DOI 10.12694/scpe.v20i2.1537
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 2, pp. 399–431. http://www.scpe.org

RESEARCH ON AUTO-SCALING OF WEB APPLICATIONS IN CLOUD: SURVEY, TRENDS AND FUTURE DIRECTIONS

PARMINDER SINGH *, POOJA GUPTA †, KIRAN JYOTI ‡, AND ANAND NAYYAR §

Abstract. Cloud computing emerging environment attracts many applications providers to deploy web applications on cloud data centers. The primary area of attraction is elasticity, which allows to auto-scale the resources on-demand. However, web applications usually have dynamic workload and hard to predict. Cloud service providers and researchers are working to reduce the cost while maintaining the Quality of Service (QoS). One of the key challenges for web application in cloud computing is auto-scaling. The auto-scaling in cloud computing is still in infancy and required detail investigation of taxonomy, approach and types of resources mapped to the current research. In this article, we presented the literature survey for auto-scaling techniques of web applications in cloud computing. This survey supports the research community to find the requirements in auto-scaling techniques. We present a taxonomy of reviewed articles with parameters such as auto-scaling techniques, approach, resources, monitoring tool, experiment, workload, and metric, etc. Based on the analysis, we proposed the new areas of research in this direction.

Key words: Cloud computing, resource provisioning, web applications, auto scaling, resource estimation

AMS subject classifications. 68M14, 91C15

1. Introduction.

Cloud computing is emerging technology, which provides processing, bandwidth, and storage as a service. Elasticity is the main characteristic of cloud computing. This technique allocated and de-allocated the resources from the processes as per increment or decrement in requirement. The resource pools are seeming unlimited for the users and can acquire or release the resources anytime [5, 96]. Regular monitoring of giving services is required to ensure the Quality of Service (QoS) and need to fulfill the Service Level Agreements (SLAs). SLA violation leads to the penalty to cloud providers. It is a big challenge for the cloud service providers to provide services within budget and raise profit from datacenters. QoS assures that the behavior of cloud services towards reliability, availability, elasticity, cost, time, etc. [75]. The infrastructure providers offer different pricing policies, companies such as Amazon [131] provides resources for a fixed price per hour. Thus, the application providers should decide the resources for processes, while maintaining the SLAs. In auto-scaling, decision-making techniques to allocate the number of resources to different processes are categorized as reactive and proactive. The reactive technique regularly watches events such as CPU, workload, queue, etc., and performs the elastic operation for resources as per threshold. Proactive forecasting methods used to predict the traffic from the past workload. So far none of the technique is splendid in all the cases [84]. The reasons for ambiguity and diffusion in resource allocation for cloud environment are heterogeneity of resources, dynamic application requirements, and failures. As for now, none of the technique can tackle previously mentioned issues. Autonomic cloud computing can contribute the self-optimization, self-protecting, self-healing, and self-configuration scaling [68].

1.1. Motivation for Research.

- Auto-scaling techniques for cloud application applied to estimate the required resources to process the input requests. Web applications workload is dynamic with sudden burst due to flash workload. This study focused on various auto-scaling techniques for web applications in cloud computing.
- We have recognized the need for detailed survey specifically for the web applications. A methodological survey has been carried out for auto-scaling techniques for web applications. Hence, we summarized the present research challenges and future scope in this area.

1.2. Contribution of the Study.

- A detail investigation has done to study various existing auto-scaling techniques in cloud computing.
- The mentioned techniques classification has been done as per the common characteristics.

*Lovely Professional University, India (parminder.16479@lpu.co.in).
†Lovely Professional University, India (pooja.19580@lpu.co.in).
‡Guru Nanak Dev Engineering College, India (kiranjyotibains@yahoo.com).
§Duy Tan University, Da Nang, Vietnam (anandnayyar@duytan.edu.vn).

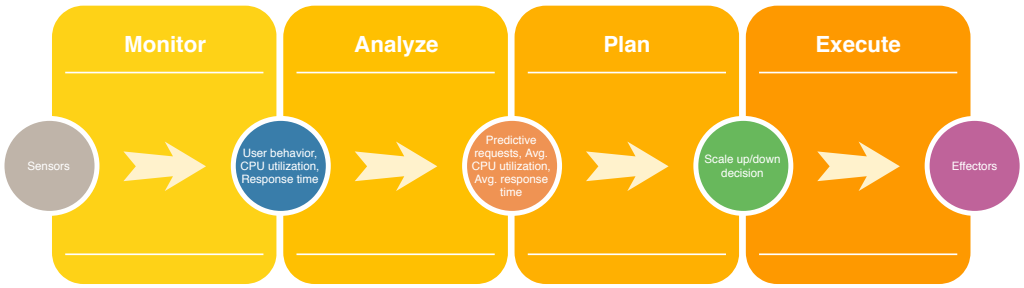

FIG. 2.1. *MAPE Control Loop System*

- Future research direction in the area of auto-scaling is presented.

1.3. Related Surveys. Earlier surveys have been conducted by the authors [49, 88, 85, 54, 28, 107, 23, 109], but the regular updates in cloud infrastructure and persistent research yielding the new research areas. There is a need to explore the present challenges and new research area in this field. This survey augments the existing studies and recent research articles to describe the research challenges for web applications in cloud computing.

2. Auto Scaling. Auto-scaling is a technique to dynamically adjusts the resources allocated to elastic applications as per the incoming workloads. Auto-scaler in the cloud environment is generic while some are application specific to meet the SLA, QoS and minimizing the cost of scaling. The auto-scaling challenge for the web applications is to dynamically grow or shrink the resources to meet fluctuated workload requirement. Autonomous scaling techniques work without human intervention. Autonomic systems are self-(configuring-optimizing-protecting-healing) [67]. The auto-scaling following the MAPE loop (Figure 2.1): Monitoring (M), Analysis (A), Planning (P) and Execution (E) [92].

- **Monitoring**: The monitoring system collects the information from a cloud environment about the compliance of user expectations, resource status, and SLA violation. It provides the state of infrastructure to the cloud provider, and users get to know about application status with expected SLA. Auto-scaling protocols are decided on the basis of performance metrics for web applications. The author suggested parameters such as resize numbers, operating interval, decision duration, decision threshold, refractory period and instance bounds [45]. Generally metrics provided by cloud providers are related to VM management; otherwise, it will be taken from the operating system. The proxy metrics are used to reduce the complexity of metrics such as hypervisor level and application level (e.g., CPU utilization, workload).
- **Analysis**: The collected information is further processed in the analysis phase. It gathers all information from metrics and current system utilization and prediction information of future workload. Some auto-scaler are working on a reactive approach. The decision is taken after analyzing the current system state. The threshold values are fixed to scale in/out decisions, while others are using a reactive approach or both. Reactive is a sophisticated approach because it's always a delay between the settings of resources for scaling decision. The VM startup time varies from 350 to 400 seconds [90]. Flash crowd and events are still a challenge with the reactive approach.
- **Planning**: Analysis phase evaluates the present state, now the planning phase has to decide to scale up/down or scale in/out to compliance with SLA and profit trade-off.
- **Execution** : Execution phase is already decided in the planning phase. Cloud providers API is responsible for the execution of planning. The client is unaware of the issues in the execution phase. VMs are available to users for a certain period, the startup time of VM takes some time, and these delays have been already discussed with the user in resource SLA.

Auto-scaling techniques have the following research challenges:

Under-provisioning: The application has not sufficient resources to serve all incoming requests from the servers. It may happen due to flash crowd or events, or poor auto-scaling algorithm. This situation leads to

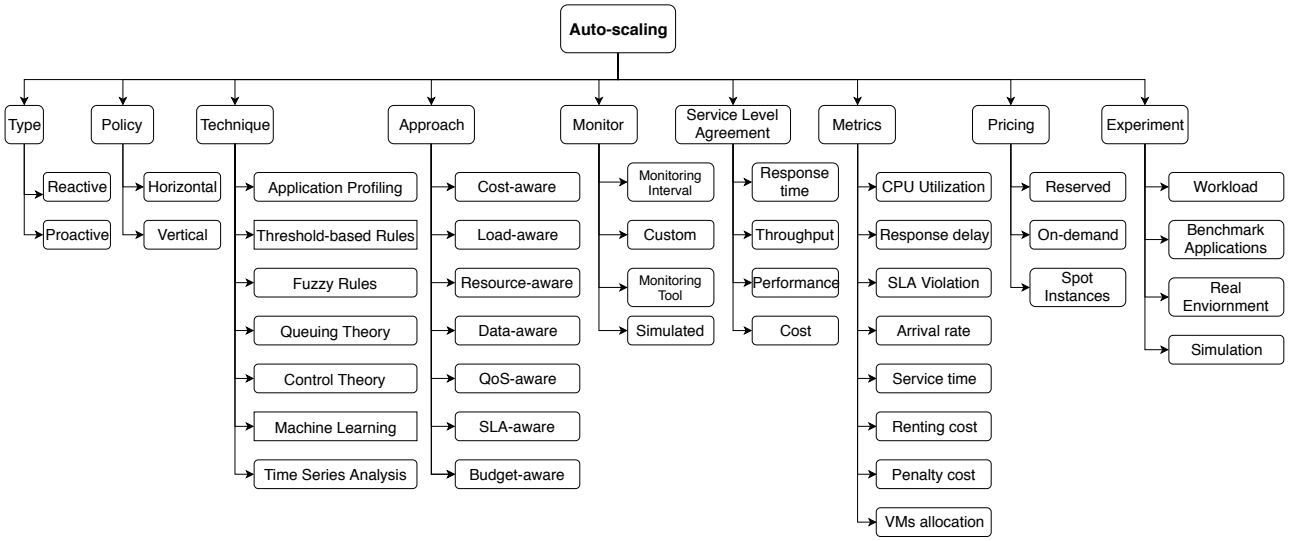

FIG. 3.1. *The taxonomy of web application in cloud computing*

SLAs violation and providers have to pay the penalty to the users, and reliability of the providers also effects. The servers take some time to be back in the normal state.

Over-provisioning: In this situation, the number of resources to process the application is more than the required resources. Service provider's reliability is increased in this case. SLAs violation is minimum in over provisioning and up to a certain level beneficial to handle the fluctuated workload. It affects the profit of the service provider, and on-demand services become costly for the clients. No perfect solution exists either in automatic or autonomous scaling.

Oscillation: It is a pack of both unwanted situations. Rapid resources scaling is taking place without considering the effect on application performance. The condition can be avoided using static and dynamic threshold value fixed for the VMs scaling. A cooling down period is another approach used to handle the oscillation [71].

3. Taxonomy of Auto-scaling. The proposed taxonomy for web application in cloud environment is shown in Figure 3.1. The existing research classified based on the parameters mention in the taxonomy. The taxonomy covered the following points:

- **Type:** Auto-scaler is a crucial component in cloud computing. Auto-scalers are grouped into two categories: Reactive and Proactive. The reactive approaches took the scaling decision by analyzing the current state of the system. Proactive technique analysis the historical data and take scaling decision. Many article formed the new techniques using hybridization of these methods.
- **Policy:** Cloud service providers widely use horizontal scaling as elasticity feature. Cloud providers are providing a fix and customizable resources where the user can configure VMs by specifying memory, cores, bandwidth, etc. Some articles developed the auto-scaling techniques using vertical scaling, where the user can re-configure the VMs resources such as CPU, memory and network bandwidth as per the requirement change. The Centurylink service provider gives the service to scale the CPU cores without vertical downtime.
- **Auto-scaling techniques:** The researcher in the cloud computing used various techniques for analysis and planning phase in MAPE loop to automate the scaling process. In literature, widely used techniques classified in 7 major techniques such as Threshold rules, fuzzy rules, application profiling,machine learning, queuing theory, control theory and time series analysis.
- **Approach:** The newly added feature in the taxonomy is the approach of the investigator on above of the techniques to auto-scale. In the literature, the investigator vision to improve one or more factors such as cost, resource optimization, QoS, SLA violation, etc. Further, the researcher get clear idea

about the articles which improve the specific objectives.

- **Monitoring Tools:** It act as performance indicators. It helps to determine the scaling decisions. Monitoring interval defines the performance of auto-scaler. Balanced performance can be achieved by selecting the right monitoring interval according to the application. Amazon cloudwatch monitoring used by many articles whether some are using custom monitoring tool.
- **Service Level Agreement (SLA):** The service providers need to know the expectations of the customers. It is the master service agreement between the customer and service providers related to various performance outage issues. This document describes the responsibilities of the customer and service providers. It helps the customer to compare the performance among different service providers. The commonly consider performance metrics in SLA related to auto-scaling are response time, budget, cost, throughput, etc.
- **Performance metrics:** It is an essential tool to enhance the reliability of the users moving their applications to the cloud. Difference articles are using various metrics to check the performance of their model such as response time, CPU usage, input request rate, etc.
- **Pricing Model:** Cloud providers are categories the cloud resources in three different pricing models: on-demand, reserved and spot-instances. On-demand resources give performance guarantee to the target application. The number of resources grows or shrink as per the workload change. The reserved resources are a fixed number of resources. Amazon provides the spot instances relatively cheaper than the on-demand resources. Spare capacity instances sell through auction mechanism and user acquire the resources by submitting the bid. Single cloud and multi-cloud resource estimation challenges are different for web applications.
- **Experiment:** The experiment part describes the experiment evaluation in the various articles. The researcher used real time workload or synthetic workload to input the user requests. The benchmarking application are also widely used in the research article for the multi-tier web application. Experiment characteristics considered for the implementation of the model has been reviewed in the taxonomy of auto-scaling. Synthetic and real workload used by an article for the analysis of the model. Cloudsim simulator used by many articles, while some manuscripts used real testbed for the study of the model.

4. Elastic Applications. The elastic applications are dynamic in nature towards change in workload and variables. The load balancer is responsible for the management of elastic applications. Cloud computing applications are managed on VMs, and VMs are managed by the servers. Auto-scaler is a decision maker for the scaling of resources. The system is said to be autonomous because human intervention is not required. Many cloud applications (e.g., Video streaming, web applications) are elastic in nature [85]. Most of the papers about elastic applications considered the web applications as compared to other elastic applications. Web applications consist of three tiers: Presentation, Application, and Database. Most of the articles focused on the application or business tier scaling. The major issue with auto-scaling is to scale the small running jobs, while long-running jobs are coming under the scheduling problem.

4.1. Web Applications Architectures. Earlier single tier architecture has used the dedicated server for each tier such as a web server (load balancer), application server and database server. The load balancer transfers the load among other instances of single layer architecture. This architecture is not suitable according to the elasticity and scalability features of cloud computing.

Most of the companies are now using multi-tier architecture (e.g., Amazon), where each tier in architecture serves a specific purpose as shown in Figure 4.1. The most important benefit of multi-tier architecture is to manage the scalability and elasticity features. Resource management of multi-tier applications is still a challenge due to higher interdependencies between the different layers [31]. Web tier, application tier, and database tier are deployed on the web server, application server, and database server. The responsibilities of the servers are:

- Web server: It accepts or rejects the incoming client request. Another important work of web server is to serve the static content. It also passes the request to the application server. The response is delivered back to the user.
- Application Server: It receives the request from the web server. The literature survey is biased towards the scaling of VMs for the application server. Business logic is processed by this tier. It requests the database for the required data. Optimization of queries can be done at this level also. After processing

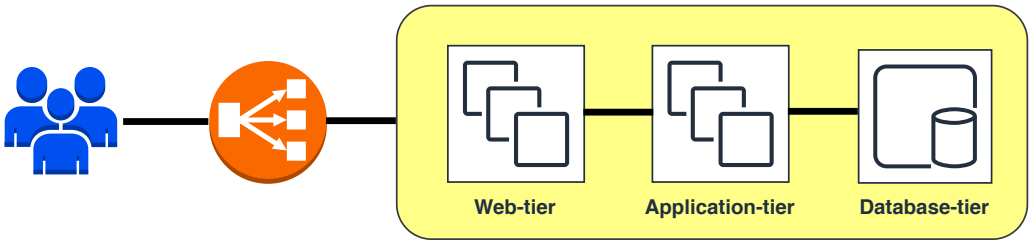

FIG. 4.1. *3-tier Web Architecture [31].*

data again send back to the web servers in the desired format.

- Database Server: Database management system is working at this level. Single tenant and multi-tenant databases compatible with the cloud architecture (e.g. Oracle 12c). Structured and unstructured relational database management system is used to store the data and pass the required data to application servers.

Another type of applications is service-based web applications such as Facebook and e-commerce website of Amazon. Each service represented as the node is connected with another service through directed edges and whole system abstracted as a directed graph. The application further classified in Micro-services and Service-Oriented Architecture (SOA). Our article is mainly focusing on multi-tier web applications.

5. Survey on Auto-scaling Techniques. This literature survey is focusing on auto-scaling techniques specifically for web applications in cloud computing. To the best of our knowledge, there is no survey specifically focuses on web applications auto-scaling for analysis and planning phase. Auto-scaling of cloud computing has a large number of articles published. We put the papers in the associated category by identifying the approach, algorithms used in the research paper for a better understanding. We again revised the models, metrics, monitoring tools, etc., from the articles and create tables according to the classification of techniques. The symbol '-' in tables represent the lack of information in the article.

Author [49] have done a survey of Internet applications deployed on dedicated or shared data centers. It focuses on the web server's admission control issue. This service is related to the SaaS layer of the cloud architecture. The elastic nature of web applications is more relevant with auto-scaling and can be done in two ways: horizontal or vertical. Identification has done on the basis of approach reactive or proactive. The metric used by the technique is identified in the survey. Workload used in the analysis of the approach is also discussed along with the experimental setups. SLA parameter has to check the validity of the model, so SLA parameters are collected from the papers.

Resources are the major factor in auto-scaling. To the best of our knowledge, no existing survey has identified all types of resources used in a cloud environment. Existing surveys considered only on-demand resources in cloud infrastructure. In this paper, the authors have identified other type of resources (e.g. Spot-instances, reserve and on-demand), for application tier elastic layer.

Another important survey done by the authors [88] on resource management in the cloud, but little attention has given to auto-scaling. Author [85] focused on the elastic application including web application. This article classifies the technique and very useful literature survey, but the survey did not focus on the type of resources such as on-demand, reserve and spot instances. Many new emerging fields such as fog computing, edge computing, dew computing takes the support from the cloud computing for elastic services [124].

We have carried out a very diverse survey by including these surveys with specifically to web applications auto-scaling techniques. A survey has done on the basis of auto-scaling technique. Author [85] has used the classification technique and we have added more approaches along with mentioned former groups. This section covers the literature survey on auto-scaling techniques. Each article is reviewed on the basis of auto-scaling taxonomy. The classification of auto-scaling techniques are:

1. Application Profiling (AP)
2. Threshold-based Rules (TR)
3. Fuzzy Rules (FR)

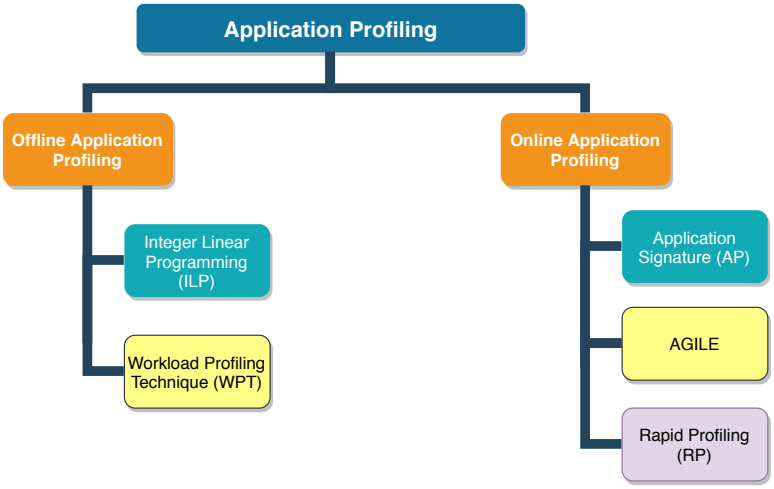

FIG. 5.1. *Methods of Application Profiling*

4. Control Theory (CTH)
5. Queuing Theory (QTH)
6. Machine Learning (ML)
7. Time Series Analysis (TSA)

5.1. Application Profiling. Application profiling is a process of finding the peak point of resource utilization while running an application workload. The workload used for profiling can be real or synthetic. The test can be both online and offline for application profiling. It is one of the simplest ways to find the desired resources at a different point of time.

Profiling of job through an offline process gives resource requirements at different levels of workload. The produced resources requirements help auto-scaler to monitor the resource provisioning task in a precise manner. Author [119] applied the integer linear programming (ILP). Authors [39, 43, 108] used workload profiling technique. The profiling cons are to reproduce requirements manually after every update in applications.

To overcome the offline profiling issues, online profiling comes in rescue. The fine grain tasks need the VMs immediately to cater to the flash workload. Author [132] devised the technique to profile the application workload with the further classification of application signatures. The classification has been carried out on the basis of a number of machines required at different time stamps of application workload. The trained decision tree update every time when the new application profiled according to its characteristics. Author [98] applied the online technique to estimate the resources of each tier while estimating specific tier other tiers gained ample resources. This process continues for each tier and gets the model for every tier. Author [30] designed a rapid profiling approach for multi-tier web applications. Analysis of resources requirement correlation with each tier of the web application inhomogeneous environment and profile the VM for a specific tier. This approach can estimate the performance of VM without running on each tier. So it helps to put the newly added VM rapidly into the service. Author [128] presented the architecture for the self management of the application in cloud infrastructure. In this article, the author performed the application profiling using micro services. [83] designed the approach using application profiling.

The method to find the critical point of utilization of resources while running of application workload (real or synthetic). It is further classified into two categories (Figure 5.1):

- Off-line Application Profiling: This method of profiling is performed in a detailed manner in resource provisioning of tasks at different stages of workload. The advantage of offline profiling is to perform profiling manually after the applications are updated. The different methods of offline application profiling are:
  - Integer Linear Programming (ILP): It is a mathematical optimization problem. In this partial or

all variables are integer. The constraints and objective function is linear [119].
  - Workload Profiling Technique (WPT): This technique gather the workload information. The workload modeling performed to estimate the performance under the stressed conditions. The workload profiling done with various types of testing techniques such as limit finding, soak testing, etc. [39, 43, 108].
- Online Application Profiling: This method of profiling is dynamic in nature and fulfill the needs of the fine grained tasks that immediately require working virtual machines with different workloads.The different techniques of online application profiling are:
  - Application Signatures (AS): This technique identifies a small set of classes of workload and classifies them according to workloads. The resource allocation of workload is cached at runtime [132, 120].
  - Elastic distributed resource scaling (AGILE): In this technique, wavelets are used for fulfilling the resource demand prediction with much large time for the applications servers to start up before falling short of performance [98].
  - Rapid Profiling (RP): In this technique, the individual performance of the virtual machine instance is profiled after it is obtained from the cloud. This technique helps to judge the best tier for the profiled instance of virtual machine [30].

The cloud is providing resources on-demand. Spot instances can be used with fault tolerance for cloud applications. Maximum techniques are considering the on-demand resources in the auto-scaling decision. Author [108] used the heterogeneous spot instances with on-demand resources which could minimize the cost to a great extent. Spot instances are up to 90% cheaper than on-demand and reserve resources. Spot instances have an out-of-bid issue, but can be useful for flash crowd and event challenges. It will provide the cost benefit to cloud providers.

Table 5.1 shows the taxonomy of reviewed articles in this section.

5.2. Threshold-based Rules. Threshold based techniques are simple to implement and very popular. For deciding the threshold value requires a deep understanding of all the parameters of the corresponding environment and workload. Amazon EC2 [4] and RightScale [113] are using threshold-based technique. The threshold based rule technique is purely related to planning. The number of resources allocated to the application in the form of VM according to a set of rules. There are two types of scaling decision: Scaling up/down vertically increase or decrease the capacity of the working node. Scaling in/out refers to horizontally increasing or decrease the VM capacity. The scaling up/out and scaling down/in is working on the principle in Algorithm 1.

Algorithm 1 The algorithm of threshold rule based auto-scaling

```
1:  if x_1 > tU_1 and/or x_2 > tU_2 and/or ... then        ▷ tU_i is upper threshold
2:      if Clock % dU == 0 then                            ▷ dU is upper duration
3:          n = n + r                                      ▷ Scale-out or Scale-up
4:      else if Clock % iU == 0 then                       ▷ iU is upper cool down period
5:          do − nothing
6:      end if
7:  end if
8:  if x_1 < tL_1 and/or x_2 < tL_2 and/or ... then        ▷ tL_i is upper threshold
9:      if Clock % dL == 0 then                            ▷ dL is lower duration
10:         n = n − r                                      ▷ Scale-in or Scale-down
11:     else if Clock % iL == 0  then                      ▷ iL is lower cool down period
12:         do − nothing
13:     end if
14: end if
```

There are two parts of Algorithm 1: condition and action. The *Clock* is a system clock, $x1$ and $x2$ are performance metrics (e.g. Number of requests, response time, budget, CPU load, etc.), tU and tL are the upper and lower threshold values of the performance metrics. Threshold conditions checked with upper duration dU

TABLE 5.1
Taxonomy on application profiling based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[119]	Reactive	Horizontal	AP and ILP	Cost-aware	-	Arrival rate	Cost and response time	Reserved	Custom testbed
[30]	Proactive	Vertical	AP and ANN	Load/ Capacity aware	Custom tool	Response time	CPU and memory usage	On-demand	Custom testbed. SpecWeb , TPC-W, TPC-C applications on Xen
[43]	Reactive and Proactive	Horizontal	AP	Resource-aware	Custom tool	No. of servers and arrival rate	Response time	on-demand	Custom testbed (Intel Xeon 38 servers)
[132]	Proactive	Vertical	AP and CMAC	Workload-aware	Custom tool (Script)	Response time	CPU, I/O, memory, swap	On-demand	Custom testbed. Applications on Xen (SpecWeb , TPC-W, TPC-C)
[98]	Proactive	Horizontal	AP	Cost and resource aware	-	Cost and response time	No. of VMs	On-demand	Custom decision agent (VirtRL) and olio application
[39]	Proactive	Horizontal	AP and TSA	User-behavior	Custom Tool.	Throughput	CPU usage and arrival time	On-demand	Custom testbed. MediaWiki Application
[108]	Reactive	Horizontal	AP	Fault tolerant	Simulated	Availability and response time	CPU load	On-demand	Custom Simulation
[128]	Mixed	Horizontal	AP	Resource aware	Real time	Response time	Request rate and avg. response time	on-demand	Amazon AWS
[82]	Reactive	Horizontal	AP	Data-aware	Simulated	Response latency	Throughput, No. of resources	On-demand	Custom testbed. IBM X3500 M4 machine with Xen
[120]	Reactive	Horizontal	AP	Resource aware	Simulated	Response time and success ratio	CPU usage, response time	On-demand	Custom Simulation
[83]	Proactive	Horizontal	AP	Workload aware	Real testbed	Throughput and response time	Input rate, response time	Reserved and on demand	6 Cassandra 3.0 nodes with Linux Ubuntu 14.04 Server x86 64. Synthetic workload.

and lower duration dL, which considered in the seconds. If certain conditions are met, the action will be taken. The manager should decide resources r acquired or released during the action performed from the total resources n. During horizontal scaling r is the number of VMs and during vertical scaling r is CPU or RAM. The lower and upper cool down period is set as iL and iU, during this time auto-scaler do nothing.

It is easy to deploy a threshold-based technique in a cloud environment. Defining rules are a difficult task, it requires input from the client on different QoS metrics. QoS parameters have performance trade-off. Application manager has to give threshold value of performance metrics parameters. The experiment carried

out by the authors [70], they define the performance benefits of the application-oriented metrics than system specific metrics. The threshold needs to be carefully decided otherwise it can cause the oscillation problem [33]. To handle the oscillation issue, certain techniques such as cool down, calm and inertia periods are set, so no scaling decision can take place in these time slots.

Most of the techniques are using one or maximum two performance metrics. Commonly used performance metrics are input request rate, CPU load time and average response time of application. Some of the authors are taking application response time as a performance metric [33, 51] while few focused on the system level metrics such as storage, network, and CPU [52]. Author [87] considered the scaling overhead and SLA violation. Due to the introduction of a new type of resources [108] evaluate the availability metrics along with response time.

The number of threshold values also varies with different techniques. Most of the providers are using two threshold values: Upper and the lower threshold value. Author [52] used four threshold values. In this technique along with upper threshold ($ThrU$), the author defines $ThrbU$, which is marginally lower than $ThrU$ and $ThroL$ is pretty above than the lower threshold ($ThrL$). The significance of setting such a threshold is to determine the trends. Scaling action could be taken as per the trends.

Metrics are fetched from the monitoring tool. Collecting the run-time data from the monitor and taking action as per the rules is a reactive approach.

RightScale [113] voting system is combined with the reactive approach. If most of VMs agree to scale up/out or down/in decisions, then scaling action will be performed. Threshold needs to set by the application manager. Several authors are using the RightScale technique [123, 71, 45, 25]. Author [25] working on the scalability of web applications by defining the set of rules for the active sessions, further extend his work [24] using RightScale. The rules are decided as per the upper and lower threshold value of the sessions. If the sessions are going upper or lower scaling decision will be taken.

RightScale is working on the voting system, but it is highly dependent on the threshold values set by the application-manager and the nature of the application workload. Author [71] compare the RightScale with another technique and concluded the disadvantages of RightScale. Researcher [123] attempted to overcome this issue using the strategy tree. The regression-based model has been used for three types of scaling policies. Strategy tree follows the parent as per the workload trend.

The threshold rule-based technique is popular due to its simplicity and the client can easily understand. Reactive nature of TR technique is a major challenge. Another issue in the TR technique is to set appropriate performance metrics. In a cloud environment, the approximate startup time of the VM is 5 to 10 minutes, so to ready, the machine in a reactive manner put a delay in service, which leads to performance degradation. To resolve this issue to some extent, an author [86] suggested the dynamic threshold values. The initial values of the threshold are static and later dynamic threshold values set as per the SLAs violation.

Multi-cloud scaling is another issue in the auto-scaling. Auto-scaling for three-tier application has been developed [48]. The rule-based technique has been devised to take scaling decision for multiple data center.

Along with the on-demand resources, spot-instances are used for web application provisioning. The spot instances are 90% cheaper than on-demand resources. So effective scaling strategy can provide significant benefit to the clients as well as a service provider to reduce the SLA violation. Author [108] provided a reliable solution to use spot instances for web application provisioning. Spot-instances have a potential for fault tolerance and tackling of web application's flash crowds or flash events workload.

Smart kill [20] is an important idea used to reduce the cost. Most of the service provider charge hourly basis. It can further improve the system performance to avoid VM starting and shutdown time. It can reduce the SLAs violation.

It is easy to implement the rule-based auto-scaling of the particular application. If the workload and nature of application can forecast easily than rule-based technique can be used. Application with an unpredictable pattern should choose other suitable scaling strategies.

Table 5.2 shows the taxonomy of reviewed articles in this section.

5.3. Fuzzy Rules. One of the rule-based technique for an auto-scaling approach is fuzzy rules. In this technique, rules are defined using if-else conditions. The major advantage of fuzzy based auto-scaler over the threshold based rule-based approach is linguistic terms e.g. low, medium, high. It uses the control system as

TABLE 5.2
Taxonomy on threshold rules based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[71]	Hybrid	Horizontal	RightScale and TSA (LR + AR(1))	User behavior	Simulated	-	CPU load, Request rate	on-demand	Custom simulator with real experiment
[123]	Reactive	Horizontal	RightScale, Strategy tree	Policy Based	Custom tool(4 minutes)	-	CPU idle time, number of sessions	on-demand	Real provider. Amazon EC2 + RightScale (PaaS) + a simple web application
[70]	Reactive	Horizontal	TR	QoS based	Custom tool	CPU load	Average waiting time in queue	on-demand	Public cloud: FutureGrid, Eucalyptus India cluster
[24]	Reactive	Horizontal	RightScale + MA to performance metric	Performance Based	Custom tool	-	Number of active sessions	on-demand	Custom testbed. Xen + custom collaborative web application
[45]	Reactive	Horizontal	RightScale	Qos/ Performance based	Amazon CloudWatch	-	CPU load	on-demand	Real provider. Amazon EC2 + RightScale (PaaS) + a simple web application
[91]	Reactive	Vertical	TR	SLA based	Simulated	-	CPU load, memory, bandwidth, storage	on-demand	Custom simulator, plus Java rule engine Drools
[52]	Reactive	Both	TR	Storage/ Network aware	-	-	CPU load, response time, network link, load, jitter and delay.	on-demand	-
[51]	Reactive	Both	TR	Cost Aware	Custom tool. 1 minutes	Response time	CPU, memory, I/O	on-demand	Custom testbed (called IC Cloud) + TPC
[20]	Proactive	Horizontal	TR + QTH	QoS Aware	Amazon CloudWatch. 1-5 minutes	Response time	Request rate	on-demand	Real provider. Amazon EC2 + Httperf + MediaWiki
[86]	Reactive	Vertical	TR	Cost/SLA Aware	Simulated. 1 minute	Response time	CPU load	on-demand	Custom simulator
[48]	Reactive	Horizontal	TR	Cost/ Performance aware	Custom tool and manual.	Response time, Sessions	CPU load	on-demand	Multi-cloud Amazon EC2 (4 data centers)
[87]	Reactive	Horizontal	TR + Learning Automata	SLA aware	Simulated. 5 minutes	-	Scaling Overhead, SLA violation	on-demand	CloudSim
[79]	Reactive	Horizontal	TR	SLA aware	Custom	Turnaround time	Turnaround time, No. of requests	On-demand	Real. Google compute engine.
[35]	Reactive	Horizontal	TR	Performance aware	Amazon CloudWatch	Response time	CPU usage	On-demand	Real. Amazon EC2

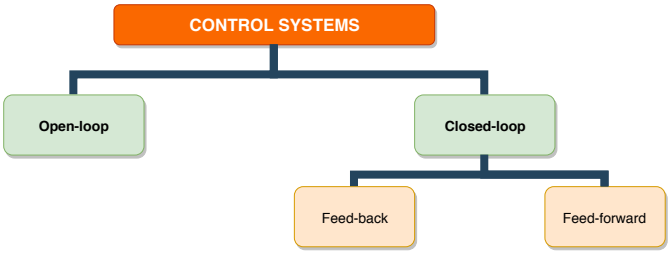

FIG. 5.2. *Categories of control system*

a performance model for estimating the required resources (output variable) for the workload (input variable). Fuzzy sets are formed from the input and output variables, the membership function defines this process between the $(0, 1)$ interval. Fuzzification is the process of transforming input variables into fuzzy sets. The inverse transformation of single numeric values to best inferred fuzzy value is known as defuzzification.

The fuzzy rule-based model used to construct the auto-scaler. This auto-scaler is not able to handle the dynamic workload because most the components of the fuzzy controller are fixed at the design time (e.g. Rule set, membership function). Regular update in the fuzzy controller with on-line monitoring can make the system of adaptive nature, can better handle the dynamic workload [136, 140]. Author [140] proposed a model to apply the fuzzy-controller at the application tier, and forecast the resources required input workload. Author [136] use the same method for the database tier. Predict the required resources r_{t+1} for the time step $t+1$, considering no flash workload during that time step. Author [47] developed a dynamic approach for horizontal scale-out for the dynamic workload. Author proposed a fitness function: $\mathcal{F}_k = \sum_{i}^{N} a_i \frac{m_{i,k}}{\mathcal{R}_i}$. Here, metric observation represented by $m_{i,k}$ and $\mathcal{R}_i$ at k sample time $a_i \epsilon [0, 1]$. The model works well under the failure of VMs, and robust for different workloads.

Author [13] designed a fuzzy controller for scaling of scientific applications. The vertical scaling policy developed based on threads in the application and load on the VM. In future, proactive policy can be combined with this technique to reduce the waiting time in VM setup.

Many research articles extend the work with the neural fuzzy controller. Four-layer neural network uses to represent the fuzzy model [74]. The first layer belongs to the input node. The second layer represents each input variable membership to the fuzzy set. Layer three determines the precondition part of fuzzy rules. Final layer act as a defuzzifier, which used to convert the layer 3 in numeric output. The rules and membership of nodes are formed on-line with the help of parameters and structure learning.

Table 5.3 shows the taxonomy of reviewed articles in this section.

5.4. Control Theory. This method is used in two phases analysis and planning of the MAPE loop. It automates the processing systems such as data centers, storage systems, and web server systems. The main goal is to automate the scaling process. The controller maintains the controlled variable y and manipulated variable y_{ref} matches to the desired level. Manipulated variable act as an input to the target system, the output is measured by the sensor.

The different categories of control systems are (Figure 5.2):

- Open-loop: It is also called the non-feedback method. In this method, feedback is not considered for validating the desired goal. The output is calculated using the present state of the system.
- Closed-loop: The closed loop systems are further categorized into two categories:
  - Feed-back: The monitoring of the output and also a deviation from the goal is monitored by the feedback controller.
  - feed-forward: The anticipation of any kind of error in output is performed by the feed forward method. The action is performed in it after considering the type of error in the system.

Feedback controller is mostly used in the reviewed articles (Figure 5.3). It further divided into several categories [104] (Figure 5.4):

- **Fixed gain controllers**: The simplest of all the available controllers. The different types of fixed gain

TABLE 5.3
Taxonomy on fuzzy rules based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[140]	Reactive	Horizontal	FR	Workload aware/ SLA aware	Custom tool	Throughput	CPU allocation, Arrival rate	Reserved containers	Custom testbed + applications (Java Pet Store)
[73]	Proactive	Horizontal	FR	QoS based	-	End-to-end delay	Number of servers per tier	on-demand	Simulation
[136]	Proactive	Vertical	FR	Qos Based	Xentop. 10 seconds	Response time, throughput	CPU load, No. of queries, disk I/O bandwidth	on-demand	Custom setup. Xen + (RUBiS and TPC-H)
[74]	Proactive	Vertical	FR + ANN	Workload aware	Custom tool. 3 minutes	End-to-end delay	Number of requests, resource usage	on-demand	Simulation
[40]	Reactive	Horizontal	FR	SLA based	Custom tool	Response time	Arrival rate	on-demand	Custom simulator
[59]	Proactive	Horizontal	FR	Load aware	Custom tool	Number of hits	RMSE	On-demand	Microsoft Azure Cloud
[47]	Proactive	Horizontal	FR + PID controller	QoS based	Amazon cloudwatch	CPU usage, NETIN, NETOUT	CPU load and network usage	on-demand	Real setup: Amazon EC2
[80]	Reactive	Horizontal	FR	Resource aware	Amazon Cloudwatch	Response time. 200 ms	Arrival rate, VM allocation	On-demand	Amazon EC2
[13]	Proactive	Vertical	Fuzzy rules + CPU controller	Parallel applications aware	Custom tool	Response time	Thread in application and load on VM	on demand	Simulation with CloudSim.

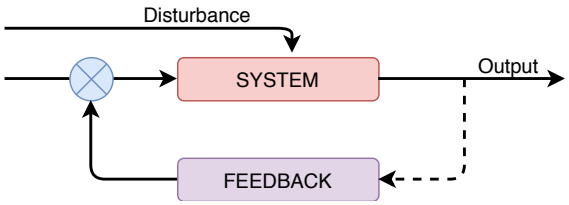

FIG. 5.3. *Working of Feedback control system*

controllers are PID(Proportional-Integral-Derivative), PI(Proportional-Integral) and I(Integral). In it, Proportional Integral Derivate (PID) is most common controller, with following control algorithm:

$$u_k = K_p e_k + K_i \sum_{j=1}^{k} e_j + K_d \left(e_k - e_{k-1}\right) \tag{5.1}$$

Manipulated variable is represented by u_k (e.g. New number of VM); e_k difference between set point y_{ref} and y_k output; k_i, k_p and k_d are the integral, proportional and derivative gain parameters, which further adjust as per the target system.

The authors of [77, 76] use the I controller to manage the required VMs on average CPU utilization. Park and the author [103] deploy PI controller to adjust the resources for batch jobs and used their execution process. k_p and k_i (gain factors) adjust manually according to trail-and-error [77] basis or target application model. Authors [142] adjusted a PID controller derivative term using the RIL agent. The agent learns to reduce the sum of squared error of control variables without affecting budget and

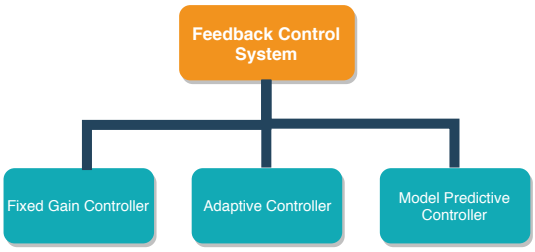

FIG. 5.4. *Categories of Feedback control system*

time constraints over time.

- **Adaptive controllers**: These controllers are the adaptive controllers that work as per the online data made available by target systems. It is incapable of handling the flash load. It is further classified into three categories
  - Self-Tuning PID controller(SPID)
  - Self-tuning regulator(STR)
  - Gain scheduling(GS)

  The adaptive controller is also used in the literature. For example, the author [2] used the two models, adaptive and proactive controllers for scaling-down, based on the input workload and dynamic gain parameters. Scaling-up is done using a reactive approach. Author also devised proportional controller using proactive technique [1]. Author [102] introduced an adaptive PID controller for MIMO and used second-order ARMA for non-linear relationships between performance and resource allocation. The controller can fit the disk I/O usage and CPU. Author [14] used smoothing splines with gain. A gain-scheduling adaptive controller applied to estimate the number of servers for input workload. Author [62] combine the Kalman filter with controllers (SISO and MIMO) for CPU allocation to VMs. Author [41] predicted job completion using kriging model. The master node enqueued all the incoming request.
- **Model predictive controllers (MPC)**: It is proactive in nature that considers the present output and also this model forecasts the future behavior of the system. One of its categories is Look-ahead controller [114]. An optimization problem is solved by considering the cost function. These types of controllers come under the proactive approach. Author [114] devised workload forecasting model using the ARMA model and look-ahead controller. Fuzzy Model Predictive Controller developed using the fuzzy model [136].

  Author [38] devised the auto-scaling technique for vertical memory. The application performance and resource utilization considered in developing the hybrid controller. The objective is to consume less memory for the task, and achieve the highest memory utilization. The author achieved 83% memory utilization. This work can further extend to increase memory utilization and considered the cache memory in vertical scaling.

  The auto-scaling task is highly dependent on the design of the controller and the target application. The controller main objective is to add and remove the resources in the auto-scaling process is very effective. The problem still persists, when the workload is dynamic and non-linear. General auto-scaling model is required, that can be an adaptive controller with MPCs.

As discussed earlier, the controller has to tune the input variable (number of VMs) to calculate the output variable (CPU load). In order to achieve this goal, a model has been devised to represent the formal relationship, which determines how input parameters affect the output variables. This relationship is known as a transfer function in control theory. PID controller represents with a linear equation, there is also the possibility to define with a non-linear equation. Simple PID controller Single-Input-Single-Output(SISO) is used, but there is also a provision to use Multiple-Input-Multiple-Output(MIMO). Following performance models have been used in literature:

- ARMA(X) [102]: ARMA (Auto-regressive Moving Average) model is used to define the characteristics of time series and draw future predictions. ARMAX (ARMA with eXogenous input) devised the

FIG. 5.5. *A simple queuing model with one server [26].*

relationship between two-time series.

- Kalman filter [62]: It is used to make a prediction of time series. It is a recursive algorithm.
- Smoothing splines [14]: It is a polynomial function used to smooth the curves to avoid the noisy observations.
- Kriging model or Gaussian Process Regression [41]: Statistical framework combined with linear regression to predict future values. Unsampled data have been used to perform the forecasting with a confidence measure.
- Fuzzy model [74, 136, 140]: Fuzzy models are used with fuzzy rules. Set of membership elements assigned a degree of value between 0 and 1 (Boolean logic).

Author [101] devised a RIL based controller. This system does not assure the performance in all type of conditions. The work carried specifically for business logic tier. The proposed solution able to handle the controller failures. Author [129] designed a hybrid technique for auto-scaling of cloud applications. SDN controller applied to control the activities of the environment. The proposed model able to reduce the SLA violation upto 0.03% only.

Tables 5.4 and 5.5 show the taxonomy of reviewed articles in this section.

5.5. Queuing Theory. It is one of the widely used for modeling Internet applications. It is used in the analysis phase of the auto-scaling process. It estimates the performance metrics and waiting time for the requests. Queuing theory is a field of applied probability to solve the queuing problem. Queuing issue is quite common in many fields such as telephone exchange, petrol station, supermarket, etc. The structure of the model is shown in Figure 5.5. The requests arrive at the server at mean arrival rate λ and remain in queue till the processing. One or more servers are available to process the requests at the mean process rate μ.

Author Kendall [65] represents the standard notation for queuing model known as kendall's notation. It describe the queue as $\mathcal{A}/\mathcal{B}/\mathcal{C}/\mathcal{K}/\mathcal{N}/\mathcal{D}$.

- $\mathcal{A}$: Arrival process.
- $\mathcal{B}$: Service time distribution.
- $\mathcal{C}$: Number of servers.
- $\mathcal{K}$: Capacity of system. The number of places in the system.
- $\mathcal{N}$: Calling population. Size of users from where the request comes. Open population has an infinite number of customers and closed model is a finite number of customers.
- $\mathcal{D}$: Queue's discipline. It represents the priority of jobs.

$\mathcal{K}, \mathcal{N}, \mathcal{D}$ elements are optional, by default variables are considered as $\mathcal{K} = \infty$, $\mathcal{N} = \infty$ and $\mathcal{D} = FIFO$. FIFO (First In First Out) is mostly used, which served the request as they come. Another important one is PS (Process sharing). M (Markovian) refers to a Poisson process characterized by λ, which indicates the number of arrival request per time unit. D stands for deterministic also refers constantly. G is also used commonly known as general distribution.

Multi-tier applications are complex in nature and the queuing network can be useful. The load balancer is represented by a single queue and distribute the coming requests to the VMs.

Queuing theory is useful for stationary nature systems. It can work with both proactive and reactive kind of environment. The main objective of the cloud-based system is to develop a model on the basis of some known parameters (e.g. Arrival rate λ). Performance metrics measured as the mean response time, and the average waiting time in the queue. The web application workload is dynamic, it requires the timely recalculation of queuing model and metrics.

The queuing model can be used in two ways: simulation and analytical method. The analytical approach can be used with simple models. $M/M/1$ and $G/G/1$ are the well-known methods used to define the arrival

TABLE 5.4
Taxonomy on control theory based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[103]	Reactive	Vertical	PI controller	Cost-aware	Sensor library	Job's deadline	Job progress	On-demand	Custom setup. BLAST , OpenLB, ADCIRC, WRF, and Montage applications on HyperV
[77]	Reactive	Horizontal	PI controller- (Proportional thresholding) and ES		(Xen) Hyperic- HQ	-	CPU load, request rate	On-demand	Custom setup. Simple web service + Xen + ORCA
[102]	Proactive	Vertical	MIMO adaptive controller	SLO based	Xen + custom tool. 20 seconds	Response time	CPU usage, disk I/O, response time	On- demand	Custom testbed. Xen + 3 applications (RUBiS, TPC-W, media server) + ARMA (performance model)
[14]	Proactive	Horizontal	CTH: Linear Regression + Gain-scheduler (adaptive) + Smoothing splines + (performance model)	Resource-aware	20 seconds	Response time	No. of servers and requests, response time	on-demand	Real provider. Amazon EC2 + CloudStone benchmark
[62]	Proactive	Vertical	CTH: Adaptive SISO and MIMO controllers + Kalman filter	Resource-aware	Custom tool. 5-10 seconds	Response time	CPU load	on-demand	Custom testbed. Xen + RUBiS application
[76]	Reactive	Horizontal	CTH: PI controller (Proportional thresholding)	SLO based/ Workload based	Hyperic SIGAR. 10 second	-	CPU load, request rate	on-demand	Custom Setup. Xen + Modified cloudStone (Hadoop)
[2]	Hybrid	Horizontal	CTH: QTH + Adaptive controllers	SLA Based	Simulated	No. of request not handled	Service rate, No. of requests	on-demand	Custom simulator in Python
[1]	Hybrid	Horizontal	CTH: Adaptive, Proportional controller + QTH	Load Aware	1 minute simulated.	-	Service rate, Total no. of requests and pending in buffer	on-demand	Custom simulator (using Python)
[41]	Proactive	Horizontal	CTH: Self-adaptive controller + Kriging model (performance model)	QoS aware	-	Execution time	Number of incoming and enqueued requests, number of VMs	on-demand	Custom setup. Private cloud + Sun Grid Engine (SGE)
[53]	Proactive	Horizontal	CTH: fuzzy controller + TS	Workload Aware	Custom tool (Fuzzy controller)	Response time	Number of hits, RMSE	on-demand	Custom testbed. Azure Small VM
[36]	Proactive	Horizontal	CTH: Learning automata	SLA Aware	Custom tool. 5 minutes	Response time	Scaling overhead	on-demand	Cloudsim

TABLE 5.5
Taxonomy on control theory based reviewed literature (continuation)

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[38]	Proactive	Vertical	CTH: Hybrid Controller + TS	Application and Resource aware	Control interval using /proc/ meminfo	Response time	CPU and memory utilization	on-demand	Custom testbed. Xen hypervisor (RUBBoS application)
[105]	Proactive	Horizontal	CTH: PID controller + Fuzzy logic	Resource aware	Amazon Cloudwatch	CPU utilization, sensitivity analysis	Scheduling gain, VM failure	on-demand	Custom tested and Amazon EC2. Wikipedia, FIFA world cup workload
[138]	Hybrid	-	CTH: Software defined network (SDN) controller	Security aware	-	DDoS attack prevention	Arrival rate	Containers	Simulated
[101]	Proactive	Horizontal	CTH: Reinforcement learning	Application aware	6 minutes	Response time	Cost and SLA violation	on-demand	RUBiS applicatoin. Custom testbed.
[129]	Hybrid	Both	Software defined network (SDN) controller	Resource aware	Simulated. 1 minute	Response time	CPU capacity, bandwidth	On-demand VMs or Containers	Simulate with CloudSimSDN

and service process. Simulation can be used for the complex system to obtain the desired metrics.

$M/M/1$ (Poisson-based) is a basic queuing model. Exponential distribution is followed by both the arrival times and the service times. The mean response time R of $M/M/1$ model can be calculated as $R = \dfrac{1}{\mu - \lambda}$. Arrival rate is represented by λ and μ shows the service time respectively. $G/G/1$ is another simple method. Inter arrival time and service time are controlled by general distributions with prior information of mean and variance. $G/G/1$ system is represented by the following equation:

$$\lambda \geq \left[s + \frac{\sigma_a^2 + \sigma_b^2}{2(R-s)} \right]^{-1} \tag{5.2}$$

R is the mean response time and s is average service time. The variance of inter-arrival time is σ_a^2 , σ_b^2.

Little's Law [64] is also used in many queuing scenarios. It states that the average number of requests $E[C]$ in the system is same as the average customer arrival rate λ multiplied by the average time of each customer in the system $E[T] : E[C] = \lambda \times E[T]$.

Simple and complex queuing used in the research articles for analysis of the performance of applications and system.

Author [1, 2] used a $G/G/n$ queue to model a cloud application, and n represents the number of servers. The model used to calculate the required resources to process the hosted application workload λ, the response time according to the configuration of servers.

An elastic cloud application can be represented using the queuing network, considered each VM as a separate queue. Author [130] devised a technique using $G/G/1$ queues. Histogram used to predict the peak workload. The number of servers calculated as per the queuing model and peak workload in specific time step. The reactive approach further used to correct the value. The deficiency of the technique is the under-utilization of resources.

Multi-tier applications can be deployed in a cloud environment and assigned one or more queues for a specific tier. Author [141] proposed a closed system with a network of queues, which handles a limited number of users. Author [50] deploy the multi-tier application as an open system. $G/G/n$ queues have been used and considered one queue per each tier. Author [8] used the queuing model for a three-tier web application. It computes the response time per each tier. Author [10] proposed a hybrid auto-scaling technique. The queuing

theory clubbed with the time series analysis technique. The prediction error and prediction interval can be reduce with appropriate selection of the model using classification technique.

The models discussed are considered the analysis part of the MAPE loop. Some techniques are used to implement the planning phase using optimization algorithm and predictive controller [2] in order to maximize the revenue for the datacenters [134]. On-line monitoring captures the number of requests, which further input to the queuing model to estimate the number of VMs required for the processing of application workload [50, 130]. Author [141] proposed a model using regression technique to approximate estimate the number of CPU by calculating the quantity of a client requests at each tier (e.g. Browsing, ordering or shopping). Author [44] proved the importance of parallelization of workload in multi-core system. The model can be improved with the generic design which can implemented in Hadoop, edge and other environments for vertical scaling.

Application of the queuing model is for cloud application and system modeling. It defines the static architecture and required an update in parameters or structure of the model. Simulation and the analytical tool can be used to solve the model. Any change in the number of input request or a number of resources needs to update the model, which is not cost-efficient. Most of the articles used CloudSim simulator [18], and some authors [135] developed a new simulation tool specifically for the auto-scaling. It includes the trace file of widely used workloads.

Queuing model is a component in auto-scaler. The model works quite efficiently with linear parametrized applications, and not fit for the applications with a non-linear workload. The queuing model requires more efforts to work with multi-tier application because of complex nature and non-linear relationships.

Table 5.6 shows the taxonomy of reviewed articles in this section.

5.6. Machine Learning. Machine learning is a technique that is used on online learning for the constructing dynamic approach for the estimation of resources. It is a self-adaptive technique as per the workload pattern available [3]. The machine learning algorithms reclassified into various categories. The review is made for the different categories used in this particular literature.

- **Reinforcement Learning (RIL)**: It is one of the widely used machine learning approaches. The agent performs an action as per the received environmental state. As in a cloud environment, auto-scaler acts as an agent. An agent works on the principle of trial and error method. It gets a response or reward for each action performed [126]. The optimal scaling decision is taken by sensing the current state, performance metric, and type of application. The auto-scalar or agent works to yield more rewards. The main objective of machine learning is to control the data. The purpose of the agent is to fetch appropriate action of each states.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{0}^{\infty} \gamma^k r_{t+k+1} \tag{5.3}$$

At time $t+1$ the rewards gained are R_{t+1}, the γ factor is the discount factor. The value function $Q(s,a)$ known as Q-value function defines the policy. The $Q(s,a)$ values evaluates the cumulative rewards for each state s by execution action a.

$$\mathcal{Q}(s,a) = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1|S_t} = s, a_t = a \right\} \tag{5.4}$$

This proactive learning method makes the decision with the state of application about the future reward(e.g. response time). The result is generated after the complete execution of the application on the cloud. The phases analyze and plan of MAPE process are covered by RIL techniques.Firstly, data about the application and rewards are taken from the lookup table (or any other structure) for later use (analyze phase).In the planning phase, the data is used to take the scaling action.

To apply RIL to auto-scaling, some basic elements need to define: first, the action set A, the state space S, and the reward function R. The first two depend upon the type of scaling: horizontal and vertical. Reward function depends upon the cost to acquire the resources (VMs, network bandwidth, etc.), and SLA violation penalty cost. While considering the horizontal scaling, the state defines as input workload and the number of VMs.

TABLE 5.6
Taxonomy on queuing theory based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[134]	Reactive	Horizontal	QTH	Budget Aware	Simulated	Response time	Arrival rate, service time	on-demand	Custom simulator (Monte-Carlo)
[141]	Proactive	-	QTH + Regression (Predict CPU load)	QoS Based	Custom tool. 1 minute	-	Number and type of transactions (requests), CPU load	on-demand	Custom simulator, based on C++Sim. + Data collected from TPC-W
[130]	Hybrid	Horizontal	QTH + Histogram + Thresholds	Resource Aware	Custom tool. 15 minutes	Response time	Peak workload	on-demand	Custom testbed. Xen + 2 applications (RUBiS and RUBBOS)
[8]	Proactive	Horizontal	QTH + Historical performance model	Prediction /Resource Aware	-	Response time	Arrival rate	on-demand	Custom testbed. Eucalyptus + IBM Performance Benchmark Sample Trade
[50]	Reactive	Horizontal	QTH (model) + Reactive scaling	Cost-Aware	Custom tool. 1 minute	Response time and cost	Arrival rate, service time	on-demand	Custom testbed (called IC Cloud) + TPC-W bench mark
[42]	Proactive	Horizontal	QTH + kalman Fil ter	User Aware	Custom tool. 10 seconds	Response time	Arrival rate	on-demand	Custom TestBed
[133]	Proactive	Horizontal	QTH	QoS based	-	Response time, Bottlenecks	Arrival rate	on-demand	Custom testbed. Open stack
[11]	Proactive	Both	QTH	Resource aware	Custom	Response time	Request arrival rate, No. of VMs per tier	on-demand	Custom testbed. RUBiS benchmarking application
[56]	Reactive	Horizontal	QTH	Resource aware	Custom	Response time	Arrival rate, No. of resource allocation	on-demand	Custom testbed. OpenStack.
[116]	Proactive	Horizontal	QTH + Markov chain	SLO aware	-	Response time	CPU utilization, throughput, Bandwidth, Session loss	on-demand	Simulation and real testbed on Amazon EC2. Synthetic workload using Apache JMeter
[135]	Proactive	Horizontal	QTH (Simulator for auto-scaling)	User behavior and resources	Custom tool. 15 minutes	VM utilization, Arrival rate	Arrival rate	on-demand	Custom Simulator for general purpose
[44]	Proactive	Both	QTH + Amdahl's Law+ Kalman Filters	Cost-aware	Custom testbed. 10 seconds	Cost and Response time	Resource allocation, service rate	on-demand	Real testbed using OpenStack. RUBiS benchmarking applications, WITS traces and synthetic workload.
[10]	Hybrid	Horizontal	QTH + threshold rules	SLO aware	Custom agent	Resource Oscillation	No. of VMs	-	BUNGEE experiment controller. FIFA World Cup, BibSonomy, IBM CICS transactions, German-Wikipedia, Retailrocket traces.

Researcher [127] devised using (w, u_{t-1}, u_t), where w is the the total number of user requests observed per time period; u_t and u_{t-1} are the number of VMs allocated to the application in the current time step, and the previous time step, respectively;
Authors [32] followed the definition of the state as (w, u, p), where w is the total number of requests and u is considered as the number of VMs allocated and p as performance in the contract of average response time. Horizontal scaling has been done on the basis of three decisions: append the new VM, deallocate the VM and do nothing.
On the other hand, the resources assigned to each VM (CPU, RAM) for vertical scaling are considered in state definition. Authors Authors [112, 139] devised the state definition as (mem_1 , $time_1$, $vcpu_1$,...., mem_u, $time_u$, $vcpu_u$), where mem_i, $time_i$ and $vcpu_i$ are the i^{th} VM's memory size, scheduler credit and number of virtual CPUs. The possible operations for three variables can be increased, decrease or no-operation. The RIL defines the task as a combination of each variable operation. Rao et al. again proposed a technique, where RIL agent learned per VM [111]. State definition is configured as CPU, bandwidth, memory, and swap.
RIL learning is also very useful for tasks that are very closely related to auto-scaling problems, for example, the configuration of application parameters [16, 139]. Author [139] include the RIL agent with ANN to configure the parameters as per the application and VM performance, such as Session timeout, MinSpareServers, Keepalive timeout, Maxclient.
RIL can also be combined with control theory. Authors [142] combine the RIL agent with a PID controller. Application parameters are guided by the controller to meet the Service Level Objectives (SLO). The resources are dynamically provisioned according to the parameters. The author [95] estimated the future prediction problems with anomaly detection technique. The isolation unsupervised tree is applied for the upgrading the model. The local and global level auto-scaler design is proposed. The author consider the containerized resources in cloud infrastructure. In real environment, there is more complex anomalies are presented. So, more number of anomalies need to detect to design the robust solution.
RIL algorithms are important to gain the best management policy for cloud applications without any initial knowledge. It is an online approach to learn and adapt as per the workload, application or system change. RIL technique could be a better approach to handle the auto-scaling for general applications, but the RIL approach is not mature enough to deal with the real cloud environment. This is an open research field and efforts are required to handle the flash workload and rapidly change of state and actions.

- **Hidden Markov Model (HMM)** : This is modeled with a hidden Markov chain with the hidden states. In the hidden Markov chain technique, the state is known to the observers whereas in HMM it is invisible to observers. For example, two friends Sam and Rick are communicating on the phone. Sam describes his activities (eating, drinking) and Rick estimated the weather condition from his activities. The estimation is performed on the basis of the maximum likelihood estimation technique. The parameters of HMM are x defines the states in the model, y is the total observations under consideration and state transition probability defined with a and b the output probability shown in Figure 5.6.
  An HMM model usually require fewer observations as compare to basic Markov chain models [99, 100]. The system condition needs to observed minutely and input to the model. The author used the Weka data mining tool for implementing the HMM model, 2 states used with 0.01 iteration cutoff and spherical covariance. As per an author, model given 97% accuracy in the auto-scaling decision. Database tier is not considered in this work.
- **Support Vector Machine (SVM)**: It is also used in some articles. It is one of the supervised learning technique that is a combination of two techniques: classification and regression. It can be applied to non-linear workload [97]. The classification is done on the basis of clusters formed and generates the classes for data under specific region. Linear SVM can be used for classifying the web workload.
  Researcher [78] proposed an adaptive technique with SVM and Linear Regression (LR). Characterization has done on the basis of priority of the job and workload pattern. The generic model has been developed

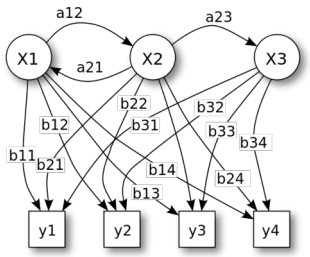

FIG. 5.6. *Hidden Markov Model (HMM) [110].*

for cloud applications. The model can be extended further considering the different QoS parameters according to the SLA and application.

- **Q-learning:** It is the most used algorithm in auto-scaling by extending it with various techniques. Author [127] used the SARSA [126] approach. Some articles [112, 139, 111, 16] did not specify which machine learning approach used in their research, but problem definition and update function resemble the SARSA. There are many problems in Q-learning and SARSA addressed various ways as following [9, 33, 32, 139, 127]:
  - Initial performance is bad and required a large training period. On-line performance is poor before the best solution is found. It requires the exploration of different states and actions.
  - The curse of dimensionality problem in Q-learning. The number of state variables grows the state exponentially. States are stored in lookup tables. As the table grows, it takes time to search for the possible state from the lookup table.
  - The environmental condition has a great impact on the performance of the RIL algorithms. As the conditions change the best optimal solution degrades the performance. RIL need to design to work with application behavior or environmental conditions.

Tables 5.7 and 5.8 show the taxonomy of reviewed articles in this section.

5.7. Time Series Analysis. Time series analysis is a very popular model and has applications in many areas including economics, bioinformatics, finance, engineering, etc., to predict the measurement for future time steps. An example, the number of requests that reach the server for an application at one-minute intervals. The time series is used in the analysis phase to forecast future workload.

Performance metrics (input workload, CPU load) sampled at fixed time intervals (e.g. 5 minutes). The predicted series X gained from the sequence of observation w.

$$X = x_t, x_{t-1}, x_{t-2}, ..., x_{t-w+1} \tag{5.5}$$

Time series applied in auto-scaling to forecast the future workload or resources required. Predefined rules are designed in planning phase [66], and optimize the resource allocation [114].

As discussed earlier, the main objective of time series analysis is to predict the future workload, on q observation from historical workload known as a history window or input workload (where $q \leq w$). Time series analysis classified in two categories: direct prediction and identification of a pattern in time series. The first category, direct prediction contains auto-regression, moving average, ARMA, ARIMA, exponential smoothing, and machine learning approaches:

- Moving average (MA): A widely used to smooth a time series to filter noise from random fluctuation and to make predictions. General formula of MA is as follows:

$$\hat{x}_{t+r} = a_1 x_t + a_2 x_{t-1} + ... + a_q x_{t-q+1} \tag{5.6}$$

$\hat{x}_{t+r}$ is a forecast value from last q observations with weighted average. Prediction interval denotes by r, and typically sets to 1. Equal or different weight factors are assigned to the values denoted

TABLE 5.7
Taxonomy on machine learning based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[127]	Proactive	Horizontal	RIL+ ANN-SARSA + Queuing model	Resource aware	-	Response time	Arrival rate, previous number of VMs	on-demand	Custom testbed (shared data center). Trade3 application (a realistic simulation of an electronic trading platform)
[112]	Proactive	Vertical	RIL+ANN	VM performance	Custom tool	Throughput, response time	CPU and memory usage	on-demand	Custom testbed. Xen + 3 applications (TPC-C, TPC-W, SpecWeb)
[33]	Proactive	Horizontal	TR + RIL	Resource aware	Custom tool. 20 seconds)	Response time	Request rate, response time, number of VMs	on-demand	-
[142]	Proactive	Vertical	CT- PID controller + RIL + ARMAX model + SVM regression	QoS based	-	Application-related benefit function	Application adaptive parameters CPU and memory	on-demand	-
[111]	Proactive	Vertical	RIL + CMAC	SLA based/ Capacity based	Custom tool (Script)	Response time	CPU, I/O, memory, swap	on-demand	Custom testbed. Xen + 3 applications (TPC-C, TPC-W, SpecWeb)
[32]	Proactive	Horizontal	RIL	Resource-aware	-	Response time, cost	Number of user requests, number of VMs, response time	on-demand	Custom testbed. Olio application + Custom decision agent (VirtRL)
[139]	Proactive	Vertical	RIL + ANN	Budget aware/QoS based	Custom tool	Response time	CPU and memory usage	on-demand	Custom testbed. Xen + 3 applications (TPC-C, TPC-W, SpecWeb)
[9]	Proactive	Horizontal	RIL	Workload aware	Simulated	Response time, cost	Number of user requests, number of VMs, response time	on-demand	Custom simulator (Matlab)
[16]	Proactive	Vertical	RIL + Simplex	Resource aware	Custom tool	Response time, throughput	CPU, memory, application parameters	on-demand	-
[99]	Reactive	Horizontal	HMM	SLA aware	-	Cost-Performance	Number of request, MAPE, RMSE	on-demand	-
[78]	Reactive	Horizontal	LR(Linear regression) + SVM	Workload based	Custom tool. (1, 10 and 60 minutes)	Cost-Performance	Prediction time (ms)	on-demand	Simulation
[60]	Reactive	Horizontal	RIL: Fuzzy Q-learning	Workload aware	Custom.	Response time	No. of VMs	On-demand	Simulated and Custom testbed.

TABLE 5.8
Taxonomy on machine learning based reviewed literature (continuation)

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[12]	Proactive	Horizontal	RIL	Resource-aware	Custom	Response time	CPU usage, throughput, Mean Q-value	On-demand	Custom testbed and CloudRL matlab simulation. Benchmarking applications (RUBiS, RUBBoS and Olio)
[137]	Reactive	Horizontal	RIL: Q-learning	Workload aware	Custom. 1-hour.	-	Arrival rate and rewards	reserved, on-demand and spot instances	Custom testbed. Matlab.
[95]	Proactive	Vertical	Unsupervised learning: Isolation based tree	Resource aware	Custom	Response time	CPU utilization, response time, no. of failed sessions	Reserved and on-demand	CloudSim. Custom testbed. RUBiS benchmarking application.
[63]	Proactive	Horizontal	HMM: Multi layered	Data-streaming applications	Custom.	-	CPU usage, Prediction time, RMSE, MAPE	on-demand	Custom testbed. Hadoop for data streaming.

by $a_1, a_2, a_3, ..., a_q$. There are types of MA's are Simple Moving Average (SMA) and the Weighted Moving Average (WMA). Moving average performs in various forms, but the objective of MA is the same. In simple moving average's general formula is considering the arithmetic mean of previous q values. It considers the same weight of all the observations. WMA considers the different weight for each observation. The highest weight assigned to new observation, while less weight is given to previous observations.

- Exponential Smoothing (ES): In contrast to the moving average, the weighted average of previous observation is also calculated, but the ES considers all the observation of time series (w values) from past history. The new parameter α, the smoothing factor has very less influence on the predicted value, because exponentially decreasing weight assigned to the new observations. There are many versions of ES such as simple ES and Brown's double ES [15]. The smoothed value s_t is calculated from the present observation and past smoothed value based on recursive formula: $s_t = \alpha x_t + (1-\alpha)s_{t-1}$. Time series with fewer trends can be forecast using simple ES. Brown's double ES is suitable for linear trend time series. Two smoothing series are required to calculate as:

$$\begin{aligned} s_t^1 &= \alpha x_t + (1-\alpha)s_{t-1}^1 \\ s_t^2 &= \alpha s_t^1 + (1-\alpha)s_{t-1}^2 \end{aligned} \tag{5.7}$$

s_t^2 is calculated from simple ES to s_t^1. The trend d_t and level c_t is obtained from s_t^1 and s_t^2 smoothed values. The forecast value $\hat{x}_{t+r}$ is obtained from following formula:

$$\begin{aligned} \hat{x}_{t+r} &= c_t + r_{dt} \\ c_t &= 2s_t^1 - s_t^1 \\ d_t &= \frac{\alpha}{1-\alpha}s_t^1 - s_t^2 \end{aligned} \tag{5.8}$$

- Auto-regression AR(p): The future value is forecast using the linear weighted sum of previous observation p in the time series:

$$x_{t+1} = b_1 x_t + b_2 x_{t-1} + ... + b_p x_{t-p+1} + \epsilon_t \tag{5.9}$$

p represent the number of observations in the AR equation, which could be different from history window w length. White noise ϵ_t is added in the formula. AR coefficients are calculated using different methods such as maximum likelihood or least squares. The widely used technique in the literature is Yule-Walker equations and auto-correlation coefficient. The formula of auto-correlations as follows for x_t to x_{t-k}, where $k = 1, 2, 3...$:

$$r_k = \frac{covariance(x_t, x_{t-k})}{var(x_t)} = \frac{E[(x_t - \mu)(x_{t-k} - \mu)]}{var(x_t)} \tag{5.10}$$

- Auto-Regressive Moving Average, ARMA(p,q): This model is the hybridization of both AR of order p and MA of order q. ARMA model is described as:

$$x_t = b_1 x_{t-1} + ... + b_p x_{t-p} + \epsilon_t + a_1 \epsilon_{t-1} + ... + a_q \epsilon_{t-q} \tag{5.11}$$

  ARMA model can be used either purely as AR or MA model using ARMA$(p, 0)$ and ARMA$(0, q)$ respectively. ARMA is best suitable for stationary time series.
  The extension of the ARMA model is Auto-Regressive Integrated Moving Average (ARIMA) model is suitable for non-stationary time series. If ϵ (white noise) shows no pattern, then ARIMA(p, d, q) used where d represents the degree of difference.
- Machine learning techniques: Analysis of time series was carried out by many authors using machine learning-based techniques.
  Regression-based techniques are based on statistical methods to form a polynomial function, that find the nearest points from the history window w. Linear regression is ordered 1 polynomial expression. The distance of points should be as less as possible. Multiple Linear Regression is used when there is more than one variable in the expression.
  Neural Network is a group of interconnected artificial neurons put on multiple layers. Multiple inputs are put in the hidden layer, which further given an output. In time series one output against one input from the history window. Random weights are assigned to the input vector during the training phase. The weights are adapted to the optimized output has not achieved.
- Pattern Recognition: Time series data is a combination of seasonal and non-seasonal data. The time series has patterned with a specific time period (e.g. hours, day, month, year, or season) on the basis of short term and long term workload. It finds the matching pattern in the history that relates to the current pattern. It is very similar to the string matching technique [27].
- Signal Processing Techniques: This technique is based on harmonic analysis to decompose a time series signal into different frequencies. Fast Fourier Transformation and spectral density estimation filter the noise and estimating the signal value.
- Auto-correlation: In the linear regression errors are independent, the auto-correlation function is used to deal with the dependent error pattern. The input workload from the history window is shifted recursively, and further compared with the original time series.

The histogram is used to represent the time series. It splits the time series values into equal size bins, and each bin represents the frequency. In literature, it is used to represent the forecast values, resource distribution, and usage pattern.

5.7.1. Review of Articles. Time series analysis for multi-tier applications is most frequently used in the literature. A simple moving average model yields poor results, so that the MA used to remove time-series noise [76, 103]. The resource prediction model of Author [55] was developed using double exponential smoothing and a simple medium and Weighted Moving Average (WMA) applied. Because of w history records, ES significantly provides better results. Author [94] used the Dual ES of Brown to predict the load of work and obtain good results with a small error in the HTTP workload. Author [72] applied neural network and differential evolution for workload prediction. The back propagation method give adequate accuracy. Furthermore, the accuracy of the model can be improved with the pattern classification technique.

The auto-regression models applied for [21, 22, 46, 71, 114] workload and resource forecast. The author taking three previous observations used the AR model to estimate workload [114]. Estimated additional response

time from the forecast values. A resource allocation optimization controller applied, taking SLA violation costs, reconfiguration, and leasing resources into account. In order to forecast requests per second, Author [71] used AR(order 1) and concluded that its performance is very much based on many manager-determined parameters (e.g. history window, adaptation window, size, and monitoring interval). The forecast is based on short and long term trends, depending heavily on the size of the history window. Further, the model extension determines the adjustment window.

The ARMA Model foresees the future workload (number of requests) is simple and efficient. The usage of VMs on the CPU is predicted by Author [37]. In different article [117, 17, 93], the ARIMA model is applied. The historic workload was required in ARIMA. The model's performance depends greatly on its history. ARIMA is ideal for dynamic workload applications such as web applications. The horizontal and vertical scaling was applied to increase the cost-benefit [117]. The classification given is: increasing, stable, seasonally and on/off, used by the authors [89] VMs were repacked (or reconfigured) to provide certain capacities, and workload-based application repacking was performed. The approach then finds the ideal package of VMs. The approach proposed can save 7% to 60% for the use of the resource. With additional QoS parameters, the container based approach can be further optimized. For workload forecasts and evaluate impacts on various QoS-parameters, investigator [17] used ARIMA model. The workload of the web application is dynamic and includes seasonal information. For non-seasonal data, the model provides 91% accuracy but is not suited for a highly non-seasonal workload. This work can be further extended by means of an adaptive approach for working load classification and the heuristic design for ARIMA fit for various classes. As mentioned above, an author [93] presented the GA-based approach to time-series prediction do not fit in for all types of workload. The prediction model has been assessed using traces of real workload. A new metric, describing solution optimization, was introduced in the article called the Elasticity Index (EI). The EI range varies between 0 to 1, with a value of almost 1 the solution is good. Compared to other models, the model gives less error. This approach takes longer to predict the incoming request, even hourly prediction. In addition, the mapping of only few prediction techniques with a particular application pattern and workload pattern can be enlarged. In particular, GA models can design cost, energy, resource sharing, and parameters.

Author [125] performed the workload prediction using classification technique. The historical workload took under consideration of sliding window. The past 5 lags has considered for to fit the model and predict the future request. On the behalf of peak-to-mean-ratio and $l2 - norm$ used to calculate the scale of time series. The model gave sufficient accuracy in terms of RMSE and MAPE for web application workload. There is no existing model which can capture all types of pattern. The error rate is still more than 10%. The model accuracy can be improved with application profiling. Author [69] designed the CloudInsight approach using regression, time series and machine learning techniques. The predictor pool has built and select the prediction model according to the testing phase output.

The precise of neural networks [58, 106] are highly dependent on the size of the input in the history window in several regression equations [14, 106, 71]. Author [58] have used more than one historical value and achieved a better result. The need for a balanced size of the input history window is defined by the author [71]. Regression of different window sizes used to locate forecast values. Another important factor is the r prediction interval. The size of the interval window is examined and found 12 minutes at the right point due to the startup time of VM being between 5 and 15 minutes. The neural network applied to 2 minutes to forecast the gaming load by the author [106]. The neural network in terms of accuracy, however, is better than MA and ES.

The literary authors also focused on horizontal and vertical scaling. It can be taken individually or in a hybrid manner. The researcher [34] investigated that horizontal scaling costs are higher than vertical scaling, but also relatively high throughput. The author prefers to scale horizontally. Model of regression to estimate the future workload. The author [37] concentrated on the flash crowd's horizontal scaling (CPU and memory), while regular changes are made with vertical scaling.

Proactive time series forecasting and a reactive approach can be combined. The author uses a reactive strategy for scale-up and scale-down scale model and developed a model of hybrids that can be self-sized [57]. Polynomial regression is used to determine the number of instances of application and database VMs.

Some authors have also applied a pattern identification method on time series analysis [19, 46, 122]. FFT-based techniques for identifying patterns matching the use of resources (RAM, CPU, I/O, and network) are

proposed [46]. The comparison is done using auto-correlation, histogram, and auto-regression. The authors [19] have developed the algorithm with more parameters and poor performance.

The use of application resources in some articles is also estimated by the use of the mean distribution [21] and, the highest frequency with simple histogram [46]. The current usage and historical data [29], the dynamic load balancer is introduced by the Holt's technique. This can be used with web workloads and able to give efficiency in prediction. This can solve problems with load balancing.

Techniques for time series analysis can predict future web applications workload. Moreover, resource requirements can be predicted through this information. This approach is very attractive because the auto-scaler is known to be able to prepare the VMs in advance. The inconvenience of techniques is precision depending on workload, selection of historical windows, measurements, the interval of prevision and the target application. The time series forecasting exists no best solution for all types of pattern. The future scope of the adaptive or GA-based time series forecast for a specific application, taking QoS parameters into account. Tables 5.9 and 5.10 show the taxonomy of reviewed articles in this section.

6. Future Directions. As per the literature survey, still, there is scope for further improvement in the present auto-scaling solutions for the multi-tier web applications. The following sections describe important future directions in this field.

6.1. Monitoring Tools. Multi-tier web applications are installed on the servers. Cost effective monitoring tools are required to implement, which explore the hidden parameters such as cache memory, service time and type of request (e.g. Compute intensive and data intensive). Application and workload-specific monitoring interval can further improve the auto-scaler results.

6.2. Pricing Model. Companies such as Amazon, Google and Microsoft have their different pricing model. Most of the researchers consider auto-scaling techniques on on-demand resources while considering the unlimited resources. Other types of model are also introduced, for example, Spot instances by the Amazon. The cost of such resources is very less as compare to on-demand resources. Very few authors start working on a spot instance, so the area is very immature and have different research challenges. Auto-scaling technique for spot-instances using reactive and proactive techniques give benefit to cloud providers and clients.

6.3. Resource Allocation. While allocating the resource for input workload, the information of data center resources is also required. Some parameters such as CPU, RAM, Disk are well-known factors considered, but some parameters such as cache memory, network bandwidth, and fault tolerance are least considered in the literature. VMs for input workload for different tiers of the web application is the future scope of many articles. SLA base resource allocation with different QoS parameters need improvements. It can be further improved by extending the queuing network model.

6.4. Horizontal and vertical Scaling. Horizontal scaling is used in most of the articles. Operating system and cloud architecture support horizontal scaling. The vertical scaling is easier in cloud infrastructure and gives better cost benefit. Hypervisors and operating system support could be enhanced for the vertical scaling. Energy and cost-effective VMs allocation, and consolidated migration are future areas for research.

6.5. Workload Predictor. The existing workload predictors are considering the historical workload. Flash workload is hard to predict from the past workload. As growth in the online data mining and deep learning techniques, real-time data can be used to avoid the sudden burst condition in the data center. Web applications face this problem due to any hackers attack (DDOS) or flash event. Categorization of application will be more helpful to handle the flash crowd. Spot instances can be more effective in terms of cost optimization to serve flash workload.

6.6. Multi-cloud Auto-scaling. Multiple cloud providers are supporting multi-tier web architecture. Cloud providers individually proving the reliability of their services. Multi-cloud architecture for web applications needs to work upon. The cloud provider can be selected by considering the application feature as the type of application, input workload, geographical area, etc. The reliability-aware auto-scaling in the multi-cloud environment by factorizing the various parameters.

Table 5.9
Taxonomy on time-series analysis based reviewed literature

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[21]	Proactive	Horizontal	TSA: AR(1) and Histogram + QTH	Resource Aware	Simulated. 1, 5, 10 and 20 minutes	Response time	Request rate and service demand	On- demand	Custom simulator + algorithms in Matlab
[22]	Proactive	Horizontal	TSA:	AR Energy Aware	Simulated	Service not available (login), energy consumption	Login rate, number of active connections, CPU load	On- demand	Simulator
[106]	Proactive	Horizontal	TSA: ML Neural Network (compared to MA, last value and simple ES)	QoS based	2 minutes	Prediction accuracy	Number of entities (players)	On- demand	Simulator of a MMORPG game
[46]	Proactive	Vertical	TSA: FFT and Discrete Markov Chains. Compared with auto-regression, auto-correlation, histogram, max and min.	SLO based	Libxenstat library. 1 minute	Response time	CPU load	On- demand	Custom testbed. Xen + RUBiS + part of Google Cluster Data trace for CPU usage
[94]	Proactive	-	TSA: Brown's double ES	Performance based	10 minutes	-	Number of requests per VM	On- demand	Custom testbed. TPC-W
[66]	Proactive	Both	TSA: AR + TR	Resource Aware	Zabbix	-	Number of requests	On- demand	Hybrid: Amazon EC2 + Custom testbed (Xen + Eucalyptus + PhpCollab application)
[19]	Proactive	Horizontal	TSA: Pattern matching	Workload Aware	100 seconds	Number of serviced requests, cost	Total number of CPUs	On- demand	Analytical models
[122]	Hybrid	Vertical	TSA: FFT and Discrete-time Markov Chain	SLO Aware	Libxenstat library. 1 second	Response time, job progress	CPU load, memory usage	On- demand	-
[114]	Proactive	Horizontal	TSA: AR	QOS Aware	-	Response time, VM cost, application reconfiguration cost	Number of users in the system	On- demand	No experimentation on systems
[57]	Hybrid	Horizontal	TR (scale out) + TSA, polynomial regression (scale in)	SLA based	1 minute	Response time	CPU load (scale out), number of requests, number of VMs (scale in)	On- demand	-
[37]	Proactive	Both	TSA: ARMA	SLA based	-	Prediction accuracy	Number of requests, CPU load	On- demand	Custom testbed. Xen and KVM
[55]	Proactive	-	TSA: Brown's double ES. Compared with WMA	Cost-Aware	Simulated	Prediction accuracy	CPU load, memory usage	On- demand	Custom testbed. TPC-W

TABLE 5.10
Taxonomy on time-series analysis based reviewed literature (continuation)

Ref	Type	Policy	Technique	Approach	Monitor	SLA	Metric	Pricing	Experiment
[58]	Proactive	Horizontal	TSA: ML Neural Network and (Multiple) LR + Sliding window	Resource Aware	Amazon CloudWatch. 1 minute	Prediction accuracy	CPU load (aggregated value for all VMs)	On- demand	Real provider. Amazon EC2 and TPC-W application to generate the dataset and R-Project.
[34]	Proactive	Both	TSA: Polynomial regression	Resource/ Cost Aware	Custom tool. 1 minute	Response time, cost for VM, application license and reconfiguration	Number of requests	On- demand	Custom testbed. KVM + Olio
[117]	Hybrid	Both	TSA: ARIMA + Repacking	Cost-Aware	4 and 20 minutes	Response time	Number of request, cost	On- demand	Custom simulator
[39]	Hybrid	Horizontal	TSA + Profiler	QoS Aware	5 minutes	Response time	Number of request, CPU usage and throughput	On- demand	-
[17]	Proactive	Horizontal	TSA: ARIMA	QoS aware	simulated	Response time	Request rate, RMSE	On- demand	CloudSim toolkit
[93]	Proactive	Horizontal	TSA: ARIMA + GA	Resource Aware	simulated	Response time, Cost	Request rate, Bootstrap, Mean Elasticity Index (MEI), RMSE, MSE	On- demand	Custom testbed in cloud
[29]	Proactive	Horizontal	TSA: Holt model + Reverse trend (RT) + GA + Fuzzy logic	Load Balance aware	controlled environment	Execution time, Number of migrations	Average and standard error and Breakdown	On- demand	Custom testbed using two clusters with Globus toolkit
[6]	Proactive	Horizontal	TSA: Double Exponential Smoothing (DES)	Cost aware	Custom	Resources and SLA	Number of requests	On- demand	CloudSim
[7]	Hybrid	Horizontal	TSA: Weight moving average (WMA)	cost aware	Custom	cost	Number of requests + CPU utilization	On- demand	CloudSim
[72]	Proactive	-	TSA: ANN + Differential evolution	Workload aware	Prediction interval (1, 5, 10, 20, 30, 60)	-	Arrival rate, RMSE	On-demand	Sumlation in Matlab. NASA and Saskatchewan traces
[69]	Proactive	-	TSA: CloudInsight	Workload aware	-	-	No. of requests, Prediction overhead, Commutative distribution function (CDF)	-	Python 2.7 on Ubuntu 16.07. Google workload, Wiki, Facebook dataset
[125]	Proactive	Horizontal	TSA: LR, ARIMA and SVR	Resource aware	Custom	Prediction accuracy and resources	Number of request	On- demand	R-tool
[61]	Hybrid	Horizontal	TSA: QTH + Continuous Time Markov Chain (CTMC) + AR	Resource aware	Custom	Response time	Arrival rate, VMs allocated, Resource utilization	On-demand	Real experiments on Amazon EC2. RUBiS, RUBBoS, Cassandra and Olio benchmarking applications

6.7. Energy aware Auto-scaling. The existing work mostly focused on cost optimization and QoS requirement. The data center is also facing environmental issues also. Solar datacenters and renewable energy resources in the data center can reduce the carbon problem. So the carbon and energy-aware auto-scaling provide the preference to the environmental friendly data centers in a single and multi-cloud environment. The weather condition and maximum solar power generating data centers give the highest priority for resource allocation.

6.8. Bin-packing Auto-scaling. The bin packing approach offers potential research topics in auto-scaling of web applications in cloud computing. The size of bins could vary, which makes this approach more robust. Resource provisioning of bins is lightweight and gives cost benefits.

7. Conclusions. Auto-scaling technique provisions the resources as per the incoming workloads. The application providers reap the benefit with this technique in terms of cost while maintaining the QoS. However, the auto-scaling design and development process having number of challenges. In literature, authors proposed the auto-scaling techniques with various characteristics in order to overcome the problems in auto-scaling in cloud.

In this article, a literature survey of the state-of-the-art auto-scaling techniques has carried out and, identified the key challenges in auto-scaling of multi-tier web applications. Moreover, the auto-scaling techniques are classified and taxonomy is presented based on various characteristics. Furthermore, the future research directions have been proposed based on the key challenges.

REFERENCES

[1] Ali-Eldin, A., Kihl, M., Tordsson, J., and Elmroth, E. (2012a). Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In *Proceedings of the 3rd workshop on Scientific Cloud Computing Date*, pages 31–40. ACM.

[2] Ali-Eldin, A., Tordsson, J., and Elmroth, E. (2012b). An adaptive hybrid elasticity controller for cloud infrastructures. In *2012 IEEE Network Operations and Management Symposium*, pages 204–212. IEEE.

[3] Alzubi, J., Nayyar, A., and Kumar, A. (2018). Machine learning from theory to algorithms: an overview. In *Journal of Physics: Conference Series*, volume 1142, page 012012. IOP Publishing.

[4] Amazon (2019). Amazon EC2. `https://aws.amazon.com/ec2/`. [Online; accessed 30-March-2019].

[5] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., et al. (2009). Above the clouds: A berkeley view of cloud computing.

[6] Aslanpour, M. S. and Dashti, S. E. (2017). Proactive auto-scaling algorithm (pasa) for cloud application. *International Journal of Grid and High Performance Computing (IJGHPC)*, 9(3):1–16.

[7] Aslanpour, M. S., Ghobaei-Arani, M., and Toosi, A. N. (2017). Auto-scaling web applications in clouds: a cost-aware approach. *Journal of Network and Computer Applications*, 95:26–41.

[8] Bacigalupo, D. A., van Hemert, J., Usmani, A., Dillenberger, D. N., Wills, G. B., and Jarvis, S. A. (2010). Resource management of enterprise cloud systems using layered queuing and historical performance models. In *Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1–8. IEEE.

[9] Barrett, E., Howley, E., and Duggan, J. (2013). Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Concurrency and Computation: Practice and Experience*, 25(12):1656–1674.

[10] Bauer, A., Herbst, N., Spinner, S., Ali-Eldin, A., and Kounev, S. (2019). Chameleon: A hybrid, proactive auto-scaling mechanism on a level-playing field. *IEEE Transactions on Parallel and Distributed Systems*, 30(4):800–813.

[11] Beltrán, M. (2015). Automatic provisioning of multi-tier applications in cloud computing environments. *The Journal of Supercomputing*, 71(6):2221–2250.

[12] Benifa, J. B. and Dejey, D. (2018). Rlpas: Reinforcement learning-based proactive auto-scaler for resource provisioning in cloud environment. *Mobile Networks and Applications*, pages 1–16.

[13] Bhardwaj, T. and Sharma, S. C. (2018). Fuzzy logic-based elasticity controller for autonomic resource provisioning in parallel scientific applications: a cloud computing perspective. *Computers & Electrical Engineering*, 70:1049–1073.

[14] Bodik, P., Griffith, R., Sutton, C., Fox, A., Jordan, M., and Patterson, D. (2009). Statistical machine learning makes automatic control practical for internet datacenters. In *Proceedings of the 2009 conference on Hot topics in cloud computing*, pages 12–12.

[15] Brown, R. G. and Meyer, R. F. (1961). The fundamental theorem of exponential smoothing. *Operations Research*, 9(5):673–685.

[16] Bu, X., Rao, J., and Xu, C.-Z. (2013). Coordinated self-configuration of virtual machines and appliances using a model-free learning approach. *IEEE transactions on parallel and distributed systems*, 24(4):681–690.

[17] Calheiros, R. N., Masoumi, E., Ranjan, R., and Buyya, R. (2015). Workload prediction using arima model and its impact on cloud applications qos. *IEEE Transactions on Cloud Computing*, 3(4):449–458.

[18] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience*, 41(1):23–50.

[19] Caron, E., Desprez, F., and Muresan, A. (2011). Pattern matching based forecast of non-periodic repetitive behavior for cloud clients. *Journal of Grid Computing*, 9(1):49–64.

[20] Casalicchio, E. and Silvestri, L. (2013). Autonomic management of cloud-based systems: the service provider perspective. In *Computer and Information Sciences III*, pages 39–47. Springer.

[21] Chandra, A., Gong, W., and Shenoy, P. (2003). Dynamic resource allocation for shared data centers using online measurements. In *International Workshop on Quality of Service*, pages 381–398. Springer.

[22] Chen, G., He, W., Liu, J., Nath, S., Rigas, L., Xiao, L., and Zhao, F. (2008). Energy-aware server provisioning and load dispatching for connection-intensive internet services. In *NSDI*, volume 8, pages 337–350.

[23] Chen, T., Bahsoon, R., and Yao, X. (2018). A survey and taxonomy of self-aware and self-adaptive cloud autoscaling systems. *ACM Computing Surveys (CSUR)*, 51(3):61.

[24] Chieu, T. C., Mohindra, A., and Karve, A. A. (2011). Scalability and performance of web applications in a compute cloud. In *e-Business Engineering (ICEBE), 2011 IEEE 8th International Conference on*, pages 317–323. IEEE.

[25] Chieu, T. C., Mohindra, A., Karve, A. A., and Segal, A. (2009). Dynamic scaling of web applications in a virtualized cloud computing environment. In *e-Business Engineering, 2009. ICEBE'09. IEEE International Conference on*, pages 281–286. IEEE.

[26] Cohen, J. W. (2012). *The single server queue*, volume 8. Elsevier.

[27] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). Introduction to algorithms second edition.

[28] Coutinho, E. F., de Carvalho Sousa, F. R., Rego, P. A. L., Gomes, D. G., and de Souza, J. N. (2015). Elasticity in cloud computing: a survey. *annals of telecommunications-annales des télécommunications*, 70(7-8):289–309.

[29] De Grande, R. E., Boukerche, A., and Alkharboush, R. (2017). Time series-oriented load prediction model and migration policies for distributed simulation systems. *IEEE Transactions on Parallel and Distributed Systems*, 28(1):215–229.

[30] Dejun, J., Pierre, G., and Chi, C.-H. (2011). Resource provisioning of web applications in heterogeneous clouds. In *Proceedings of the 2nd USENIX conference on Web application development*, pages 5–5. USENIX Association.

[31] Douglas, K. B. (2003). Web services and service-oriented architectures: the savvy manager's guide.

[32] Dutreilh, X., Kirgizov, S., Melekhova, O., Malenfant, J., Rivierre, N., and Truck, I. (2011). Using reinforcement learning for autonomic resource allocation in clouds: towards a fully automated workflow. In *ICAS 2011, The Seventh International Conference on Autonomic and Autonomous Systems*, pages 67–74.

[33] Dutreilh, X., Moreau, A., Malenfant, J., Rivierre, N., and Truck, I. (2010). From data center resource allocation to control theory and back. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 410–417. IEEE.

[34] Dutta, S., Gera, S., Verma, A., and Viswanathan, B. (2012). Smartscale: Automatic application scaling in enterprise clouds. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 221–228. IEEE.

[35] Evangelidis, A., Parker, D., and Bahsoon, R. (2018). Performance modelling and verification of cloud-based auto-scaling policies. *Future Generation Computer Systems*, 87:629–638.

[36] Fallah, M., Arani, M. G., and Maeen, M. (2015). Nasla: Novel auto scaling approach based on learning automata for web application in cloud computing environment. *International Journal of Computer Applications*, 113(2).

[37] Fang, W., Lu, Z., Wu, J., and Cao, Z. (2012). Rpps: a novel resource prediction and provisioning scheme in cloud data center. In *Services Computing (SCC), 2012 IEEE Ninth International Conference on*, pages 609–616. IEEE.

[38] Farokhi, S., Jamshidi, P., Lakew, E. B., Brandic, I., and Elmroth, E. (2016). A hybrid cloud controller for vertical memory elasticity: A control-theoretic approach. *Future Generation Computer Systems*, 65:57–72.

[39] Fernandez, H., Pierre, G., and Kielmann, T. (2014). Autoscaling web applications in heterogeneous cloud infrastructures. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 195–204. IEEE.

[40] Frey, S., Lüthje, C., Reich, C., and Clarke, N. (2014). Cloud qos scaling by fuzzy logic. In *Cloud Engineering (IC2E), 2014 IEEE International Conference on*, pages 343–348. IEEE.

[41] Gambi, A. and Toffetti, G. (2012). Modeling cloud performance with kriging. In *Proceedings of the 34th International Conference on Software Engineering*, pages 1439–1440. IEEE Press.

[42] Gandhi, A., Dube, P., Karve, A., Kochut, A., and Zhang, L. (2014). Adaptive, model-driven autoscaling for cloud applications. In *11th International Conference on Autonomic Computing (ICAC 14)*, pages 57–64.

[43] Gandhi, A., Harchol-Balter, M., Raghunathan, R., and Kozuch, M. A. (2012). Autoscale: Dynamic, robust capacity management for multi-tier data centers. *ACM Transactions on Computer Systems (TOCS)*, 30(4):14.

[44] Gandhi, A., Dube, P., Karve, A., Kochut, A., and Zhang, L. (2018). Model-driven optimal resource scaling in cloud. *Software & Systems Modeling*, 17(2):509–526.

[45] Ghanbari, H., Simmons, B., Litoiu, M., and Iszlai, G. (2011). Exploring alternative approaches to implement an elasticity policy. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 716–723. IEEE.

[46] Gong, Z., Gu, X., and Wilkes, J. (2010). Press: Predictive elastic resource scaling for cloud systems. In *2010 International Conference on Network and Service Management*, pages 9–16. IEEE.

[47] Grimaldi, D., Pescape, A., Salvi, A., Persico, V., et al. (2017). A fuzzy approach based on heterogeneous metrics for scaling out public clouds. *IEEE Transactions on Parallel and Distributed Systems*.

[48] Grozev, N. and Buyya, R. (2014). Multi-cloud provisioning and load distribution for three-tier applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 9(3):13.

[49] Guitart, J., Torres, J., and Ayguadé, E. (2010). A survey on performance management for internet applications. *Concurrency and Computation: Practice and Experience*, 22(1):68–106.

[50] Han, R., Ghanem, M. M., Guo, L., Guo, Y., and Osmond, M. (2014). Enabling cost-aware and adaptive elasticity of

multi-tier cloud applications. *Future Generation Computer Systems*, 32:82–98.

[51] Han, R., Guo, L., Ghanem, M. M., and Guo, Y. (2012). Lightweight resource scaling for cloud applications. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, pages 644–651. IEEE.

[52] Hasan, M. Z., Magana, E., Clemm, A., Tucker, L., and Gudreddi, S. L. D. (2012). Integrated and autonomic cloud resource scaling. In *2012 IEEE Network Operations and Management Symposium*, pages 1327–1334. IEEE.

[53] Heinze, T., Pappalardo, V., Jerzak, Z., and Fetzer, C. (2014). Auto-scaling techniques for elastic data stream processing. In *Data Engineering Workshops (ICDEW), 2014 IEEE 30th International Conference on*, pages 296–302. IEEE.

[54] Huang, D., He, B., and Miao, C. (2014). A survey of resource management in multi-tier web applications. *IEEE Communications Surveys & Tutorials*, 16(3):1574–1590.

[55] Huang, J., Li, C., and Yu, J. (2012). Resource prediction based on double exponential smoothing in cloud computing. In *Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on*, pages 2056–2060. IEEE.

[56] Huang, G., Wang, S., Zhang, M., Li, Y., Qian, Z., Chen, Y., and Zhang, S. (2016). Auto scaling virtual machines for web applications with queueing theory. In *2016 3rd International Conference on Systems and Informatics (ICSAI)*, pages 433–438. IEEE.

[57] Iqbal, W., Dailey, M. N., Carrera, D., and Janecek, P. (2011). Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Generation Computer Systems*, 27(6):871–879.

[58] Islam, S., Keung, J., Lee, K., and Liu, A. (2012). Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155–162.

[59] Jamshidi, P., Pahl, C., and Mendonça, N. C. (2016). Managing uncertainty in autonomic cloud elasticity controllers. *IEEE Cloud Computing*, 3(3):50–60.

[60] Jin, Y., Bouzid, M., Kostadinov, D., and Aghasaryan, A. (2018). Testing a q-learning approach for derivation of scaling policies in cloud-based applications. In *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, pages 1–3. IEEE.

[61] JV, B. B. and Dharma, D. (2018). Has: Hybrid auto-scaler for resource scaling in cloud environment. *Journal of Parallel and Distributed Computing*, 120:1–15.

[62] Kalyvianaki, E., Charalambous, T., and Hand, S. (2009). Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In *Proceedings of the 6th international conference on Autonomic computing*, pages 117–126. ACM.

[63] Kaur, G., Bala, A., and Chana, I. (2019). An intelligent regressive ensemble approach for predicting resource usage in cloud computing. *Journal of Parallel and Distributed Computing*, 123:1–12.

[64] Keilson, J. and Servi, L. (1988). A distributional form of little's law. *Operations Research Letters*, 7(5):223–227.

[65] Kendall, D. G. (1953). Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, pages 338–354.

[66] Khatua, S., Ghosh, A., and Mukherjee, N. (2010). Optimizing the utilization of virtual resources in cloud environment. In *2010 IEEE International Conference on Virtual Environments, Human-Computer Interfaces and Measurement Systems*, pages 82–87. IEEE.

[67] Kim, H., el Khamra, Y., Jha, S., and Parashar, M. (2009a). An autonomic approach to integrated hpc grid and cloud usage. In *e-Science, 2009. e-Science'09. Fifth IEEE International Conference on*, pages 366–373. IEEE.

[68] Kim, H., Parashar, M., Foran, D. J., and Yang, L. (2009b). Investigating the use of autonomic cloudbursts for high-throughput medical image registration. In *2009 10th IEEE/ACM International Conference on Grid Computing*, pages 34–41. IEEE.

[69] Kim, I. K., Wang, W., Qi, Y., and Humphrey, M. (2018). Cloudinsight: Utilizing a council of experts to predict future cloud application workloads. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, pages 41–48. IEEE.

[70] Koperek, P. and Funika, W. (2011). Dynamic business metrics-driven resource provisioning in cloud environments. In *International Conference on Parallel Processing and Applied Mathematics*, pages 171–180. Springer.

[71] Kupferman, J., Silverman, J., Jara, P., and Browne, J. (2009). Scaling into the cloud. *CS270-advanced operating systems*.

[72] Kumar, J. and Singh, A. K. (2018). Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems*, 81:41–52.

[73] Lama, P. and Zhou, X. (2009). Efficient server provisioning with end-to-end delay guarantee on multi-tier clusters. In *Quality of Service, 2009. IWQoS. 17th International Workshop on*, pages 1–9. IEEE.

[74] Lama, P. and Zhou, X. (2013). Autonomic provisioning with self-adaptive neural fuzzy control for percentile-based delay guarantee. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 8(2):9.

[75] Lango, J. (2014). Toward software-defined slas. *Communications of the ACM*, 57(1):54–60.

[76] Lim, H. C., Babu, S., and Chase, J. S. (2010). Automated control for elastic storage. In *Proceedings of the 7th international conference on Autonomic computing*, pages 1–10. ACM.

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

[78] Liu, C., Liu, C., Shang, Y., Chen, S., Cheng, B., and Chen, J. (2017). An adaptive prediction approach based on workload pattern discrimination in the cloud. *Journal of Network and Computer Applications*, 80:35–44.

[79] Liu, X., Yuan, S.-M., Luo, G.-H., Huang, H.-Y., and Bellavista, P. (2017). Cloud resource management with turnaround time driven auto-scaling. *IEEE Access*, 5:9831–9841.

[80] Liu, B., Buyya, R., and Toosi, A. N. (2018a). A fuzzy-based auto-scaler for web applications in cloud computing environ-

ments. In *International Conference on Service-Oriented Computing*, pages 797–811. Springer.

[81] Liu, X., Dastjerdi, A. V., Calheiros, R. N., Qu, C., and Buyya, R. (2018b). A stepwise auto-profiling method for performance optimization of streaming applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(4):24.

[82] Liu, X., Dastjerdi, A. V., Calheiros, R. N., Qu, C., and Buyya, R. (2018). A stepwise auto-profiling method for performance optimization of streaming applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 12(4):24.

[83] Lombardi, F., Muti, A., Aniello, L., Baldoni, R., Bonomi, S., and Querzoni, L. (2019). Pascal: An architecture for proactive auto-scaling of distributed services. *Future Generation Computer Systems*.

[84] Lorido-Botrán, T., Miguel-Alonso, J., and Lozano, J. A. (2012). Auto-scaling techniques for elastic applications in cloud environments. *Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09-12*.

[85] Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J. A. (2014). A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing*, 12(4):559–592.

[86] Lorido-Botrcn, T., Miguel-Alonso, J., and Lozano, J. A. (2013). Comparison of auto-scaling techniques for cloud environments.

[87] Mahallat, I. (2015). Astaw: Auto-scaling threshold-based approach for web application in cloud computing environment.

[88] Manvi, S. S. and Shyam, G. K. (2014). Resource management for infrastructure as a service (iaas) in cloud computing: A survey. *Journal of Network and Computer Applications*, 41:424–440.

[89] Mao, M. and Humphrey, M. (2011). Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pages 1–12. IEEE.

[90] Mao, M. and Humphrey, M. (2012). A performance study on the vm startup time in the cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 423–430. IEEE.

[91] Maurer, M., Brandic, I., and Sakellariou, R. (2011a). Enacting slas in clouds using rules. In *European Conference on Parallel Processing*, pages 455–466. Springer.

[92] Maurer, M., Breskovic, I., Emeakaroha, V. C., and Brandic, I. (2011b). Revealing the mape loop for the autonomic management of cloud infrastructures. In *Computers and Communications (ISCC), 2011 IEEE Symposium on*, pages 147–152. IEEE.

[93] Messias, V. R., Estrella, J. C., Ehlers, R., Santana, M. J., Santana, R. C., and Reiff-Marganiec, S. (2016). Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure. *Neural Computing and Applications*, 27(8):2383–2406.

[94] Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., and Yuan, L. (2010). Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 514–521. IEEE.

[95] Moghaddam, S. K., Buyya, R., and Ramamohanarao, K. (2019). Acas: An anomaly-based cause aware auto-scaling framework for clouds. *Journal of Parallel and Distributed Computing*, 126:107–120.

[96] Nayyar, A. (2011). Private virtual infrastructure (pvi) model for cloud computing. *International Journal of Software Engineering Research and Practices*, 1(1):10–14.

[97] Nayyar, A., Puri, V., et al. (2017). Comprehensive analysis & performance comparison of clustering algorithms for big data. *Review of Computer Engineering Research*, 4(2):54–80.

[98] Nguyen, H., Shen, Z., Gu, X., Subbiah, S., and Wilkes, J. (2013). Agile: Elastic distributed resource scaling for infrastructure-as-a-service. In *ICAC*, volume 13, pages 69–82.

[99] Nikravesh, A. Y., Ajila, S. A., and Lung, C.-H. (2014). Cloud resource auto-scaling system based on hidden markov model (hmm). In *Semantic Computing (ICSC), 2014 IEEE International Conference on*, pages 124–127. IEEE.

[100] Nikravesh, A. Y., Ajila, S. A., and Lung, C.-H. (2017). An autonomic prediction suite for cloud resource provisioning. *Journal of Cloud Computing*, 6(1):3.

[101] Nouri, S. M. R., Li, H., Venugopal, S., Guo, W., He, M., and Tian, W. (2019). Autonomic decentralized elasticity based on a reinforcement learning controller for cloud applications. *Future Generation Computer Systems*, 94:765–780.

[102] Padala, P., Hou, K.-Y., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., and Merchant, A. (2009). Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 13–26. ACM.

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

[104] Patikirikorala, T. and Colman, A. (2010). Feedback controllers in the cloud. In *Proceedings of APSEC*.

[105] Persico, V., Grimaldi, D., Pescape, A., Salvi, A., and Santini, S. (2017). A fuzzy approach based on heterogeneous metrics for scaling out public clouds. *IEEE Transactions on Parallel and Distributed Systems*, 28(8):2117–2130.

[106] Prodan, R. and Nae, V. (2009). Prediction-based real-time resource provisioning for massively multiplayer online games. *Future Generation Computer Systems*, 25(7):785–793.

[107] Qu, C., Calheiros, R. N., and Buyya, R. (2016a). Auto-scaling web applications in clouds: a taxonomy and survey. *arXiv preprint arXiv:1609.09224*.

[108] Qu, C., Calheiros, R. N., and Buyya, R. (2016b). A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances. *Journal of Network and Computer Applications*, 65:167–180.

[109] Qu, C., Calheiros, R. N., and Buyya, R. (2018). Auto-scaling web applications in clouds: A taxonomy and survey. *ACM Computing Surveys (CSUR)*, 51(4):73.

[110] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.
[111] Rao, J., Bu, X., Xu, C.-Z., and Wang, K. (2011). A distributed self-learning approach for elastic provisioning of virtualized cloud resources. In *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 45–54. IEEE.
[112] Rao, J., Bu, X., Xu, C.-Z., Wang, L., and Yin, G. (2009). Vconf: a reinforcement learning approach to virtual machines auto-configuration. In *Proceedings of the 6th international conference on Autonomic computing*, pages 137–146. ACM.
[113] RightScale (2019). RightScale Universal Cloud Management. `http://www.rightscale.com/`. [Online; accessed 30-March-2019].
[114] Roy, N., Dubey, A., and Gokhale, A. (2011). Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 500–507. IEEE.
[115] Runsewe, O. and Samaan, N. (2018). Cloud resource scaling for time-bounded and unbounded big data streaming applications. *IEEE Transactions on Cloud Computing*.
[116] Salah, K., Elbadawi, K., and Boutaba, R. (2016). An analytical model for estimating cloud resources of elastic services. *Journal of Network and Systems Management*, 24(2):285–308.
[117] Sedaghat, M., Hernandez-Rodriguez, F., and Elmroth, E. (2013). A virtual machine re-packing approach to the horizontal vs. vertical elasticity trade-off for cloud autoscaling. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, page 6. ACM.
[118] Shah, N. B., Shah, N. D., Bhatia, J., and Trivedi, H. (2019). Profiling-based effective resource utilization in cloud environment using divide and conquer method. In *Information and Communication Technology for Competitive Strategies*, pages 495–508. Springer.
[119] Sharma, U., Shenoy, P., Sahu, S., and Shaikh, A. (2011). A cost-aware elasticity provisioning system for the cloud. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 559–570. IEEE.
[120] Shah, N. B., Shah, N. D., Bhatia, J., and Trivedi, H. (2019). Profiling-based effective resource utilization in cloud environment using divide and conquer method. In *Information and Communication Technology for Competitive Strategies*, pages 495–508. Springer.
[121] Shekhar, S., Barve, Y., and Gokhale, A. (2017). Understanding performance interference benchmarking and application profiling techniques for cloud-hosted latency-sensitive applications. In *Proceedings of the10th International Conference on Utility and Cloud Computing*, pages 187–188. ACM.
[122] Shen, Z., Subbiah, S., Gu, X., and Wilkes, J. (2011). Cloudscale: elastic resource scaling for multi-tenant cloud systems. In *Proceedings of the 2nd ACM Symposium on Cloud Computing*, page 5. ACM.
[123] Simmons, B., Ghanbari, H., Litoiu, M., and Iszlai, G. (2011). Managing a saas application in the cloud using paas policy sets and a strategy-tree. In *Proceedings of the 7th International Conference on Network and Services Management*, pages 343–347. International Federation for Information Processing.
[124] Singh, S. P., Nayyar, A., Kumar, R., and Sharma, A. (2018). Fog computing: from architecture to edge computing and big data processing. *The Journal of Supercomputing*, pages 1–36.
[125] Singh, P., Gupta, P., and Jyoti, K. (2018). Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. *Cluster Computing*, pages 1–15.
[126] Sutton, R. S. and Barto, A. G. (1998). *Introduction to reinforcement learning*, volume 135. MIT Press Cambridge.
[127] Tesauro, G., Jong, N. K., Das, R., and Bennani, M. N. (2006). A hybrid reinforcement learning approach to autonomic resource allocation. In *2006 IEEE International Conference on Autonomic Computing*, pages 65–73. IEEE.
[128] Toffetti, G., Brunner, S., Blöchlinger, M., Spillner, J., and Bohnert, T. M. (2017). Self-managing cloud-native applications: Design, implementation, and experience. *Future Generation Computer Systems*, 72:165–179.
[129] Toosi, A. N., Son, J., Chi, Q., and Buyya, R. (2019). Elasticsfc: Auto-scaling techniques for elastic service function chaining in network functions virtualization-based clouds. *Journal of Systems and Software*.
[130] Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., and Wood, T. (2008). Agile dynamic provisioning of multi-tier internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1):1.
[131] Varia, J. and Mathew, S. (2014). Overview of amazon web services. *Amazon Web Services*.
[132] Vasić, N., Novaković, D., Miučin, S., Kostić, D., and Bianchini, R. (2012). Dejavu: accelerating resource allocation in virtualized environments. In *ACM SIGARCH computer architecture news*, volume 40, pages 423–436. ACM.
[133] Vilaplana, J., Solsona, F., Teixidó, I., Mateo, J., Abella, F., and Rius, J. (2014). A queuing theory model for cloud computing. *The Journal of Supercomputing*, 69(1):492–507.
[134] Villela, D., Pradhan, P., and Rubenstein, D. (2007). Provisioning servers in the application tier for e-commerce systems. *ACM Transactions on Internet Technology (TOIT)*, 7(1):7.
[135] Vondra, T. and Šedivỳ, J. (2017). Cloud autoscaling simulation based on queueing network model. *Simulation Modelling Practice and Theory*, 70:83–100.
[136] Wang, L., Xu, J., Zhao, M., Tu, Y., and Fortes, J. A. (2011). Fuzzy modeling based resource management for virtualized database systems. In *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pages 32–42. IEEE.
[137] Wei, Y., Kudenko, D., Liu, S., Pan, L., Wu, L., and Meng, X. (2019). A reinforcement learning based auto-scaling approach for saas providers in dynamic cloud environment. *Mathematical Problems in Engineering*, 2019.
[138] Xing, J., Zhou, H., Shen, J., Zhu, K., Wangt, Y., Wu, C., and Ruan, W. (2018). Asidps: Auto-scaling intrusion detection and prevention system for cloud. In *2018 25th International Conference on Telecommunications (ICT)*, pages 207–212. IEEE.
[139] Xu, C.-Z., Rao, J., and Bu, X. (2012). Url: A unified reinforcement learning approach for autonomic cloud management.

Journal of Parallel and Distributed Computing, 72(2):95–105.

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

[141] Zhang, Q., Cherkasova, L., and Smirni, E. (2007). A regression-based analytic model for dynamic resource provisioning of multi-tier applications. In *Fourth International Conference on Autonomic Computing (ICAC'07)*, pages 27–27. IEEE.

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

Edited by: Pijush Kanti Dutta Pramanik
Received: Mar 17, 2019
Accepted: Apr 4, 2019