand, the capacity based *Reference input* depends on the type of elasticity. In the case of horizontal elasticity, CPU utilisation of the cluster is the common metric used for *Reference input* [44, 50, 51, 60, 61], whereas, in the case of vertical elasticity, the utilisation of individual resources is utilised as *Reference input*. For example, the control solutions in [55, 58] rely on the use of CPU utilisation, whereas *Memory consumption* was focused in [53] and both components were utilised in [48]. In last, very few approaches including the control solutions proposed in [28, 29, 75] use both performance based (i.e., *Response time*) and capacity based (i.e., *CPU utilisation*) as *Reference input*.

4.3 Control input

The choice of control input depends on the *Elasticity type*. Thus for the control solutions, where the *Elasticity type* is horizontal, the *Control input* is the *Cluster size* (*Number of servers*). This is evident from Tables 1, 2, 3, 4, 5, 6 and 7, with the only exception in the cases of [40, 41], where the control input consist of an additional parameter for disturbance rejection. In the case of vertical elasticity, the choice of *Control input* depends on the nature of application type, i.e., either CPU (or memory) sensitive. However, in terms of vertical elasticity, the existing literature lacks on providing the justification of the choice of *Control input*. It is evident from Tables 1, 2, 3, 4, 5, 6 and 7 that the *CPU allocation* is the most common choice considering their use as single *Control input* in the following control solutions [32, 34, 55–57, 59, 63–66, 68, 70, 73, 75, 76] and as combined with *Memory allocation* in [30, 42, 45, 71], combined with bandwidth in [62, 77] and combined with both memory and bandwidth in [30, 69]. In last, the control solutions proposed in [31, 53, 54] only use *Memory allocation* as *Control input*.

4.4 Controller

Generally, the various modelling approaches used in the design of control systems are categorized into the following three main classes [78–80]:

(i) *White-box* such models are used when it is possible to construct the model based on the prior knowledge and the availability of the physical insight about the system. White-box modelling derive mathematical models based on the use of first principles.
(ii) *Black-box* such models are data driven and no physical insights or prior knowledge of the system is required. Statistical methods are used to derive the model based on the measurement of data using well designed experiments, where the underlying system is considered as black-box.
(iii) *Grey-box* such models are hybrid in nature and are used in situations where some physical insights or prior knowledge about system is available, however, certain parameters are required to be derived from observed measurements.

The use of white-box modelling approaches in the context of cloud elasticity are rare. However, in certain few cases *Queuing Theory* and *State-space* modelling approaches are utilized. Thus, in the context of cloud elasticity, we found the following types of modelling techniques are used in the construction of control systems to address dynamic resource provisioning problem:

(i) *Queuing theory* the elastic system is considered as a queue so that different analysis can be performed such as prediction of queue length, average service rate, and average waiting time.
(ii) *Black-box*/*Grey-box* as earlier described, such methods are used when the detailed knowledge

of the target system is not available. Such approaches involve the construction of SID experiments, where well-designed system input signal is generated to record outputs of the system. Statistical techniques are then utilised to infer system input and output relationship.

(iii) *State-space* the target system using such an approach is characterised and represented using a set of state variables to express their dynamics. For detail description of these modelling techniques, please refer to [27].

The study of existing elasticity approaches implemented using control theory indicate, the use of various types of controllers. We have clustered them into the following four groups inspired by the controller types used in [19].

4.4.1 Classic

This family of control solutions contains the commonly used controller types that are comparatively simpler in nature. This category is further distributed into the following three types:

4.4.1.1 Fixed gain This subclass of controllers are referred to those control solutions, where the tuning/gain/model parameters are estimated off-line and then remains fixed at runtime. The most commonly used controller of this category is called Proportional–Integral–Derivative (PID) or its different variants such as Proportional–Integral (PI) or only Integral (I).

The authors of [44, 50, 60] focused on horizontal elasticity and proposed fixed gain control solutions for web applications. All these solutions aim to achieve a desire performance level using CPU utilisation as reference input. Lim et al. [50] used an Integral controller, where Gergin et al. [60] and Barna et al. [44] adopted PID based approach. In both of these papers, the design of the controller is similar but the adaptation is different. More specifically, Barna et al. [44] only considered one tier, whereas Gergin et al. [60] considered n-tiered transactional application and proposed a distributed architecture, i.e., using multiple controller in parallel but one for each tier. The multiple controllers do not have any interaction and synchronisation towards the achievement of end to end objective. Gergin et al. [60] decomposed the primary control objective to the objective of individual tiers based on the utilisation demand. Each control is then responsible for their tier's objective. The influence of tiers on each other are not considered in their approach. Furthermore, no relation identification between the desired performance and CPU utilisation is provided. On the other hand, Barna et al. [44] only considered the application business tier. However, they assumed that the intensity of the incoming workload would not saturate the other tiers. Lim et al. [50] in contrast focused on the application as a whole rather than tier specific. They further proposed a dynamic threshold based approach for the reference input (CPU utilisation in this case) to avoid oscillation. They used a dynamic range of CPU utilisation with lower and upper thresholds rather than the commonly used static target value as in the case of [60]. Barna et al. [44] also used the static value but avoided scaling decision within the range of $\pm 15\%$ of the target threshold.

Similar to Barna et al. [44], Ashraf et al. [48, 81] also focused on application business tier. However, there approach is different in the following ways: (1) they used a Proportional–Derivative (PD) controller without any performance model. (2) They have also used memory consumption as reference input instead of only CPU utilisation. Apart from above difference, they proposed the concept of shared hosting, where multiple web applications could be hosted to similar VM. However, it is not clear how to measure (and guarantee) the performance of web applications that share the same VM. Furthermore, no insight on the selection of lower and upper threshold for the memory and CPU utilisation is provided. Lastly, it is not clear how to model and quantify the relationship between utilisation metrics and application performance. Heo et al. [53] in contrast to the above mentioned approaches, focused on vertical elasticity, where they used the reallocation of memory and CPU resources at VM level to comply with the SLO requirements.

The control policy of [50] has been further utilised to maintain the performance (Response time) of storage tier of a multi-tier application in [61]. Whereas, a PID controller is proposed in [72] to adjust nodes of a cloud-based storage system (named Voldmart [82]) to maintain the desired service time. Both of these control solutions focused on Storage tier and used the data rebalancing component to distribute the data load among the available servers as a result of scaling decision. The approach of Lim et al. [61] is accompanied by a state machine, that is responsible to synchronise the actions of the Integral controller and the rebalancing component of the storage tier. However, no such details are provided in the case of [72]. In contrast, the authors of [43, 83] used a PI feedback controller for big data application. They focused to adjusts the computing nodes of a map reduce cluster to guarantee the desired service time of map reduce jobs. Their proposed approach guarantee the desired system performance. Furthermore, it is also associated with an admission control component that is responsible to stop disturbance from impacting the performance of the system. However, the response to disturbance is kept slow to minimise the number of scaling decision. Furthermore, they have

provided analysis of the control properties such as closed-loop analysis and 0% overshoot. The other approaches lack in this regard.

The gains of the fixed gain controllers can be estimated off-line either using a trial-and-error methods or performance model [50], application specific model [81] or using a standardised method such as ZieglerNichols and Root-locus. However, they remain fixed at runtime. The summarise details of the proposals reviewed in this category are provided in Table 1.

4.4.1.2 State space feedback (SSF) This class of controllers used state-space modelling approach to design the control solutions [27]. Li et al. [30] proposed an integrated three-layer automatic management approach. The three layers include resource management at VM, node and cloud level. Amongst these layers, the method at VM level is only relevant to the scope of the paper, where they proposed a MIMO based SSF controller responsible for determining VM resource requirements. They focused on vertical elasticity and considered multiple resources including the CPU power, memory and I/O allocation to achieve application SLO requirements consisting of Throughput and Response time. Their method uses an online model estimator to capture the relationships between application performance and resources at runtime. Similarly, Moulavi et al. [28] also used SSF, however for the horizontal scaling of distributed cloud storage systems' computational nodes.

Both of the above approaches use linear quadratic regulator (LQR) method to obtain the gains (Proportional and Integral) of their controllers. However, in [30], the gain is adaptive and may change at runtime, whereas the gain remains fixed in [28]. Moreover, Moulavi et al. [28] used an additional fuzzy controller, which is responsible for allowing or discarding the control decision considering the standard deviations of CPU loads. The purpose of the additional controller is to avoid unnecessary fluctuations. However, no description is provided for the design of fuzzy controller. Their approach also considers cost as one of the reference input. However, it is not clear how it influences the decision of the controller. A similar approach is utilised in [29] with a difference of using Throughput as one of the reference input rather than cost. However, no details are provided on the performance aspects of the proposed method. The details of the papers mentioned in this category are provided in Table 2.

4.4.1.3 Adaptive This class of controllers have the ability to estimate the model/control parameters at runtime thus adjusting itself to changes in the environment, e.g., the self-tuning PID controllers [17].

The authors of [32, 55, 62, 68, 75] focused on vertical elasticity, where they all used the dynamic readjustment of CPU allocation to maintain the target CPU utilisation of the system. Amongst these solutions, Zhu et al. [75] used a nested control design that comprise of two integral feedback loops. Their idea is to use CPU utilisation as a reference input in association with the target response time. However the threshold value for the CPU utilisation is not fixed and will be adjusted by the outer loop to maintain the target QoS goal (Response time). The inner loop is then responsible to maintain the variable CPU utilisation by adjusting the CPU allocation of a VM. Padala et al. [68] utilised a similar idea of multiple controllers but in hierarchical style. Furthermore, the focus is on multi-tiered applications that share the same physical node. The control solution in this case is responsible to adjust the CPU allocation of VMs that host individual tiers of multi-tier applications. They introduced a utilisation controller implemented using an adaptive Integral controller that runs at each VM and is responsible to maintains the target CPU utilisation by adjusting the CPU allocation, whereas an arbiter controller implemented using a fixed Integral controller is responsible to allocate the CPU share based on the requested CPU allocations by the utilisation controllers.

In the case of Zhu et al. [75], due to the variability of reference input of the inner loop, the target response time can be obtained using different CPU utilisation thus handling the uncertainty aspect between resource utilisation and performance. Moreover, this also improve cost efficiency. In contrast, in the case of Padala et al. [68], the target CPU utilisation for each VM is fixed that can be obtained at design time. Furthermore, no performance parameter at runtime is considered. However, their approach at node level resolves any conflicting decisions made by the utilisation controller in the case of resource contention situation, whereas no such scenarios are discussed in the case of Zhu et al. [75]. The approach of Padala et al. [68] is further enhanced in [62], where they not only considered the readjustment of CPU allocation but also included Disk I/O allocation. Moreover, they changed the adaptive Integral controller with an optimiser controller aiming to minimise a cost function comprise of performance and control cost while obtaining the values of required resource allocations for next interval.

Contrary to the integral controller based approaches of [68, 75], Kalyvianaki et al. [55] exploited the use of Kalman filter based feedback controller to adjust the CPU allocation for multi-tier applications. They proposed three Kalman filter based control solutions including a SISO, MIMO process noise covariance controller (PNNC) and an adaptive MIMO PNCC controller. The SISO is responsible for the CPU allocation of individual VMs that hosts a single application tier. MIMO PNNC is responsible for the

CPU allocation of all VMs of a multi-tier application. However, both their controllers (i.e., SISO and MIMO PNNC) are fixed, where the gain of the Kalman filter does not change at runtime. The adaptive MIMO PNNC is the adaptive counterpart of MIMO PNNC, where the gain of the controller is obtained at runtime. Their approach further utilised filters to track noise in the CPU utilisation. Moreover, the coupling between multi-tiers of the application is also considered, which was not discussed in other related approaches.

Spinner et al. [32], in contrast to the above mentioned approaches focused on a different perspective of CPU allocation, wherein they adjusted the number of virtual CPUs of VMs to meet target latency of an application. They used a queuing theory based on layered performance modelling approach in collaboration with a runtime model estimation component. Their performance model not only considers the application demand of the resources, but the demands of the virtual and physical resource as well. They also further considered the scheduling and contention delays caused by the rearrangement of VMs due to the over subscription or overloading of the physical host. However, the response of their resource controller to the disturbance in some scenario may be slow as their scaling decision is fixed where at each control interval, only one virtual CPU can be added or removed at a time.

All the above approaches were capacity based, where the decision making mechanism was based on resource utilisation except [75], where both CPU utilisation and response time were used as reference inputs. The control solutions that are only based on capacity based are inadequate to guarantee application performance because of their no consideration of performance aspects at runtime [54].

In contrast to these approaches, the authors of [31, 49, 54, 63, 73] included performance based metrics into their decision making mechanism. Amongst these, the target performance measurements were used to readjust, the memory size in [31] and CPU entitlement in [63, 73]. All these three methods only rely on performance based metrics that may not be adequate to ensure the desired quality of service, because the performance can be also disturbed by an internal bug [54]. The control solution of [31] is further extended in [54], where memory utilisation is considered as part of the decision additional to response time, making their methodology hybrid, which consider both aspects. Zhu et al. [49], in contrast utilised both CPU and memory allocation by proposing an adaptive version of the MIMO PI controller in collaboration with a reinforcement learning component. This approach, however, is different in two aspects from any other approaches mentioned in this section. Firstly, it does not directly control resources but rather change adaptive parameters of the cloud applications. Secondly, it aims to maximise the application specific benefits (QoS) within a pre-specified time limit and budget constraints. Similarly to this approach, the control solutions proposed in [84] also consider cost and efficiency constraint in order to find an optimal trade off between these two aspects. However, the control objective in this case is to obtain the optimal numbers of VMs rather than the readjustment of adaptive parameters of the application. Analogously, Mao et al. [85] also focused on maximizing efficiency with lowest cost possible under the deadline constraints. Furthermore, they consider different kinds of VM instances as well as heterogeneous deadlines. The nature of the application types in all three proposals are different, i.e., Zhu et al. [49] focused on adaptive systems of Jiang et al. [84] on web based systems, whereas Mao et al. [85] target on long running jobs.

In contrast to all of the above approaches in this section where the focus was on vertical elasticity, Ali-Eldin et al. [33] proposed an adaptive hybrid controller for horizontal scaling using queuing theory as a modelling technique. The queuing based model determines the total service capacity required per control interval while considering the arrival rate of the concurrent requests. The output of the controller is dependent on a gain parameter that is obtained at runtime using the change in demand on the past time unit and the necessary service capacity. The approach is independent of any performance controller. However, it does not consider how the application performance can be guarantee. Furthermore, the approach does not consider the impact of the delay caused by the start up of a VM. The same approach is applied in [86] considering scientific domain with an enhanced model and controller design. The enhanced model also considers buffer size, the delay caused due to VM start process, allocated capacity and changing request service rate of the VM. Furthermore, they not only predicted the future load changes but also considered the monitored load changes and delay caused due to VM start up. The summarise details of the proposals reviewed in this category as per the taxonomy attributes are provided in Tables 2 and 3.

4.4.2 Optimal

This class of controllers refers to all those control solutions that formulate and solve the cloud resource provisioning as an optimisation problem. There are two types of optimal methods utilised in the cloud elasticity domains. Their brief explanation and review of the proposals are provided as follows.

4.4.2.1 Model predictive controller (MPC) The MPC is based on a twofold concept [87]. Firstly, it uses an internal

dynamic model to predict future system behaviour, and optimises the forecast to generate the best decision at the current time. Secondly, it uses the previous moves of the controller to determine the best possible initial state of the system as the current move of the optimal control depends on it. For further details on MPC, refer to [87].

The readjustment problem of data centre capacity with a focus on energy saving perspective is addressed using an MPC based approach in Zhang et al. [35]. They included aspects like cluster reconfiguration cost (due to saving/loading/migration of systems' state), electricity price fluctuation (as rates vary at different time of the day in some countries, e.g., the USA). Their work, however, assumes that (1) all data centre machines have the same computational capabilities and (2) the instances of incoming workload shares similar characteristics, e.g., task length, resource requirements. Roy et al. [88] proposed a similar MPC based solution with the exception that they do not include electricity price fluctuation in their cost function but consider cluster reconfiguration cost, resource renting cost. Furthermore, they also consider SLA violation cost, which isn't the case in [35]. The work of Zhang et al. [35] is further extended in [36], where they considered heterogeneous hardware and workload behaviour. They approached the heterogeneity issue using MPC controllers coupled with a k-means clustering algorithm, which is used to cluster the tasks into various groups based on the identical characteristics, i.e., performance and resource requirements.

The MPC based approach in [37] decides a sequence of resource reservations actions for N steps rather than a single decision for next control input. The paper, however, lacks details about the results obtained. Whereas, Cerf et al. [38] focused on the idea of reducing the number of elasticity decisions for big data cloud systems using an MPC controller coupled with an event triggering mechanism. The event triggering mechanism serves as an additional layer to determine whether the MPC decision will be carried out or not. The mechanism is based on the optimal cost function rather than the state of the system or any form of control error mechanism.

In contrast to MPC approaches mentioned above, Lama et al. [45] proposed a distributed MIMO control solution to address vertical elasticity. They used multiple MPC to manage the allocation of resources (CPU and memory) to achieve the target performance requirements of the co-located multi-tier web applications deployed on shared computational nodes of a data center. Each MPC handles one application and controls the allocation of resources of all their respective VMs whilst considering each tier of the application deployed on a separate VM, which may reside in different computational nodes than other tiers' VMs. They used neural network based fuzzy models and considered variables of the local controller as well as the neighbour controller, which manages VMs of other applications that share the underlying physical resources.

A few other MPC based proposals include reconfiguration of storage system [89], resource management of multiple client classes in shared environment [90], performance optimisation using power control [91], and dynamic resource allocation using an integrated approach of fuzzy model and MPC [92].

4.4.2.2 Limited lookahead controller (LLC)

The LLC follows the similar concept as MPC, where the next action of the controller is determined using the projected behaviour of the system over a limited look-ahead horizon [93]. The key difference between MPC and LLC is that the former deals with the systems operating in continuous, whereas the latter deals in discrete input-output domains [34].

Kusic and Kandasamy [39] utilised an LLC mechanism for enterprise computing system by formulating the resource provisioning problem as a sequential optimisation problem. They approached the problem by considering multiple fixed three client classes, each with different QoS requirements. A different cluster is used to manage one client class focusing on reducing operating cost regarding switching cost and minimisation of energy consumption. The distinct feature of their approach is the consideration of provisioning decision risk as a factor and encoding it into the cost function while considering the variability of workload patterns. Each of their decision determines not only the number of machines in each cluster but also the operating processing frequencies associated with different pricing regarding power.

The same approach is adopted in [34] for virtualised environments with the following modifications. The controller decision concludes the allocation of CPU share of each VM for each cluster rather than specifying operating frequencies, the indication of active host machines and the share of workload to be assigned to each VM. Bai and Abdelwahed [59] demonstrated the use of artificial intelligence based search methods on a case study of processor power management to address the problem of computational overhead caused while using LLC.

4.4.3 Advance

This family of controllers clustered all those control solution methodologies that either combine multiple control method into one or have some notion of runtime adaptive behaviour. However, the adaptation mechanism is not based on runtime parameter estimation, which is described earlier for the *Adaptive* type in the *Classic* category. This family of controllers includes the following types.

4.4.3.1 Hybrid This set of controllers refers to all those control solutions that combine more than one controllers and all of them are active at the same time. The control solutions proposed in [42, 46, 57, 69] utilised multiple controllers in various capacity to directly maintain performance based metrics that identify the target quality of service. Amongst these, the control solution proposed in [42] consists of the use of three feedback controllers to dynamically allocate CPU, memory and application performance tuning respectively. All of the three controllers run in parallel albeit independent from each other and it is not clear how and why the decision of one controller does not have any effect on others. Analogously, Dutreilh et al. [94] proposed a generic framework, where multiple control policies can be used. Each policy is responsible to manage the scaling of a group of VMs (referred to as scaling point by the authors), which host one component of an application. Similar to [42], it is not clear, why the decisions of each control policy will not have an impact on each other. This approach however deals with horizontal elasticity in contrast to that of Dawoud et al. [42].

Farokhi et al. [46] also used two different controllers each for CPU and memory allocation of VMs. Their approach, however in contrast to [42, 94] consists of a fuzzy based coordination mechanism. This mechanism consider CPU and memory consumption as well application performance to determines the contribution of each of the controller to the final decision making. The use of fuzzy control solution in their approach also consider noise in measurement and uncertainty aspects during decision making, which rarely consider by cloud resource provisioning mechanism.

Xiong et al. [57], only focused on the CPU share allocation using a two-level hybrid control scheme to individual VMs hosting the different tiers of a N-tiered application. This approach consists of an inter-dependent application implemented using a PI feedback and resource partitioner implemented using optimal controller. The application controller is responsible to compute the CPU budget required for the application, whereas the resource controller is responsible to optimally distribute the computed CPU share among the individual VMs hosting the different tiers. This approach provide details about the resource partitioner controller, which solves an optimisation problem to derive optimal share for each VM using fixed CPU budget. However, no details are provided of how the total CPU budget for the application can be obtained, i.e., the details about the application controller are missing. Rao et al. [69] on the other hand also approached using a two-level approach. However, they considered multi-objectives and proposed a self tuning fuzzy controller (STFC) for each objective and a gain scheduler controller to compute the final decision by aggregating the decisions of all STFCs. This approach readjust three different components including CPU, memory and disk bandwidth. The gain scheduler is responsible to synchronise the decisions by the individual controller using the control errors of each STFC. Furthermore this approach provides resolution strategy in the case of conflicting decisions by the individual controllers and/or resource contention. The other methods discussed in this category lacks in these aspects.

In contrast to above approaches discussed in this category, the control solutions proposed in [56, 64, 74, 95] use a combination of feedback and feed-forward methods. The idea behind such merging is to exploit feed-forward to predict large spikes of the workload using a proactive method to make scaling decision in advance, whereas, the feedback approach can be exploited to manage the gradual changes and to rectify any modelling errors. Al-Shishtawy and Vlassov [74] used such an approach to perform horizontal scaling of cloud-based key-value stores. The scaling decision in their methodology will be either carried out with feed-forward or feedback method and that depends on the intensity of the workload. Their approach, however, is different in general than any other related horizontal approaches as they use average throughput per server as the control input in contrast to the commonly used number of servers.

Wang et al. [95] focused on vertical elasticity to adjust CPU allocation of virtual containers that hosts different tiers of an application. Their feed-forward method estimates optimal CPU utilisation level, whereas the feedback controller further tunes the utilisation target for individual containers that are maintained by the distributed utilisation controllers. In the case of [74], either feedback or feed-forward control executes at a time as their control interval is the same, whereas, in the case of [95], all controllers run at different control intervals. Their approach is further extended in [64] to manage the performance of multiple applications. The extension includes the consideration of hybrid controller of [95] as the application controller, which is responsible for calculating required CPU entitlement necessary to obtain the desired performance target. Furthermore, the addition of a node level controller (Integral) to manage the CPU entitlement of the respective VMs on a given node. Kjær et al. [56] combined a PI and a Proportional based feed-forward method. Their feed-forward approach however, is different as it uses an on-line performance model in contrast to other related hybrid approaches mentioned in this category which use off-line performance model.

4.4.3.2 Gain scheduled/switched controllers This family of control solutions refers to those methods, where multiple models/controllers/gain parameters are used

simultaneously. Such methodologies are accompanied with an associated reconfigurable/switching/gain scheduler layer to select the suitable model/controller/gains at runtime.

Grimaldi et al. [58] proposed a PID gain scheduling approach to dynamically adjust the number of VMs to maintain the desired CPU utilisation. The gain scheduler is based on the minimisation of a cost function to obtain optimised values of the controller gains in a particular operating region (characterised by different workload and timeslots) with an objective to reduce the control error close to zero. Certain details such as modelling aspects and how CPU utilisation is related to the end-user performance are missing. A linear parameter varying (LPV) modelling approach is followed in [40] to guarantee web server performance (Throughput and Response time) by dynamically adjusting the number of VMs. They utilised a gain scheduled LQR design approach, where the CPU utilisation of VMs is used as a scheduling parameter. In contrast, Qin and Wang [52] used $LPV - H_\infty$ controllers as their design approach for calculating the aggregate CPU frequency needed to maintain a target Response time, which was then used to compute the number of VMs. They used arrival rate of the workload and average service rate as the scheduling parameter for the characterisation of time-varying operating conditions. Similarly, Tanelli et al. [66] also used arrival rate and effective service rate per application as their scheduling variables in their proposed MIMO LPV approach. However, they focused on the dynamic allocation of CPU capacity to individual VMs that share the same physical system.

Patikirikorala et al. [65] proposed a multi-model switching control system to dynamically allocate CPU capacity to the VMs of a physical machine to maintain response time objectives of an individual application. Their approach distributes the overall system in two operating regions using a threshold level of Response time. A specific model is then designed to represent the behaviour of the system in each operating region. Their approach uses multiple fixed PI feedback controllers, which on runtime change based on the operating region using if-else based switching mechanism. In contrast, Saikrishna and Pasumarthy [41] used ten distinct operating regions and arrival rate as the switching signal. Similarly, our early work in [51] is based on the use of multiple controllers to dynamically adjust the number of VMs to guarantee application performance. The distribution of overall system among different operating region is based on the various intensity levels of incoming workload, whereas the selection of suitable controller is realised at runtime using a fuzzy control system based switching mechanism. The switching mechanism consists of attributes like workload intensity, application performance and resource utilisation level. Morais et al. [96] in contrast, combine multiple predictors, where they switch amongst them in order to find best settings for the resources to meet the target utilization level. They however, do not consider any target performance and assume that the target utilization level meet the purpose. Their method, in contrast to other approaches mentioned in this category, is also accompanied with a reactive strategy, where the scaling actions shall be taken based on predefined utilization threshold.

4.4.4 Intelligent

This set of controllers is based on uniting the underlying knowledge of the system in the form of ontology or rules to reason about the behaviour of the system, e.g., knowledge-based fuzzy control and neural network based control solutions.

The authors of [47, 67, 97] proposed horizontal elasticity solution for Internet based systems using fuzzy systems. They all have used performance based metrics to make scaling decisions. Jamshidi et al. [47] highlighted the lack of handling uncertainty related issues in existing auto-scaling approaches and the static scaling behaviour of commercially available rule based systems. They proposed the idea of qualitative elasticity rules using a fuzzy control system to the aforementioned issues. The inputs to their method consist of *Arrival rate* and *Response time*, whereas the output is the number of VMs to be added or removed. Their approach utilised the knowledge of domain experts to design fuzzy rules at design time. These fuzzy rules are then responsible to make scaling decisions at runtime. The key problems of this approach is their reliance on domain knowledge, which may not always available. Furthermore, the output (number of VMs) of their approach are a predefined fixed range. Their approach is further extended in [67], where they have used fuzzy Q-Learning to learn the best elastic policies (fuzzy rules) at runtime to cope with the issue mentioned above.

In both of the above mentioned approaches, they have consider application as a whole rather than handling the complexities of the multiple tiers of the application. In contrast, Lama et al. [97] focused on multi-tier applications to guarantee the 90th percentile end-to-end delay. This approach utilised an optimal approach to determine the optimal number of servers needed for a multi-tier application and integrate a self tuning fuzzy controller to compensate the delay caused due to the addition of new server. This approach however, assume that all the servers are homogeneous. Furthermore, in contrast to [47, 67], they also consider resource utilisation in decision making in accompanying to their performance measurements.

In contrast to the approaches mentioned above, the control solutions proposed in [70, 71, 76, 77] focused on vertical elasticity. Xu et al. [76] used two fuzzy based

methods, i.e., modelling and prediction to dynamically estimate the CPU capacity of a VM needed by an application. Their modelling approach builds a fuzzy model at runtime by directly monitoring/learning the relationship between application workload, resource usage and performance. Whereas, their adaptive prediction technique only considers the observations of resource usage to estimate future resources. Their approach is generic in nature and can be utilised for different type of applications. In contrast, Ref. [70] focused on multi-tier application to dynamically allocate CPU share of VMs of each tier. This approach is further extended in [71] to also take into account the memory readjustment in collaboration to CPU capacity. Moreover, they also provided a cascade based coordination mechanism at the time of scaling decision to consider the joint effect of both kinds of resources. Such coordination mechanism has not been considered by many authors for the same problem, e.g., [42, 92, 98]. Wang et al. [77], in contrast focused on data based system to readjustment the CPU capacity and disk IO bandwidth using an adaptive fuzzy modelling approach. Some other examples of fuzzy approaches include neural fuzzy control [99], fuzzy logic based feedback controller [100], fuzzy model coupled with a performance prediction model [101] and multi-agent fuzzy control [102].

4.5 Architecture

The analysis of implementation pattern of each reviewed proposal presented in Tables 1, 2, 3, 4, 5, 6 and 7 indicate the use of following patterns:

- *Centralised* the control system following this architecture is implemented as one unit, which is responsible for managing the control objective from a central place, e.g., at a global system level. It is evident from Tables 1, 2, 3, 4, 5, 6 and 7 that the majority of the control solutions are *Centralized*. A solution can be centralized at one of the following three levels, i.e., Application, Node or Cloud. The solutions that focused on horizontal elasticity from the SP perspective are centralized at Application level (e.g., [43, 44, 48, 50, 61, 72]), whereas the control solutions that cater CPs perspective runs centrally at Cloud level, where they could be responsible for different applications (e.g., [33, 48, 86]). The application level control solutions can be executed outside of the cloud environment and therefore they can control interactions with multiple control. The centralizes solution at node level are those, where the control solutions are responsible for the resource management of VMs running at that computational node (e.g., [49, 53, 54]).
- *Distributed* the control systems adopting distributed pattern implement at sub system level. Such control methods are usually responsible for achieving the control objective at sub system level. It can be seen from Tables 1, 2, 3, 4, 5, 6 and 7 that such an approach is mostly common in the cases of vertical elasticity, where the implementation of the control solutions are proposes at each VM of the cluster, (e.g., [30, 62, 73]). In the case of horizontal elasticity, there are few approaches including [40, 41, 60], where sub controllers are responsible to handle resource management task at per tier (or objective) level.
- *Hierarchical* the control system in this case is implemented at two different levels, i.e., lower and upper level. At the lower level, the distributed controllers manage a sub-system, whereas, at the upper level, another controller mediate distributed controllers to achieve the control objective at the global scale. This category only include the following control solutions [34, 64, 68].
- *Cascade* using such an approach, multiple controllers work simultaneously in a way, where the decision of one control solution becomes input for the next one, (e.g., [28, 75]).

5 Discussion, issues and challenges

The feedback control solutions that follow the fixed gain design principle such as [44, 48, 50, 53, 60, 61, 72] in general work well for systems that are subject to stable or slowly varying workload conditions [17]. However, due to the lack of adaptive behaviour at runtime, the performance suffers in scenarios where the operating conditions change quickly or when the environmental conditions and configuration spaces are too wide to be explored effectively [18].

The lack of adaptivity issue has been addressed by incorporating runtime adaptation mechanism using online learning algorithms such as the use of linear regression [31], optimisation [62], Kalman filter [55] and reinforcement learning [49]. In general, such adaptive control methodologies have the ability to modify themselves to the changing behaviour in the system environment that make them suitable for systems with changing workload conditions. However, such methodologies are criticised for the associated additional computational cost caused due to the online learning [7], their associated risk of reducing the quality assurance of the resulted system, and the impossibility of deriving a convergence or stability proof [18]. Moreover, they are unable to cope with sudden changes in the workloads.

Similarly, the optimal methodologies such as MPC and LLC based approaches are also blamed for their computationally expensive nature due to solving complex optimization models [103] despite their ability of producing optimal results. For example, Ali-Eldin et al. [33] reported that for the LLC based solution proposed in [34], it takes half an hour for computing the elasticity decision for a system consist of 60 virtual machines hosted by 15 physical servers. The computational time for such solutions demonstrate exponential increase with the increase in the problem size, i.e., the number of computational servers.

The hybrid approaches integrates multiple controllers to achieve efficient control over cloud resources. The integration includes controllers either of same kind (e.g., [42, 46]) or different kinds (e.g., [57, 69]). The proposals reviewed in this category hint on the following three different types of integration. Firstly, the multiple controllers act in parallel, e.g., [42, 46]. Secondly, the different controllers used are executing at different levels, e.g., [57, 69]. Lastly, a combination of feed-forward and feedback controllers are utilized, e.g., [56, 64, 74, 95].

The hybrid approaches where the controllers run in parallel require a synchronization method to determine the contribution of each controller into the final objective. For example, consider the case of vertical elasticity, where individual feedback controllers are provided to handle memory and CPU requirements. In such a solution, the role of the decisions taken by individual controller shall be synchronized and coordinated, otherwise it may result in unpredictable system behaviour [46]. However, establishing better coordination mechanism is challenging as it requires to determine the relation between application performance and the variations in the combination of different resources.

The hybrid approaches [57, 69] running at different levels have some sort of coordination. In such approaches, the multiple controllers at first level aim to maintain individual objectives (e.g., the resource requirements of individual tiers of the application), where the next level resolves any conflicting decision made by individual controllers to deduce the final output. The hybrid scheme integrating the combination of feed-forward and feedback methods are effective, where the feed-forward controller follows a predictive approach that takes scaling decisions for a longer time in advance, whereas the feedback method is responsible for making gradual changes in a reactive style. However, the performance and accuracy of the proposals are more dependent on the choice of the type of controllers used as feed-forward and feedback methods. For example, Al-Shishtawy and Vlassov [74] used MPC based feed-forward control solution and a fixed-gain PI based feedback control method. MPC based approaches as earlier mentioned are very accurate but computationally expensive, whereas fixed gain PI feedback controller suffers from lack of adaptivity at runtime.

The gain scheduled (switched) controllers like the hybrid approaches also consist of multiple controllers, but only one controller (model) is active at a time. The gain scheduled (switched) controllers have the ability to adapt themselves to the changing environment at runtime using the pre-configured repository of multiple controllers (or models), where each works well in a different operating region. Such a controller helps to achieve certain level of adaptivity at runtime without learning at runtime, e.g., [41, 51, 65]. However, such controllers require a large amount of work to be done at design time to identify and partition the system among the various operating regions. Furthermore at runtime, the handling of any situation that was unseen at design time will be uncertain. Lastly, they are also criticized more often for their associated unwanted behaviour, termed as bumpy transition, that could lead the system to an oscillatory state [27, 104, 105]. On the other hand, knowledge-based control solutions utilizing machine learning [67, 106, 107] or neural networks [99, 108] provide high levels of flexibility and adaptivity. However, such flexibility and adaptivity come at the cost of long training delays, poor scalability, slower convergence rate, and the impossibility of deriving stability proof [7, 18, 103, 109].

It is concluded from the above discussion that the different elastic controllers due to their underlying implementation techniques have different pros and cons, hence there is no best solution and the choice of selecting suitable approaches depends on the requirements [18]. Furthermore, irrespective of the considerably wide range of control theoretical approaches to implement cloud elasticity, there are still various issues and challenges that have not received much attention. Based on our analysis, we list the following important open issues and challenges that need to be further addressed:

(i) *Heterogeneity* majority of the existing horizontal auto-scaling systems consider hiring of VMs with same computational resources. Such consideration eases the design and implementation of the control systems. However, renting homogeneous servers is not always pragmatic. Therefore, further research is required for the development of control systems that consider acquisition and release of servers having different computational capabilities. This creates challenges for building efficient, accurate and robust performance models that consider heterogeneous computational capabilities.

(ii) *Vertical elasticity* the commercial CPs (such as Amazon, Microsoft, etc.) only provide horizontal elasticity features. However, vertical elasticity facilitate users to control their rented resources at the fine-grained level. It allows to dynamically increase (and decrease) certain components, e.g., CPU and memory. It is evident from Sect. 4 that there are many academic control solutions that handle vertical scaling. Some of such approaches either rely on the dynamic adjustments of one computational resource or independently control different resources using multiple independent controllers. However, considering the uncertain nature of applications, it is difficult to predict at design time the dependent resource components and therefore, relying on the dynamic adjustments of one computational resource is not always pragmatic. On the other hand, the existing approaches handle multiple resources, lack coordination mechanisms that can address the collaborative behaviour of the readjustment decisions made by different controllers runs in parallel to each other. Such mechanisms are needed to avoid unnecessary adjustments, avoid oscillations, resolve conflicting decisions and avoid over (and under) provisioning.

(iii) *Hybrid* (*feed-forward and feedback*) *solutions* the hybrid solutions that integrate feed-forward and feedback methods are effective as they can obtain the benefits of both style of auto-scaling decisions, i.e., predictive and reactive. The feed-forward method in the hybrid approach anticipate system's demand in advance and takes scaling decision to make sure the virtual machines are ready at the right time thus avoiding the additional delay caused by the virtual machine start-up in the case of reactive auto-scaling. On the other hand, the feedback method handles the small variations in reactive style to handle any unpredictable situation. It is evident from Sect. 4.4.3.1 that there are very few research works [56, 64, 74, 95] that address cloud elasticity problem using such an approach. Furthermore, most of these approaches focused only on vertical elasticity. Therefore, considering the benefits of such hybrid approaches, further research is required in this direction.

(iv) *Interoperability* the cloud is perceived as the repository of unlimited computational resources. However, in reality, every CP has a limited set of computational resources. Therefore, the application providers may need to hire resources from different CPs. However, reliance on resources from different CPs raised a number of challenges related to interoperability for the application providers. Some of these challenges in the prospect of elasticity (including the interaction between control solution and auto-scaling APIs of CPs, the delay, the design of accurate performance model, etc.) have not been addressed in the existing control solutions. Therefore further research is required for the consideration of interoperability with respect to addressing elasticity.

(v) *Oscillation* the design of control solutions for auto-scaling requires careful attention and detailed evaluation because badly designed controller may result in oscillation and instability [24]. The switched controllers in particular are criticized for the phenomenon like bumpy transitions that could leads system to an oscillatory state [27, 104, 105], where cloud resources are acquired and released periodically. The occurrences of bumpy transitions may be due to an inappropriate switching or some larger changes in the system state. The existing research works lack on providing an explanation on how a proposed method deals with such undesirable oscillatory behaviour.

(vi) *Resource usage analysis* over-provisioning is used to avoid performance violation considering peak workload scenarios [110, 111]. However, this results in the wastage of resources. It is therefore undesirable and should be avoided. In the existing research works, the authors mostly provide the evaluation of the proposed methodologies in terms of achieving the performance objective. However, an explanation or comparative analysis of the methodology in the prospect of minimising the computational resources is mostly missing. Therefore, further research on cloud elasticity shall consider to conduct comparative cost analysis against state of the art approaches in accordance to obtaining the performance objectives.

(vii) *Evaluation and benchmarking* the performance of a control methodology is sensitive to the changes in workload. Therefore, extensive evaluation of control solutions are

necessary. However, majority of the existing research works have been evaluated only using less than three workload scenarios [112]. Moreover, as is evident from Tables 1, 2, 3, 4, 5, 6 and 7, the majority of the papers also lack on providing details of comparative evaluation. Furthermore, the unavailability of benchmark frameworks makes it difficult to compare and evaluate the related approaches. A recent development in this regard is the performance evaluation framework proposed in [112]. However, this framework is generic and there is a stronger need for specific control-theoretical benchmarks that have the ability to facilitate analysis of control solution specific characteristics, e.g., SASO (Stability, Accuracy, Short settling and Overshoot) properties as discussed in [2].

(viii) *Computational overhead analysis* most of the control methodologies are adaptive in nature as they have the ability to dynamically adapt to the changing environments. Such methodologies provide flexibility and adaptivity. However, they are also criticized for their long training delays and slower convergence rate [7, 18, 103, 109] because they are required to learn the system behaviour at runtime. The existing research works that have utilised such methods are often lack on providing details regarding the associated computational overhead. Therefore, further research works proposing such adaptive methods shall consider providing details on overhead analysis of their respective methods.

(ix) *Uncertainty* the deployed application over cloud environment automatically inherits the uncertainty related challenges associated with the cloud environment [113]. Hence the underlying elastic method, which is responsible for the resource management of the application, has to deal with these challenges. However, handling uncertainty aspects in the existing elasticity research has not yet received much attention [114]. Some examples of such uncertainty behaviour include impreciseness in domain knowledge, inaccuracy in monitoring information, delays caused due to actuator operation, failure of a VM, noise in input data, the unpredictability of workload and inaccuracies in performance model [114–116]. The existing research works on cloud elasticity in general has not paid much attention to consider such uncertainty aspects, while designing auto-scaling system [114, 115]. Therefore, further research works in this direction is needed.

(x) *Scalability* it is observed, during our analysis that most of the control methods are either designed or tested for web applications. Furthermore, the evaluation and analysis by the authors of respective approaches are performed at small scale, i.e., using a workload spanning of hours/days [112] or fewer number of VMs. The experimentation and description on the suitability of control solutions at larger scale considering realistic enterprise level web applications is missing.

6 Conclusion

With the increasing popularity of cloud computing in recent years, the quest for better elasticity methods has received a lot of attention. However, determining the right amount of computational resources needed at runtime is a challenging task. Over the years, the use of control theory has been stood out as one of the very few main techniques to implement cloud elasticity. In this paper, we survey the cloud elasticity literature by focusing on control theoretical approaches to provide a detailed review of the literature, using a proposed taxonomy consisting of characteristics belonging to both domains, i.e., control theory and cloud elasticity. Finally, we highlight some open issues and challenges that have not received sufficient attention in the literature.

7 Summarized results

The details of all the proposals reviewed in Sect. 4 are clustered with respect to the type of controller and presented in Tables 1, 2, 3, 4, 5, 6 and 7. The extracted information from the reviewed proposals are presented as per the attributes of the taxonomy explained in Sect. 3. The rows of the tables represent the characteristics of the taxonomy where each column indicate a different approach.

Note the *Ingredients* attribute of the taxonomy combine four characteristics (including Provider, Application type, Trigger and Elasticity type). This can be seen from Fig. 2. In light of this, each cell of the *Ingredients* row present respective value for each of the four characteristics mentioned above. The possible values (and their corresponding acronyms) for each characteristic is as follow (this can also be seen from Fig. 2)

– *Provider* the possible values include CP and SP.

- *Application type* the possible values include Generic (G), Web (W), Scientific (Sc), Storage (St) and Database (Db).
- *Trigger* the possible values include Reactive (R), Predictive (P) and Hybrid (H).
- *Elasticity type* the possible values include Horizontal (Ho) and Vertical (V).

References

1. Galante, G., De Bona, L.C.E.: A survey on cloud computing elasticity. In: Proceedings—2012 IEEE/ACM 5th International Conference on Utility and Cloud Computing, UCC 2012, pp. 263–270 (2012)
2. Abdelzaher, T., Diao, Y., Hellerstein, J.L., Lu, C., Zhu, X.:. Introduction to control theory and its application to computing systems. In: Performance Modeling and Engineering, pp. 185–215. Springer, New York (2008)
3. Abdelzaher, T.F., Shin, K.G., Bhatti, N.: Performance guarantees for web server end-systems: a control-theoretical approach. IEEE Trans. Parallel Distrib. Syst. **13**(1), 80–96 (2002)
4. Gandhi, N., Tilbury, D.M., Diao, Y., Hellerstein, J., Parekh, S.: MIMO control of an Apache web server: modeling and controller design. In: Proceedings of the 2002 American Control Conference, 2002, vol. 6, pp. 4922–4927. IEEE (2002)
5. Parekh, S., Rose, K., Diao, Y., Chang, V., Hellerstein, J., Lightstone, S., Huras, M.: Throttling utilities in the IBM DB2 universal database server. In: Proceedings of the 2004 American Control Conference, 2004, vol. 3, pp. 1986–1991. IEEE (2004)
6. Karlsson, M., Karamanolis, C., Zhu, X.: Triage: performance differentiation for storage systems using adaptive control. ACM Trans. Storage **1**(4), 457–480 (2005)
7. Lorido-Botran, T., Miguel-Alonso, J., Lozano, J.A.: A review of auto-scaling techniques for elastic applications in cloud environments. J. Grid Comput. **12**(4), 559–592 (2014)
8. Naskos, A., Gounaris, A., Sioutas, S.: Cloud elasticity: a survey. In: Algorithmic Aspects of Cloud Computing, pp. 151–167. Springer, Cham (2016)
9. Coutinho, E.F., de Carvalho Sousa, F.R., Rego, P.A.L., Gomes, D.G., de Souza, J.N.: Elasticity in cloud computing: a survey. Ann. Telecommun. **70**, 1–21 (2015)
10. Al-Dhuraibi, Y., Paraiso, F., Djarallah, N., Merle, P.: Elasticity in cloud computing: state of the art and research challenges. IEEE Trans. Serv. Comput. **11**, 430–447 (2017)
11. Singh, S., Chana, I.: QoS-aware autonomic resource management in cloud computing: a systematic review. ACM Comput. Surv. (CSUR) **48**(3), 42 (2015)
12. Jennings, B., Stadler, R.: Resource management in clouds: survey and research challenges. J. Netw. Syst. Manag. **23**(3), 567–619 (2015)
13. Mustafa, S., Nazir, B., Hayat, A., Madani, S.A., et al.: Resource management in cloud computing: taxonomy, prospects, and challenges. Comput. Electr. Eng. **47**, 186–203 (2015)
14. Manvi, S.S., Shyam, G.K.: Resource management for Infrastructure as a Service (IaaS) in cloud computing: a survey. J. Netw. Comput. Appl. https://doi.org/10.1016/j.jnca.2013.10.004 (2014)
15. Singh, S., Chana, I.: Resource provisioning and scheduling in clouds: QoS perspective. J. Supercomput. **72**, 926–960 (2016)
16. Yfoulis, C.A., Gounaris, A.: Honoring SLAs on cloud computing services: a control perspective. In: 2009 European Control Conference (ECC), pp. 184–189 (2009)
17. Patikirikorala, T., Colman, A.: Feedback controllers in the cloud. In: Proceedings of APSEC (2010)
18. Gambi, A., Toffetti, G., Pezze, M.: Assurance of self-adaptive controllers for the cloud. In: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). LNCS, vol. 7740, pp. 311–339. Springer, Berlin (2013)
19. Patikirikorala, T., Colman, A., Han, J., Wang, L.: A systematic survey on the design of self-adaptive software systems using control engineering approaches. In: Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, pp. 33–42. IEEE Press (2012)
20. Shoaib, Y., Das, O.: Performance-oriented cloud provisioning: taxonomy and survey (November):1–14 (2014). arXiv:1411.5077
21. Najjar, A., Serpaggi, X., Gravier, C., Boissier, O.: Survey of elasticity management solutions in cloud computing. In: Continued Rise of the Cloud, pp. 235–263. Springer, London (2014)
22. Hummaida, A.R., Paton, N.W., Sakellariou, R.: Adaptation in cloud resource configuration: a survey. J. Cloud Comput. **5**(1), 1–16 (2016)
23. Coutinho, F.E., De Carvalho Sousa, F.R., Rego, P.A.L., Gomes, D.G., De Souza, J.N.: Elasticity in cloud computing: a survey. Ann. Telecommun. **70**, 289–309 (2015)
24. Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A., Padala, P., Shin, K.: What does control theory bring to systems research? ACM SIGOPS Oper. Syst. Rev. **43**(1), 62–69 (2009)
25. 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), pp. 716–723. IEEE (2011)
26. Patikirikorala, T., Wang, L., Colman, A., Han, J.: Hammerstein–Wiener nonlinear model based predictive control for relative QoS performance and resource management of software systems. Control Eng. Pract. **20**(1), 49–61 (2012)
27. Hellerstein, J.L., Diao, Y., Parekh, S., Tilbury, D.M.: Feedback Control of Computing Systems. Wiley (2004). https://doi.org/10.1002/047166880X
28. Moulavi, M.A., Al-Shishtawy, A., Vlassov, V.: State-space feedback control for elastic distributed storage in a cloud environment. In: The 8th International Conference on Autonomic and Autonomous Systems (ICAS 2012), pp. 589–596 (2012)
29. Saikrishna, P.S., Pasumarthy, R., Kruthika, H.A.: Stability analysis of cloud computing systems under uncertain time delays. In: Proceedings of the 21st International Symposium on Mathematical Theory of Networks and Systems, Groningen, The Netherlands (2014)
30. Li, Q., Hao, Q.F., Xiao, L.M., Li, Z.J.: An integrated approach to automatic management of virtualized resources in cloud environments. Comput. J. **54**(6), 905–919 (2011)
31. Farokhi, S., Jamshidi, P., Lucanin, D., Brandic, I.: Performance-based vertical memory elasticity. In: Proceedings—IEEE International Conference on Autonomic Computing, ICAC 2015, pp. 151–152 (2015)
32. Spinner, S., Kounev, S., Zhu, X., Lu, L., Uysal, M., Holler, A., Griffith, R.: Runtime vertical scaling of virtualized applications via online model estimation. In: 2014 IEEE Eighth International

Conference on Self-Adaptive and Self-Organizing Systems, pp. 157–166. IEEE (2014)
33. Ali-Eldin, A., Tordsson, J., Elmroth, E.: An adaptive hybrid elasticity controller for cloud infrastructures. In: 2012 IEEE Network Operations and Management Symposium (NOMS), pp. 204–212. IEEE (2012)
34. Kusic, D., Kephart, J.O., Hanson, J.E., Kandasamy, N., Jiang, G.: Power and performance management of virtualized computing environments via lookahead control. Clust. Comput. **12**(1), 1–15 (2009)
35. Zhang, Q., Zhani, M.F., Zhang, S., Zhu, Q., Boutaba, R., Hellerstein, J.L.: Dynamic energy-aware capacity provisioning for cloud computing environments. In: Proceedings of the 9th International Conference on Autonomic Computing, pp. 145–154 (2012)
36. Zhang, Q., Zhani, M.F., Boutaba, R., Hellerstein, J.L.: Dynamic heterogeneity-aware resource provisioning in the cloud. IEEE Trans. Cloud Comput. **2**(1), 14–28 (2015)
37. Ghanbari, H., Simmons, B., Litoiu, M., Barna, C., Iszlai, G.: Optimal autoscaling in a IaaS cloud. In: Proceedings of the 9th International Conference on Autonomic Computing—ICAC '12, vol. 2(i), p. 173 (2012)
38. Cerf, S., Berekmeri, M., Robu, B., Marchand, N., Bouchenak, S.: Cost function based event triggered Model Predictive Controllers application to Big Data Cloud services. In: 2016 IEEE 55th Conference on Decision and Control (CDC), pp. 1657–1662. IEEE (2016)
39. Kusic, D., Kandasamy, N.: Risk-aware limited lookahead control for dynamic resource provisioning in enterprise computing systems. Clust. Comput. **10**(4), 395–408 (2007)
40. Saikrishna, P.S., Pasumarthy, R., Bhatt, N.P.: Identification and multivariable gain-scheduling control for cloud computing systems. IEEE Trans. Control Syst. Technol. **25**, 792–807 (2016)
41. Saikrishna, P.S., Pasumarthy, R.: Multi-objective switching controller for cloud computing systems. Control Eng. Pract. **57**, 72–83 (2016)
42. Dawoud, W., Takouna, I., Meinel, C.: Elastic VM for cloud resources provisioning optimization. In: Communications in Computer and Information Science, CCIS, vol. 190, pp. 431–445 (2011)
43. Berekmeri, M., Serrano, D., Bouchenak, S., Marchand, N., Robu, B.: A control approach for performance of big data systems. IFAC Proc. Vol. **47**(3), 152–157 (2014)
44. Barna, C., Fokaefs, M., Litoiu, M., Shtern, M., Wigglesworth, J.: Cloud adaptation with control theory in industrial clouds. In: 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW), pp. 231–238. IEEE (2016)
45. Lama, P., Guo, Y., Jiang, C., Zhou, X.: Autonomic performance and power control for co-located web applications in virtualized datacenters. IEEE Trans. Parallel Distrib. Syst. **27**(5), 1289–1302 (2016)
46. Farokhi, S., Lakew, E.B., Klein, C., Brandic, I., Elmroth, E.: Coordinating CPU and memory elasticity controllers to meet service response time constraints. In: 2015 International Conference on Cloud and Autonomic Computing (ICCAC), pp. 69–80. IEEE (2015)
47. 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, pp. 95–104. ACM (2014)
48. Ashraf, A., Byholm, B., Porres, I.: CRAMP: Cost-efficient Resource Allocation for Multiple web applications with Proactive scaling. In: CloudCom 2012—Proceedings: 2012 4th IEEE International Conference on Cloud Computing Technology and Science, pp. 581–586 (2012)
49. Zhu, Q., Agrawal, G.: Resource provisioning with budget constraints for adaptive applications in cloud environments. IEEE Trans. Serv. Comput. **5**(4), 497–511 (2012)
50. Lim, H.C., Babu, S., Chase, J.S., Parekh, S.S.: Automated control in cloud computing: challenges and opportunities. In: Proceedings of the 1st Workshop on Automated Control for Datacenters and Clouds, pp. 13–18. ACM (2009)
51. Ullah, A., Li, J., Hussain, A., Yang, E.: Towards a biologically inspired soft switching approach for cloud resource provisioning. Cogn. Comput. (2016). https://doi.org/10.1007/s12559-016-9391-y
52. Qin, W., Wang, Q.: Modeling and control design for performance management of web servers via an LPV approach. IEEE Trans. Control Syst. Technol. **15**(2), 259–275 (2007)
53. Heo, J., Zhu, X., Padala, P., Wang, Z.: Memory overbooking and dynamic control of Xen virtual machines in consolidated environments. In: 2009 IFIP/IEEE International Symposium on Integrated Network Management, IM 2009, pp. 630–637 (2009)
54. Farokhi, S., Jamshidi, P., Lakew, E.B., Brandic, I., Elmroth, E.: A hybrid cloud controller for vertical memory elasticity: a control-theoretic approach. Future Gener. Comput. Syst. **65**, 57–72 (2016)
55. Kalyvianaki, E., Charalambous, T., Hand, S.: Self-adaptive and self-configured CPU resource provisioning for virtualized servers using Kalman filters. In: Proceedings of the 6th International Conference on Autonomic Computing, pp. 117–126. ACM (2009)
56. Kjær, M.A., Kihl, M., Robertsson, A.: Resource allocation and disturbance rejection in web servers using SLAs and virtualized servers. IEEE Trans. Netw. Serv. Manag. **6**(4), 226–239 (2009)
57. Xiong, P., Wang, Z., Malkowski, S., Wang, Q., Jayasinghe, D., Pu, C.: Economical and robust provisioning of n-tier cloud workloads: a multi-level control approach. In: 2011 31st International Conference on Distributed Computing Systems (ICDCS), pp. 571–580. IEEE (2011)
58. Grimaldi, D., Persico, V., Pescapé, A., Salvi, A., Santini, S.: A feedback-control approach for resource management in public clouds. In: 2015 IEEE Global Communications Conference (GLOBECOM), pp. 1–7. IEEE (2015)
59. Bai, J., Abdelwahed, S.: Efficient algorithms for performance management of computing systems. In: Fourth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks FeBID. IEEE (2009)
60. Gergin, I., Simmons, B., Litoiu, M.: A decentralized autonomic architecture for performance control in the cloud. In: Proceedings—2014 IEEE International Conference on Cloud Engineering, IC2E 2014, pp. 574–579 (2014)
61. Lim, H.C., Babu, S., Chase, J.S.: Automated control for elastic storage. In: Proceedings of the 7th International Conference on Autonomic Computing, pp. 1–10. ACM (2010)
62. Padala, P., Hou, K.-Y., Shin, K.G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A.: Automated control of multiple virtualized resources. In: EuroSys'09, pp. 13–26 (2009)
63. Liu, X., Zhu, X., Singhal, S., Arlitt, M.: Adaptive entitlement control of resource containers on shared servers. In: 2005 9th IFIP/IEEE International Symposium on Integrated Network Management, IM 2005, vol. 2005, pp. 163–176 (2005)
64. Wang, Z., Chen, Y., Gmach, D., Singhal, S., Watson, B.J., Rivera, W., Zhu, X., Hyser, C.D.: AppRAISE: application-level performance management in virtualized server environments. IEEE Trans. Netw. Serv. Manag. (2009). https://doi.org/10.1109/TNSM.2009.04.090404
65. Patikirikorala, T., Colman, A., Han, J.: 4M-Switch: multi-mode–multi-model supervisory control framework for performance differentiation in virtual machine environments. In: 2014 10th

International Conference on Network and Service Management (CNSM), pp. 145–153. IEEE (2014)
66. Tanelli, M., Ardagna, D., Lovera, M.: Identification of LPV state space models for autonomic web service systems. IEEE Trans. Control Syst. Technol. **19**(1), 93–103 (2011)
67. Jamshidi, P., Sharifloo, A., Pahl, C., Arabnejad, H., Metzger, A., Estrada, G.: Fuzzy self-learning controllers for elasticity management in dynamic cloud architectures. In: Proceedings—2016 12th International ACM SIGSOFT Conference on Quality of Software Architectures, QoSA 2016, pp. 70–79 (2016)
68. Padala, P., Shin, K.G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A., Salem, K.: Adaptive control of virtualized resources in utility computing environments. In: ACM SIGOPS Operating Systems Review, vol. 41, pp. 289–302. ACM (2007)
69. Rao, J., Wei, Y., Gong, J., Xu, C.-Z.: QoS guarantees and service differentiation for dynamic cloud applications. IEEE Trans. Netw. Serv. Manag. **10**(1), 43–55 (2013)
70. Anglano, C., Canonico, M., Guazzone, M.: FC2Q: exploiting fuzzy control in server consolidation for cloud applications with SLA constraints. Concurr. Comput. Pract. Exp. **27**(17), 4491–4514 (2015)
71. Anglano, C., Canonico, M., Guazzone, M.: FCMS: a fuzzy controller for CPU and memory consolidation under SLA constraints. Concurr. Comput. Pract. Exp. (2017). https://doi.org/10.1002/cpe.3968
72. Arman, A., Al-Shishtawy, A., Vlassov, V.: Elasticity controller for Cloud-based key-value stores. In: Proceedings of the International Conference on Parallel and Distributed Systems—ICPADS, pp. 268–275 (2012)
73. Park, S.-M., Humphrey, M.: Self-tuning virtual machines for predictable escience. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 356–363. IEEE Computer Society (2009)
74. Al-Shishtawy, A., Vlassov, V.: ElastMan: elasticity manager for elastic Key-Value stores in the cloud. In: Cloud and Autonomic Computing Conference (CAC '13), p. 1 (2013)
75. Zhu, X., Wang, Z., Singhal, S.: Utility-driven workload management using nested control design. In: American Control Conference, 2006. IEEE (2006)
76. Xu, J., Zhao, M., Fortes, J., Carpenter, R., Yousif, M.: Autonomic resource management in virtualized data centers using fuzzy logic-based approaches. Clust. Comput. **11**(3), 213–227 (2008)
77. Wang, L., Xu, J., Zhao, M., Tu, Y., Fortes, J.A.B.: Fuzzy modeling based resource management for virtualized database systems. In: 2011 IEEE 19th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 32–42. IEEE (2011)
78. Ardagna, D., Ghezzi, C., Mirandola, R.: Rethinking the use of models in software architecture. In: Quality of Software Architectures. Models and Architectures, pp. 1–27 (2008)
79. Ljung, L.: Black-box models from input–output measurements. In: Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference, 2001. IMTC 2001, vol. 1, pp. 138–146. IEEE (2001)
80. Ruano, A.E.: Intelligent Control Systems Using Computational Intelligence Techniques, vol. 70. IEE, London (2005)
81. Ashraf, A., Byholm, B., Lehtinen, J., Porres, I.: Feedback control algorithms to deploy and scale multiple web applications per virtual machine. In: 2012 38th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 431–438 (2012)
82. Sumbaly, R., Kreps, J., Gao, L., Feinberg, A., Soman, C., Shah, S.: Serving large-scale batch computed data with project Voldemort. In: Proceedings of the 10th USENIX Conference on File and Storage Technologies, p. 18. USENIX Association (2012)
83. Berekmeri, M., Serrano, D., Bouchenak, S., Marchand, N., Robu, B.: Feedback autonomic provisioning for guaranteeing performance in MapReduce systems. IEEE Trans. Cloud Comput. (2016). https://doi.org/10.1109/TCC.2016.2550047
84. Jiang, J., Lu, J., Zhang, G., Long, G.: Optimal cloud resource auto-scaling for web applications. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 58–65. IEEE (2013)
85. Mao, M., Li, J., Humphrey, M.: Cloud auto-scaling with deadline and budget constraints. In: 2010 11th IEEE/ACM International Conference on Grid Computing (GRID), pp. 41–48. IEEE (2010)
86. 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, pp. 31–40. ACM (2012)
87. Rawlings, J.B., Mayne, D.Q.: Model Predictive Control: Theory and Design. Nob Hill Pub, Madison (2009)
88. Roy, N., Dubey, A., Gokhale, A.: Efficient autoscaling in the cloud using predictive models for workload forecasting. In: 2011 IEEE International Conference on Cloud Computing (CLOUD), pp. 500–507. IEEE (2011)
89. Trushkowsky, B., Bodik, P., Fox, A., Franklin, M.J., Jordan, M.I., Patterson, D.A.: The SCADS Director: scaling a distributed storage system under stringent performance requirements. In: FAST '11 Proceedings of the 9th USENIX Conference on File and Storage Technologies, pp. 163–176 (2011)
90. Patikirikorala, T., Wang, L., Colman, A.: Towards optimal performance and resource management in web systems via model predictive control. In: Australian Control Conference (AUCC) 2011, pp. 469–474 (2011)
91. Wang, X., Chen, M.: Cluster-level feedback power control for performance optimization. In: 2008 IEEE 14th International Symposium on High Performance Computer Architecture, pp. 101–110 (2008)
92. Wang, L., Jing, X., Duran-Limon, H.A., Zhao, M.: QoS-driven cloud resource management through fuzzy model predictive control. In: IEEE International Conference on Autonomic Computing (ICAC) 2015, pp. 81–90 (2015)
93. Tchernykh, A., Schwiegelsohn, U., Alexandrov, V., Talbi, E.: Towards understanding uncertainty in cloud computing resource provisioning. Procedia Comput. Sci. **51**, 1772–1781 (2015)
94. Dutreilh, X., Moreau, A., Malenfant, J., Rivierre, N., Truck, I.: From data center resource allocation to control theory and back. In: 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 410–417. IEEE (2010)
95. Wang, Z., Liu, X., Zhang, A., Stewart, C., Zhu, X., Kelly, T., Singhal, S., et al.: AutoParam: automated control of application-level performance in virtualized server environments. In: Proceedings of the 2nd IEEE International Workshop on Feedback Control Implementation in Computing Systems and Networks (FeBid). Citeseer (2007)
96. Almeida Morais, F.J., Vilar Brasileiro, F., Vigolvino Lopes, R., Araújo Santos, R., Satterfield, W., Rosa, L.: Autoflex: service agnostic auto-scaling framework for IaaS deployment models. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 42–49. IEEE (2013)
97. Lama, P., Zhou, X.: Efficient server provisioning with end-to-end delay guarantee on multi-tier clusters. In: 17th International Workshop on Quality of Service, 2009. IWQoS, pp. 1–9. IEEE (2009)

98. Lama, P., Zhou, X.: Coordinated power and performance guarantee with fuzzy MIMO control in virtualized server clusters. IEEE Trans. Comput. **64**(1), 97–111 (2015)
99. Lama, P., Zhou, X.: Autonomic provisioning with self-adaptive neural fuzzy control for percentile-based delay guarantee. ACM Trans. Auton. Adapt. Syst. **8**(2), 9 (2013)
100. Grimaldi, D., Pescape, A., Salvi, A., Persico, V., et al.: A fuzzy approach based on heterogeneous metrics for scaling out public clouds. IEEE Trans. Parallel Distrib. Syst. (2017). https://doi.org/10.1109/TPDS.2017.2651810
101. Frey, S., Luthje, C., Reich, C., Clarke, N.: Cloud QoS scaling by fuzzy logic. In: IEEE International Conference on Cloud Engineering (IC2E) (2014)
102. Minarolli, D., Freisleben, B.: Virtual machine resource allocation in cloud computing via multi-agent fuzzy control. In: Proceedings—2013 IEEE 3rd International Conference on Cloud and Green Computing, CGC 2013 and 2013 IEEE 3rd International Conference on Social Computing and Its Applications, SCA 2013, pp. 188–194 (2013)
103. Qu, C., Calheiros, R.N., Buyya, R.: Auto-scaling Web Applications in Clouds: A Taxonomy and Survey. arXiv preprint arXiv:1609.09224 (2016)
104. Qin, G., Duan, Z., Wen, G., Yan, Y., Jiang, Z.: An improved anti-windup bumpless transfer structures design for controllers switching. Asian J. Control **16**(4), 1245–1251 (2014)
105. 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 **71**(13), 2727–2741 (2008)
106. Liu, J., Zhang, Y., Zhou, Y., Zhang, D., Liu, H.: Aggressive resource provisioning for ensuring QoS in virtualized environments. IEEE Trans. Cloud Comput. **02**(03), 119–131 (2014)
107. Xu, C.-Z., Rao, J., Bu, X.: URL: a unified reinforcement learning approach for autonomic cloud management. J. Parallel Distrib. Comput. **72**(2), 95–105 (2012)
108. Islam, S., Keung, J., Lee, K., Liu, A.: Empirical prediction models for adaptive resource provisioning in the cloud. Future Gener. Comput. Syst. **28**(1), 155–162 (2012)
109. Gambi, A., Pezze, M., Toffetti, G.: Kriging-based self-adaptive cloud controllers. IEEE Trans. Serv. Comput. **9**(3), 368–381 (2016)
110. Fargo, F., Tunc, C., Al-Nashif, Y., Akoglu, A., Hariri, S.: Autonomic workload and resources management of cloud computing services. In: 2014 International Conference on Cloud and Autonomic Computing (ICCAC), pp. 101–110. IEEE (2014)
111. Gandhi, A., Chen, Y., Gmach, D., Arlitt, M., Marwah, M.: Minimizing data center SLA violations and power consumption via hybrid resource provisioning. In: 2011 International Green Computing Conference and Workshops (IGCC), pp. 1–8. IEEE (2011)
112. Vittorio Papadopoulos, A., Ali-Eldin, A., Årzén, K.-E., Tordsson, J., Elmroth, E.: PEAS: a performance evaluation framework for auto-scaling strategies in cloud applications. ACM Trans. Model. Perform. Eval. Comput. Syst. **1**(4), 15 (2016)
113. Lu, Q., Xu, X., Zhu, L., Bass, L., Li, Z., Sakr, S., Bannerman, P.L., Liu, A.: Incorporating uncertainty into in-cloud application deployment decisions for availability. In: 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), pp. 454–461. IEEE (2013)
114. Farokhi, S., Jamshidi, P., Brandic, I., Elmroth, E.: Self-adaptation challenges for cloud-based applications: a control theoretic perspective. In: 10th International Workshop on Feedback Computing, Seattle, USA (2015)
115. Jamshidi, P., Pahl, C., Mendonça, N.C.: Managing uncertainty in autonomic cloud elasticity controllers. IEEE Cloud Comput. **3**(3), 50–60 (2016)
116. Vittorio Papadopoulos, A.: Design and performance guarantees in cloud computing: challenges and opportunities. In: 10th International Workshop on Feedback Computing (2015)
117. Kang, J.-M., Bannazadeh, H., Leon-Garcia, A.: SAVI testbed: control and management of converged virtual ICT resources. In: 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), pp. 664–667. IEEE (2013)
118. Cappello, F., Caron, E., Dayde, M., Desprez, F., Jegou, Y., Primet, P., Jeannot, E., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Quetier, B., Richard, O.: Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In: Proceedings—IEEE/ACM International Workshop on Grid Computing 2005, pp. 99–106 (2005)
119. Chase, J., Grit, L., Irwin, D., Marupadi, V., Shivam, P., Yumerefendi, A.: Beyond virtual data centers: toward an open resource control architecture. In: Selected Papers from the International Conference on the Virtual Computing Initiative (ACM Digital Library) (2007)
120. Zimbra. Zimbra Collaboration Serve
121. Banks, J., Carson II, J.S., Barry, L., et al.: Discrete-Event System Simulation, 4th edn. Pearson, Forlag (2005)
122. Iqbal, W., Dailey, M.N., Carrera, D., Janecek, P.: Adaptive resource provisioning for read intensive multi-tier applications in the cloud. Future Gener. Comput. Syst. **27**(6), 871–879 (2011)
123. Chieu, T.C., Mohindra, A., Karve, A.A., Segal, A.: Dynamic scaling of web applications in a virtualized cloud computing environment. In: IEEE International Conference on E-Business Engineering, 2009. ICEBE'09, pp. 281–286. IEEE (2009)
124. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Stat. Comput. **14**(3), 199–222 (2004)
125. Kelly, F.P.: Models for a self-managed Internet. Philos. Trans. R. Soc. Lond. A **358**(1773), 2335–2348 (2000)
126. White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., Joglekar, A.: An integrated experimental environment for distributed systems and networks. ACM SIGOPS Oper. Syst. Rev. **36**(SI), 255–270 (2002)
127. Park, S.M., Humphrey, M.: Feedback-controlled resource sharing for predictable escience. In: 2008 SC—International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2008 (2008)
128. Moltó, G., Caballer, M., Romero, E., de Alfonso, C.: Elastic memory management of virtualized infrastructures for applications with dynamic memory requirements. Procedia Comput. Sci. **18**, 159–168 (2013)
129. Box, G.E.P., Jenkins, G.M., Reinsel, G.C.: Forecasting and Control, Time Series Analysis. Prentice Hall, Englewood Cliffs (1994)
130. Ren, Z., Xu, X., Wan, J., Shi, W., Zhou, M.: Workload characterization on a production Hadoop cluster: a case study on Taobao. In: 2012 IEEE International Symposium on Workload Characterization (IISWC), pp. 3–13. IEEE (2012)
131. Google. Google cluster workload traces
132. Arlitt, M.F., Williamson, C.L.: Web server workload characterization: the search for invariants. ACM SIGMETRICS Perform. Eval. Rev. **24**(1), 126–137 (1996)
133. Gong, Z., Gu, X.: PAC: pattern-driven application consolidation for efficient cloud computing. In: 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 24–33. IEEE (2010)

134. Stewart, C., Kelly, T., Zhang, A.: Exploiting nonstationarity for performance prediction. ACM SIGOPS Oper. Syst. Rev. **41**, 31–44 (2007)
135. Kelly, T.: Detecting performance anomalies in global applications. In: Proceedings of the 2nd Conference on Real, Large Distributed Systems, vol. 2, pp. 42–47. USENIX Association (2005)
136. Henriksson, D., Lu, Y., Abdelzaher, T.: Improved prediction for web server delay control. In: Proceedings. 16th Euromicro Conference on Real-Time Systems, 2004. ECRTS 2004, pp. 61–68 (2004)
137. Liu, X., Heo, J., Sha, L., Zhu, X.: Adaptive control of multi-tiered Web applications using queueing predictor. In: IEEE Symposium Record on Network Operations and Management Symposium, pp. 107–114 (2006)
138. IRCache. Web Caching project, 2001
139. Chen, F., Lambert, D., Pinheiro, J.C.: Incremental quantile estimation for massive tracking. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 516–522. ACM (2000)
140. Dunning, T.: t-digest: an algorithm for computing extremely accurate quantiles (2015)

Amjad Ullah received the MSc Degree in Computer Science from Quaid-i-Azam University, Islamabad Pakistan in 2005, MSc in Advanced Distributed System from University of Leicester, UK, in 2011 and a PhD Degree in Computer Science from University of Stirling, Scotland, UK in 2017. His research interests cloud computing and software engineering with a focus on enabling computer systems with intelligent dynamic adaptive behaviour using a blend of techniques including Bio-inspired adaptive control methodologies, computational intelligence, fuzzy control systems, machine learning, scheduling and optimization techniques.

Jingpeng Li is presently a Reader at the Division of Computer Science and Mathematics, University of Stirling, UK. Dr. Li received the MSc Degree in Computational Mathematics from Huazhong University of Science and Technology, China, in 1998, and the PhD in Operational Research from University of Leeds, UK, in 2002. His research areas include intelligent transport scheduling and planning, metaheuristics, multi-objective decision making, optimization and search methodologies, machine learning, data mining, fuzzy logic, and software engineering. He has published over 50 technical papers in which more than 20 papers are in the top international journals (e.g., Evolutionary Computation, IEEE Trans. on Evolutionary Computation, European Journal of Operational Research, Transportation Research Part B, Knowledge-Based systems, etc.).

Yindong Shen holds a Doctoral Degree in Operations Research and Artificial Intelligence from University of Leeds, UK. She is now a Professor at Huazhong University of Science and Technology, Wuhan, China. She is also a Committee Member of International Federation of Operational Research Societies (IFORS) Developing Countries Committee, a Council Person of Operations Research Society of China (ORSC) and Vice-presiding Officer of Operations Research Society of Hubei Province. Her major research interests are on modeling and applications of operations research, intelligent transport scheduling, public transit planning, metaheuristic. She was awarded with IFORS OR in Development Prize as a Runner-up in 2005 and as a Finalist in 2014 for her work on public transit planning and scheduling.

Amir Hussain is Professor of Cognitive Computing Science at the University of Stirling in Scotland. He obtained his BEng in Electronic and Electrical Engineering (with the highest 1st Class Honours, with distinction) and PhD (in Novel Neural Network Architectures and Algorithms for Real-World Applications), both from the University of Strathclyde in Glasgow, UK, in 1992 and 1997, respectively. Following a Research Fellowship at the University of Paisley (now West of Scotland), UK (1996–1998), and a Research Lectureship at the University of Dundee, UK (1998–2000), he joined the University of Stirling in Scotland, in 2000, where he is currently Professor and Founding Director of the Cognitive Big Data Informatics (CogBID) Laboratory (http://cosipra.cs.stir.ac.uk). He is Founding Editor-in-Chief of Springers Cognitive Computation Journal and the New BMC/Springer Journal of Big Data Analytics. He is Founding Series Editor for the Springer Book Series on Socio-affective Computing and Spring Briefs on Cognitive Computation. He also serves on the Editorial Board of a number of other leading journals including, the IEEE Transactions on Neural Networks and Learning Systems and the IEEE Computational Intelligence Magazine. His research interests are cross-disciplinary and industry focused, aimed at pioneering next-generation brain-inspired multimodal Big Data cognitive technology for solving complex real world problems. He has (co)authored over 300 publications (including over a dozen Books, over 100 journal papers, and the worlds first research monographs on the multi-disciplinary areas of: cognitively inspired audio-visual speech filtering for multi-modal hearing-aids, sentic computing for natural language processing, and cognitive agent based computing). He has led more than 50 major multi-disciplinary research projects, as Principal Investigator, funded by national and European research councils, local and international charities and industry. He has supervised more than 30 PhDs to-date, and serves as an International Advisor to various Governmental Higher Education and Research Councils, Universities and Companies. He regularly acts as invited Keynote Speaker, and has organized (as General/Organizing Co-Chair) over 50 leading international conferences to-date

(including IEEE WCCI, IEEE SSCI, IJCNN, BICS and INNS Big Data Conference Series). He is an Invited Member of several IEEE TCs, including the IEEE SMC TC on Cognitive Computing, and the IEEE CIS Emergent Technologies TC. He is Chapter Chair of the IEEE UK and RI Industry Applications Society Chapter, and Founding Co-Chair of the INNS Big Data Section. He is a Fellow of the UK Higher Education Academy (HEA), and Senior Fellow of the Brain Sciences Foundation (USA). More details on his research profile can be found on his homepage: http://cs.stir.ac.uk/ahu/.