Cluster Computing
https://doi.org/10.1007/s10586-020-03148-5

RHAS: robust hybrid auto-scaling for web applications in cloud computing

Parminder Singh[1] · Avinash Kaur[1] · Pooja Gupta[1] · Sukhpal Singh Gill[2] · Kiran Jyoti[3]

Received: 27 December 2018 / Revised: 15 June 2020 / Accepted: 26 June 2020

Abstract
The elasticity characteristic of cloud services attracts application providers to deploy applications in a cloud environment. The scalability feature of cloud computing gives the facility to application providers to dynamically provision the computing power and storage capacity from cloud data centers. The consolidation of services to few active servers can enhance the service sustainability and reduce the operational cost. The state-of-art algorithms mostly focus either on reactive or proactive auto-scaling techniques. In this article, a Robust Hybrid Auto-Scaler (RHAS) is presented for web applications. The time series forecasting model has been used to predict the future incoming workload. The reactive approach is used to deal with the current resource requirement. The proposed auto-scaling technique is designed with the threshold-based rules and queuing model. The security mechanism is used to secure the user's request and response to the web-applications deployed in cloud environment. The designed approach has been tested with two real-time web application workloads of ClarkNet and NASA. The proposed technique achieves 14% reduction in cost, and significant improvement in response time, service level agreement (SLA) violation, and gives consistency in CPU utilization.

Keywords Auto-scaling · Cloud computing · Web applications · Resource provisioning · Time series prediction · Cloud Security

1 Introduction

Cloud computing provides infrastructure resources, storage, and computing through web services [55]. The large-scale applications are mostly host and managed on Infrastructure-as-a-Service (IaaS) model [44]. The infrastructure is allowed to control and manage by the users with programming [68]. This helps the application providers (APs) to deploy the applications in a cloud environment with the quality of service (QoS) and cost efficient [32,35]. The web applications workload is dynamic which is characterized by flash-workload and time-varying traffic. The performance of applications becomes a challenging job. IaaS feature of cloud computing allows us to scale-up or scale-down the resources on-demand to get the desired response-time for service level agreement (SLA) fulfillment [9]. In a single-tier application, the response time prediction is sufficient to scale the resources [10]. However, the multi-tier applications workload is complex in nature due to the multi-tier architecture and flash traffic [26]. Thus, the operational cost and SLA penalty can be minimized with the autonomic scaling process. The profiling information of low-level hardware such as I/O usage, bandwidth, memory, and CPU is used to find multi-tier web application's bottlenecks theoretically. However, sometimes it is not possible due to security concerns of the application and, adds the complexity to virtualization and decreases the performance of the application

✉ Parminder Singh
parminder.16479@lpu.co.in

Avinash Kaur
avinash.14557@lpu.co.in

Pooja Gupta
dr.pooja.g29@gmail.com

Sukhpal Singh Gill
s.s.gill@qmul.ac.uk

Kiran Jyoti
kiranjyotibains@yahoo.com

[1] School of Computer Science and Engineering, Lovey Professional University, Phagwara, Punjab, India

[2] School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK

[3] Department of Information Technology, Guru Nanak Dev Engineering College, Ludhiana, Punjab, India

Published online: 20 July 2020

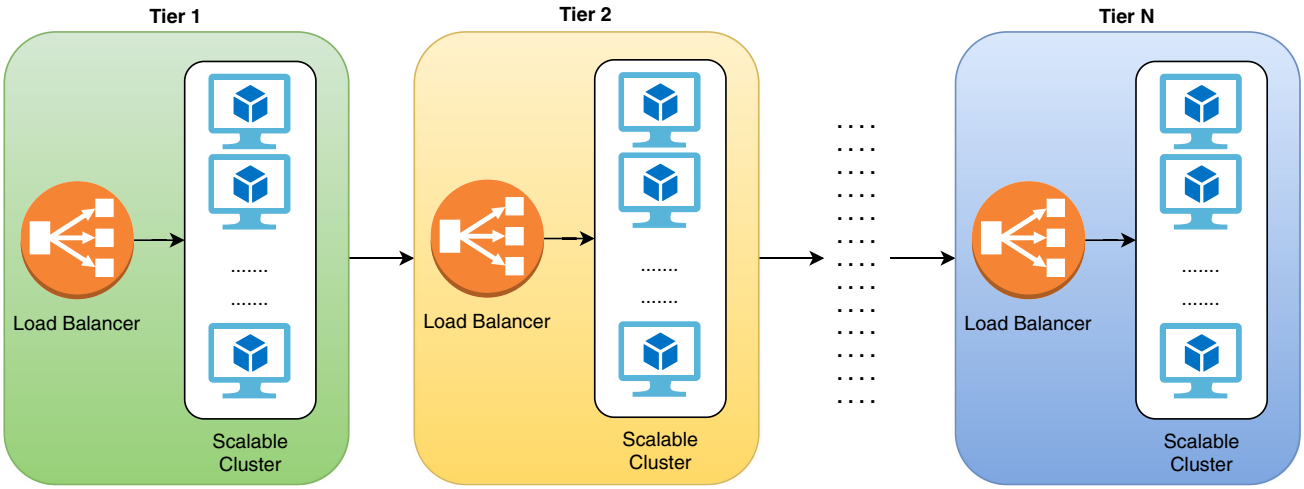


Fig. 1 Multi-tier scalable architecture

in cloud [21]. Furthermore, the software misconfiguration issues are not identifiable at the low-level hardware usage. This misconfiguration has a high impact on the response time of the application. Therefore, we used the fine grained and coarse-grained parameters to identify and resolve the deployed multi-tier web applications in the cloud environment. There are N tiers in a multi-tier application, the virtual machines are dynamically provisioned with load balancer as shown in Fig. 1. There are heuristic methods [31,37] and machine learning techniques [5,36,69] to scale the multi-tier web applications dynamically. However, the variance in virtual machines (VMs) performance and fluctuating traffic of web applications make the predictive auto-scaling a tedious job for application providers to maintain the service level objectives (SLOs).

As per the recent trends in the era of cloud computing, different APs host applications on the cloud instead of buying computing infrastructure. The resources are offered in the form of VMs to APs by cloud providers like Amazon EC2 with add-on features of scalability and pay-per-use model [15,46,58]. The cloud providers are offered three pricing models, named as *on-demand*, *reservation* and *spot* instances. The VMs are provided at a fixed price in the on-demand pricing model and can be acquired periodically. In the reservation policy, APs have to fix the contract period and price for a certain number of VMs. Amazon introduced a spot pricing policy for spare capacity. Amazon sells spare VMs capacity through the bidding mechanism in an open market. The cloud user participates in auction mechanism through a bid defining the maximum per-unit price and VMs number and type. If the bid price is higher then the current spot price, the resources will be allocated to the cloud user. The spot instances can be interrupted for the number of reasons like spot price increases than the maximum bidding price, the capacity is no longer available, or the demand increases for spot instances [19].

The spot instances are cost-effective and suitable for non-time-critical applications and can be interrupted. It is generally believed the spot instances are not suitable for web applications because of their availability and time constraints.

The web APs are concerned about the dynamic features of the web environment and different requests from the end-users, hence static provisioning of resources is not an efficient technique. Therefore, the current resources are not able to cater to all the incoming requests and rise the under-provisioning state. This, in turn, leads to delayed response or interruption of user requests. At another end of a situation where the traffic is reduced, the factor of over-provisioning arises and hence leading to an increase in costs of APs [22,58]. While considering the different models of pricing in cloud [58,61], the minimum number of resources are prepaid by application providers for use of a long or short term in order to receive a discount in the rental (for example, In EC2 reserved instances receives 75 percent discount on rental). Thereafter, with the varying load, application providers prefer to use the short-term rental model in order to cover the temporary needs. However, this method requires a highly capable mechanism for automatic determination of capacity and on-demand resources rented as per the proportion of incoming load [2].

It is essential to provide security to the user's data which makes the techniques robust through cloud data security. The virtual private networks and firewalls make the system robust along with the auto-scaling mechanism because of the sharing of the pool of resources among different users. The numerous state-of-the-art techniques are there for providing security in the events in the cloud and user authentication such as ISO-27001/27002, ITIL and PCI-DSS [25]. Two

concerns are major in cloud security: owner's information is breached by the third party and second is without the owner's permission another person accesses their information. These problems required a robust solution along with an auto-scaling mechanism to make the cloud environment the first choice of entrepreneurs to deploy their applications.

1.1 Motivation and our contributions

In this paper, we designed the *Robust Hybrid Auto-Scaling (RHAS)* policy for web applications in a cloud environment. The technique is carefully designed for cost saving along with QoS. The monitor-analyze-planning-execution (MAPE) loop has significant features to implement the autonomic scaling system to save the renting cost of computing resources. All the features of the MAPE loop used for the implementation of the auto-scaling technique.

The *major contributions* of this article is as follows:

- The design and development of a hybrid analysis approach for resource auto-scaling for Web applications in the cloud.
- The design and development of a hybrid planning approach for scaling decisions in cloud infrastructure.
- A series of experiments are conducted for the performance evaluation of the proposed approach under real-world workload traces for different metrics.

1.2 Article structure

The rest of this article is organized as follows: Sections 2 and 3 presents a brief overview and the state of the art of auto-scaling techniques respectively. Section 4 presents our proposed robust hybrid auto-scaler (RHAS) for mutli-tier web applications in cloud computing. Section 5 provides the performance evaluation and experimental results. Finally, Sect. 7 presents the concluding remarks and future research directions.

2 Background

Auto-scaling is a technique to dynamically adjust 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 renting cost. The auto-scaling challenge for the web applications is to dynamically grow or shrink the resources to meet fluctuated workload requirements. Autonomous scaling techniques work without human intervention. Autonomic systems are self-(configuring-optimizing-protecting-healing) [40]. The auto-scaling following the MAPE-K loop: Monitoring (M), Analysis (A), Planning (P) and Execution (E), knowledge(k) [13, 49] shown in Fig. 2.

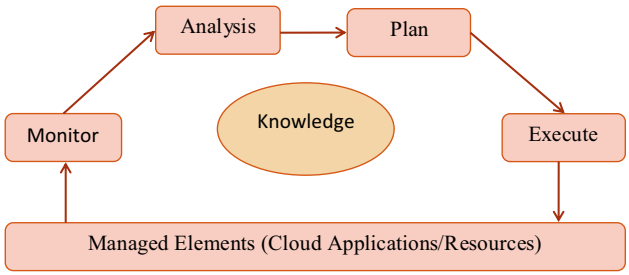


Fig. 2 K-MAPE loop

1. **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. Ghanbari et al. [27] suggested parameters such as resize numbers, operating interval, decision duration, decision threshold, refractory period and instance bounds. 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).
2. **Analysis** The collected information is further processed in the analysis phase. The current system utilization and historical workload are combinedly used to estimate the required resources. 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 simple 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 [48]. Flash crowds and events are still a challenge with the reactive approach.
3. **Planning** The 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.
4. **Execution** The 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.

5. **Knowledge** The knowledge to be shared among above all four functions stored in this repository. The shared knowledge contains metrics, historical logs, topology information, and policies. The information passed to the autonomic manager.

3 Related work

In the literature, the multi-tier web applications workload is usually predicted with the help time-series approach. A simple moving average model gives poor results [30], so the moving average (MA) used to remove the time series noise [43,56]. Huang et al. [34] devised a resource prediction model using double exponential smoothing, and simple mean and weighted moving average (WMA) applied for comparison. Exponential smoothing (ES) significantly gives better results because of history records w. Mi et al. [51] applied Brown's double ES to predict the workload and achieve a good result for the HTTP workload with a small error. Aslanpour et al. [3] applied double exponential smoothing (DES) and weighted moving average (WMA) for the prediction of time series.

The auto-regression technique has also applied for workload and resource prediction [11,12,30,41,59]. Roy et al. [59] used the autoregression (AR) model for workload forecasting by taking the previous three observations. Further response time estimated from the predicted values. An optimization controller applied to find resource allocation, considering SLA violation cost, reconfiguration, and leasing resources. Kupferman et al. [41] used AR(order 1) to forecast the requests per second, and concluded that its performance highly lies on many manager-defined parameters (e.g. Size of history window, size of adaption window, monitoring-interval length). The forecasting is determined for short-term and long-term trends, it is highly dependent on the size of the history window. The adaptation window determines the future model extension.

ARMA model is a simple and efficient model to predict future workload (number of requests). Fang et al. [23] predict VMs CPU usage. ARIMA model is applied in various articles [7,50,60]. ARIMA required a historical workload. The performance of the model highly depends upon the history window. ARIMA approach is ideal for dynamic workload such as web applications. Sedaghat et al. [60] applied the horizontal and vertical scaling to increase the benefit in terms of cost. Mao and Humphrey [47] used the classification given as increasing, stable, seasonal and on/off. Repacking (or reconfiguration) of VMs to provide certain capacity, and repacking of an application based on workload has been done. The approach then finds the optimal pack of VMs and applications. The proposed approach is able to save 7% to 60% cost for resource utilization. The container-based approach can further be optimized by considering more QoS parameters.

Calheiros et al. [7] used ARIMA model for workload prediction, and evaluate the impact on different QoS parameters. The web application workload is dynamic and contains seasonal data. The model gives 91% accuracy for non-seasonal data, but not fit for the workload contains trend and an irregular component. This work can be further extended using an adaptive approach for classification of workload, and design the heuristic for ARIMA fit function for different classes. As discussed earlier, one model doesn't fit for all types of workload, Messias et al. [50] present GA based approach for time series prediction. Traces of real workload have been used to evaluate the prediction model. A new metric has been introduced in the article named as an elasticity index (EI), which describe the solution optimization. The range of EI varies from (0 to 1), a value near to 1 means the solution is good. The model gives less error as compared to other models. This approach is taking slower with the comparison to LR statistical model to predict the incoming request. The prediction interval set by the author is 1 h. Further, this model can be extended by mapping a few prediction techniques with the specific application and workload pattern. The GA model can also design, particularly for cost, energy, sharing of resources, parameters.

The accuracy of neural network [38,57] and multiple regression equation [6,41,57] model are highly dependent upon on the size of input the history window. Islam et al. [38] used more than one value from the history and got a better result. Kupferman et al. [41] devised the necessity of a balanced size of input history window. Regression of various window sizes applied to find the prediction values. The prediction interval of r is also an important factor. Islam et al. [38] investigate the size of the interval window and found 12 min an appropriate time, because of VM startup time is between 5 and 15 min. Prodan and Nae [57] applied the neural network to forecast the game load for 2 min. In contrast, the neural network is better than MA and ES in terms of accuracy.

Horizontal and vertical scaling is also an important factor considers by many authors in literature. It can be either taken separately or in a hybrid manner also. Dutta et al. [18] investigated that horizontal scaling has higher configuration cost as compare to vertical scaling, but relatively gives high throughput. The author prefers the horizontal scaling. The regression model applied to estimate future workload. Fang et al. [23] applied horizontal scaling (CPU and memory) to handle the flash crowd, whereas vertical scaling applied for irregular traffic.

Proactive time series forecasting can be combined with a reactive approach. Iqbal et al. [37] devised a hybrid model for auto-scaling, the author uses the reactive technique for scale-up and a regression model for scale down. Polynomial regression is used to calculate the number of application-tier and database-tier VM instances.

Many authors [8,30,62] applied the pattern identification technique on time series analysis. Gong et al. [30] proposed an FFT based technique to identify the matching patterns in resource utilization (CPU, RAM, Network and I/O). Auto-correlation, histogram, and auto-regression are used for the comparison. Caron et al. [8] designed the algorithm with pattern identification gives poor performance.

Resource utilization of application is also estimated with the simple histogram in some articles by using mean distribution [11], and the highest frequency in the bin [30]. The dynamic load balancer is introduced using Holt's approach by using current and historical data [17]. This work can be used with web workload, may give prediction efficiency and load balancing issues may be resolved. The detailed comparison of existing techniques mentioned in Table 1.

The researchers have proposed many techniques for encryption and decryption [65], cloud security [39,53], privacy-preserving, access control [20,52,67] and secured data storage [25]. The access control to the user's data plays a vital role in the security of web applications in the cloud. Mandatory Access Control (MAC), Discretionary Access Control (DAC) and Role-based Access Control (RBAC) are some known models used by cloud providers with a centralized system. Ganapathy et al. [25] proposed a secured data storage approach to provide better security analysis. In this article, we used the same model to enhance security in the auto-scaling mechanism of web applications.

Time series analysis techniques are able to forecast the future workload of web applications. Further, this information can be used to predict resource requirements. The technique is very appealing because of input workload is known to the auto-scaler in advance, and have enough time to prepare the VMs beforehand. The drawback of techniques is the accuracy, which depends upon the input workload, history window selection, metrics, prediction interval, and target application. There is no best solution for all types of time series forecasting. In this article, we have developed a robust auto-scaling technique with a hybrid approach. The analysis and planning phase carefully designed with classification based prediction model TASM and reactive technique to give QoS while saving the cost for application providers.

4 RHAS: proposed approach

The proposed scaling approach has been discussed in this section. The MAPE loop used for the design of a robust hybrid auto-scaling (RHAS) approach. The goal of this approach is to estimate the required resources in horizontal scaling for the incoming workload. The notations used in this approach listed in Table 2.

4.1 Auto-scaling system architecture

The cloud architecture for web applications as per the proposed approach is shown in Fig. 3. The architecture represents the communication of cloud provider (CP), application provider (AP) and end-user. The end-users communicate to AP via the Internet and send web applications request. The load balancer received the request from AP and send the request to the application tier virtual machine (VM). Generally, the multi-tier web applications have 3−tiers and each tier provisioned on separate VM. The user request passes through each tier to get the response. The user gets the response after this life cycle. The auto-scaling technique defines the required VMs for the incoming workload. This research contributes to the auto-scaling mechanism in the analysis, planning and execution phase. Cloud provider in this model provides the infrastructure with different pricing policy, e.g. Reserved and On-demand.

Algorithm 1 represents the auto-scaling mechanism used in our approach. The first approach used is the monitoring of resources. This approach is unique because it takes the scaling decisions on the basis of AP defined scaling interval ($\triangle s$) in minutes. As per Algorithm 2, monitoring triggers at every minute, and collect the infrastructure and platform level parameters. The monitoring interval is different from the scaling interval because the monitoring interval is a time to collects the environment information, while, the scaling interval defines the time to scale-up and scale-down the resources.

Algorithm 1 The pseudo code of auto-scaling management

```
1: Begin                          ▷ Boot the reserved VMs for incoming workload
2: while system is running do
3:   for every 1 min do
4:     Monitoring();                          ▷ Stores history of metrics
5:     if Clock % △s = 0 then
6:       Analysis(historical w of k size, history of M^u, history of
   M^rt)
7:       Planning(ŵ_{t+1}, A^u, A^rt);
8:       Execution(D);
9:     end if
10:  end for
11: end while
12: End
```

4.2 Monitoring phase

Auto-scaling algorithms are influenced by the dynamic behavior of incoming workload such as seasonality, non-seasonality or flash crowd. As per Algorithm 2, the monitoring phase collects the information on a regular basis (1 min) about the application and infrastructure level parameters [58]. The monitoring takes place every minute. Application-

Table 1 Summary of related work on time series based auto-scaling methods

Work	Policy	Technique	Approach	M–A–P–E	Performance metrics	Experiment
[8]	Proactive	Pattern matching	Load Aware	A	Total number of CPUs, Number of serviced requests, cost	Analytical models
[62]	Hybrid	FFT and Discrete-time Markov Chain	SLO Aware	A–P	CPU load, memory usage, Response time, job progress	–
[59]	Proactive	AR	QoS Aware	A	Number of users in the system, Response time, VM cost, application reconfiguration cost	No experimentation on systems
[37]	Hybrid	Threshold rules and polynomial regression	SLA Aware	M–A	Response time	–
[23]	Proactive	ARMA	SLA Aware	A	Number of requests, CPU load, Prediction accuracy	Custom testbed. Xen and KVM
[34]	Proactive	Brown's double ES. Compared with WMA	Cost Aware	A	CPU load, memory usage, Prediction accuracy	Custom testbed. TPC-W
[38]	Proactive	ML Neural Network and (Multiple) LR + Sliding window	Resource aware	A	CPU load , Prediction accuracy	Real provider. Amazon EC2 and TPC-W application to generate the dataset. Experiments in R-Project.
[18]	Proactive	Polynomial regression	Resource and Cost Aware	A–P	Number of requests, Response time, Cost	Custom testbed. KVM + Olio
[60]	Hybrid	ARIMA + Repacking	Cost-Aware	A–P	Number of request, cost, Response time	Custom simulator
[24]	Hybrid	TSA + Profiler	QoS Aware	A	Number of request, CPU usage, throughput, Response time	–
[7]	Proactive	ARIMA	QoS Aware	A	Request rate, RMSE, Response time	CloudSim toolkit
[50]	Proactive	ARIMA + Genetic Algorithm	Resource Aware	A	Request rate, Bootstrap, Mean Elasticity Index (MEI), RMSE, MSE, Response time, Cost	Custom testbed in cloud
[17]	Proactive	Holt model + Reverse trend (RT) + GA + Fuzzy logic	Load balancing	A–P	Average and standard error and Breakdown, Execution time, Number of migrations	Custom testbed using two clusters with Globus toolkit
[3]	Proactive	Double Exponential Smoothing (DES)	Cost-Aware	A–P	Number of requests, Resource utilization and SLA	CloudSim
[4]	Hybrid	Weight moving average (WMA)	Cost and Load Aware	E	Number of requests + CPU utilization, cost	CloudSim
[63]	Hybrid	LR + ARIMA + SVR + Sliding window	Cost Aware	A	Prediction accuracy (10 min), Resource provisioning	R tool
RHAS (Our Work)	Hybrid	TR + TASM + Sliding window	Robust and QoS Aware	A–P	Prediction accuracy (1 min), Resource provisioning, CPU utilization, response time and etc.	CloudSim and R tool. Real workload traces of ClarkNet and NASA

Table 2 Notations used in the article

Component	Description
A_t^{rt}	Analyzed response time in last minute
A_t^u	Analyzed CPU utilization during $\triangle s$
$Clock$	Simulation timer
D	Scaling decision (e.g. ScaleUp, ScaleDown, DoNothing)
k	Size of sliding window
M_t^{rt}	Average response time at time t
M_t^u	Average CPU utilization at time t
RT^{lowThr}	Lower threshold in response time
RT^{uprThr}	Upper threshold of response time
S^{rt}	Response time as per SLA
$SLAV$	Total SLA violation in an hour
$\triangle s$	Scaling Interval
sw	sliding window of size k
T	Throughput in last minute
U^{lowThr}	Lower threshold in CPU utilization
U^{uprThr}	Upper threshold of CPU utilization
VM^{AL}	Total virtual machines VM^P and VM^S
VM^F	Virtual machines required in future as per $\hat{w}_{t+1}$
VM^{max}	maximum on-demand VMs scaling limit
VM^P	Pending virtual machines list
VM^S	In service virtual machines list
w_t	Requests received at time t
$\hat{w}_{t+1}$	Analyzed future incoming request at time t
$\overline{w}_t$	Request answered at time t

level parameters are an end-user request (w), while infrastructure level parameters are the number of resources and their utilization. The response time is representing the SLA parameter. The monitoring module record the request arrival rate, capacity available and capacity utilized using the control domain.

The response time of each request is calculated as per Eq. 1. In CloudSim, the cloudlet refers to the job submitted to the cloud datacenter. Further, the average response time calculated as per Eq. 2. Where $FinishTime$ is the completion time of cloudlet as per $Clock$, the $ProcessedTime$ represents the time taken for cloudlet processing and $ArrivalTime$ is the submission time of cloudlet as per $Clock$.

$$ResponseTime = FinishTime - ProcessedTime - ArrivalTime \quad (1)$$

$$M_t^{rt} = \frac{\sum_{j=1}^{TotalCloudlets^{Finished}} ResponseTime_j}{TotalCloudlets^{Finished}} \quad (2)$$

The average CPU utilization is calculated as per Eq. 3. VM^{AL} is the sum of in-service virtual machines (VM^S) and, pending virtual machines (VM^P).

$$M_t^u = \frac{\sum_{j=1}^{VM^{AL}} VM_j Utilization}{VM^{AL}} \quad (3)$$

Algorithm 2 The pseudo code of monitoring phase

```
1: /* User behavior parameters */
2: Store w_t                          ▷ Incoming workload (1 minute)
3: /* Infrastructure and platform level parameters */
4: Store VM^P, VM^S                   ▷ VM parameters
5: Store M_t^rt and M_t^u             ▷ SLA parameters and Resource Utilization
```

4.3 Analysis phase

In our previous work, the proactive analyzer designed named the technocrat ARIMA and SVR model (TASM) time series approach with pattern discrimination in the sliding window of incoming requests [63]. In this paper, a new hybrid analysis method is presented with a combination of reactive and proactive analysis techniques. According to Algorithm 4, the proactive section (line no. 4) of analyzer design with the TASM to predict the highest value of incoming workload in a minute for the next scaling interval. The reactive section (line no. 6) is analyzed the CPU utilization and response time.

Algorithm 3 The pseudo code for WorkloadPredictor

```
1: Input: predictionModel, sw (requests per minute from the past △k minutes)
2: Output: ŵ_{t+1}
3: Calculate the Average Rate of Change (ARC) = (y2−y1)/(x2−x1)
4: Calculate the l2-norm |x| = sqrt(x_1^2 + x_2^2 + ... + x_n^2)
5: if ARC < l2 − norm then
6:     Apply ARIMA Model (Non-Seasonal)
7:     Apply Ljung Box test for Independent Residual
8:     if Residuals are Independent and lower RMSE then
9:         Apply the Seasonal Study
10:    else
11:        Apply the Linear Regression (LR) Model
12:    end if
13: else if Apply Teravirta Test to check, is the series linear? then
14:    Apply the LR Model
15:    Perform the Evaluation
16:    Apply the Seasonal Study
17: else Apply the Support Vector Regression (SVR) Model
18:    Perform the Evaluation
19:    Apply the Seasonal Study
20: end if
```

Firstly, the future arrival rate $\hat{w}_{t+1}$ is calculated using the workload prediction as described in Algorithm 3. The sequence of values (w_t) captured during time interval ($\triangle s$) with monitoring phase. The time sequence is represented

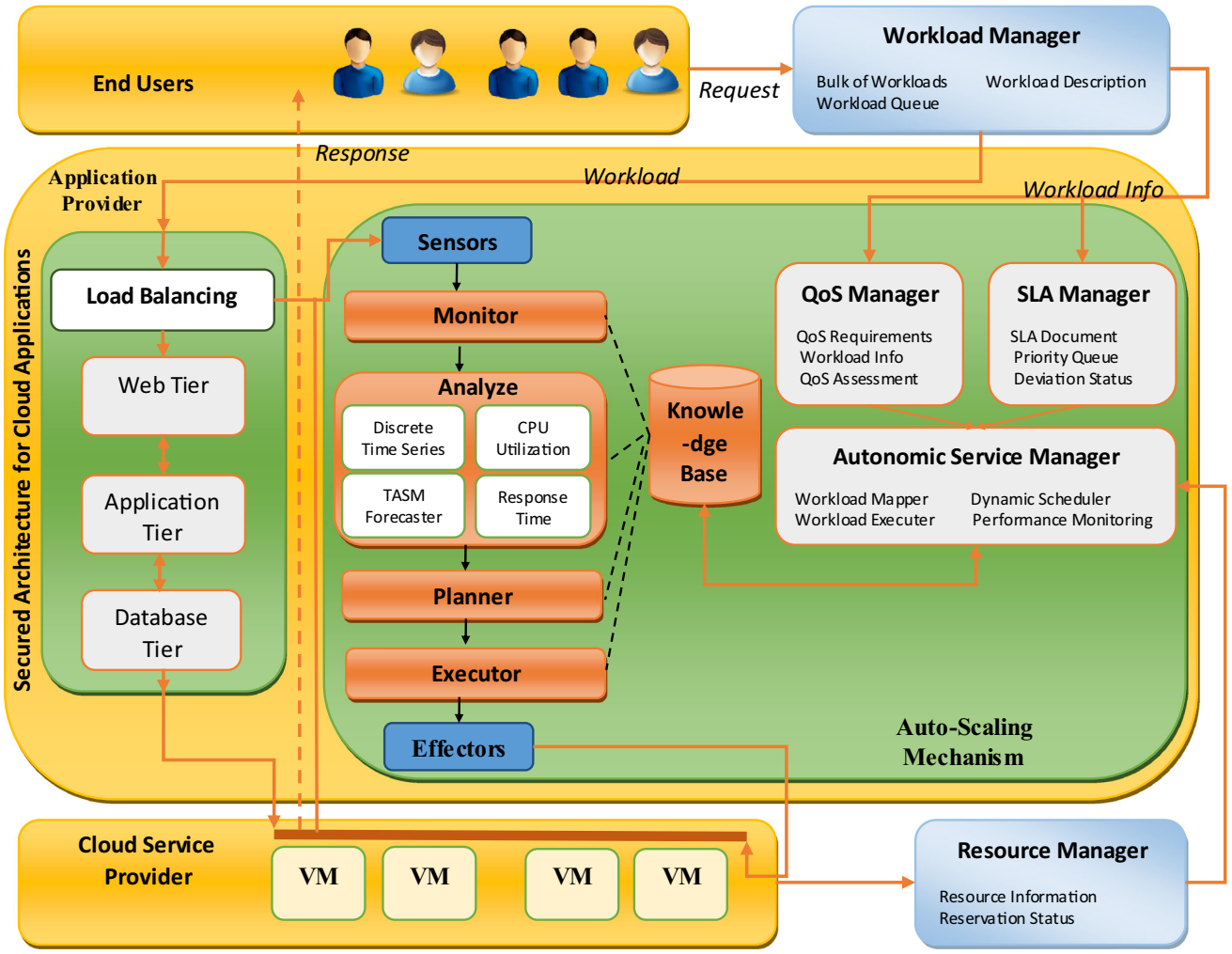


Fig. 3 The cloud architecture for web applications

with $w_s = (w_t^1, w_t^2, ..., w_t^k)$. The $\hat{w}_{i+1}$ is calculated by TASM prediction model (line no. 5), the process shown in Figure 4.

Secondly, the average CPU utilization of VMs calculated for every minute in the scaling interval (line no. 7 to 9). The CPU utilization has been calculated as per Eq. 3. Furthermore, the average CPU utilization during the last scaling interval calculated (line no. 10) as per Eq. 4.

Thirdly, the response time is collected as an SLA parameter during the last minute collected from the monitoring phase. The response time is calculated as per Eq. 1. Afterward, the average response time is calculated using Eq. 2. The average response time is a QoS indicator. In this approach, we have considered the analyzed response time A_t^{rt}, it is the average response time in the last minute (M^{rt}) (line no. 11). This way, we gave the highest priority to the response time experienced at the last minute.

$$A_t^u = \frac{\sum_{t=1}^{k} M_t^u}{k} \tag{4}$$

4.4 Planning

The planning phase makes decisions using reactive and proactive auto-scaling techniques. The reactive approach provided the support for the flash workload. The reliability of predictive analysis is still in doubt because all the time workload doesn't depend upon the historical workload. If the available resources do not meet the sufficient requirement, this approach adds the resources from the pool of resources. This approach ensures that the available capacity should be higher than the required capacity.

According to Algorithm 5, the analyzed CPU utilization (A^u) and analyzed response time (A^{rt}) evaluated with the threshold rules. If the analyzed CPU utilization is greater than the CPU utilization upper threshold value and analyzed

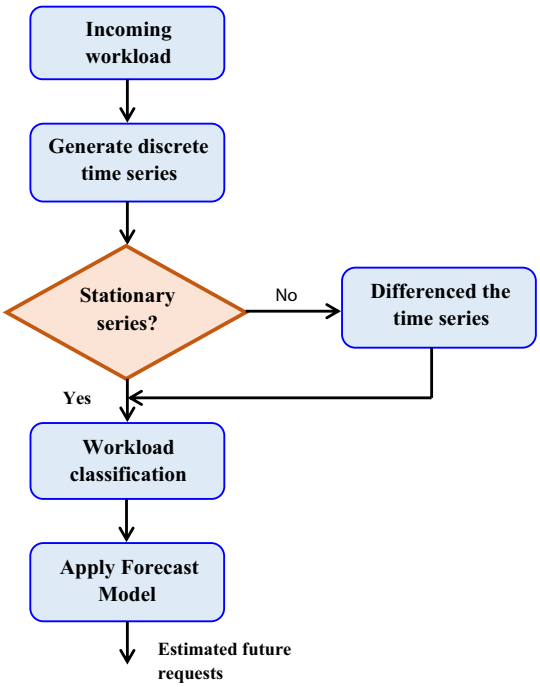


Fig. 4 The workload forecasting approach using TASM prediction model [63]

Algorithm 4 The pseudo code of hybrid analysis approach (Proposed)

1: Input: M^u (duration is the last $\triangle s$ minutes), M^{rt} (duration is the last minute), sw (requests per minute from the past $\triangle k$ minutes)
2: Output: $\hat{w}_{t+1}$, A^u and A^{rt}
3: Variable: double $cpuUtilization \leftarrow 0$, string $predictionModel \leftarrow TASM$
4: /* Proactive Section */
5: $\hat{w}_{t+1} \leftarrow$ WorkloadPredictor($predictionModel, sw$) ▷ Predict the future arrival rate 3
6: /* Reactive Section */
7: **for** $i = 0 \;\; to \;\; i < \triangle s$ **do**
8: $\quad cpuUtilization \leftarrow cpuUtilization + M^u_{t-i-\triangle s}$
9: **end for**
10: $A^u \leftarrow cpuUtilization/\triangle s$ ▷ Average CPU utilization in $\triangle s$ minutes
11: $A^{rt} \leftarrow M^{rt}$ ▷ Response time of last minute
12: return $\hat{w}_{t+1}$, A^u and A^{rt}

response time is greater than the upper threshold value of response time then (line no. 4), an immediate $ScaleUp$ decision takes place (line no. 5). No further consideration will take place and return (line no. 6). If the analyzed CPU utilization is lower than the CPU utilization lower threshold value and analyzed response time is less than the lower threshold value of response time then (line no. 7), the $ScaleDown$ decision set on a temporary basis (line no. 8). This decision further filtered through the proactive section (line no. 10) then the final decision will take place.

$$VM^F = \left\lceil \frac{\hat{w}_{t+1}}{RT^{SLA}\mu} \right\rceil \tag{5}$$

The prediction section (line no. 10) applied the queuing model to calculate the number of VMs to serve the incoming workload is the next scaling interval (line no. 11). It takes three arguments future incoming request, processing rate (μ) and RT^{SLA} is response time as per SLA. If the required capacity is less than the future capacity than the $ScaleUp$ decision takes place (line no. 13). Afterward, if the desired capacity is the same as current capacity than the decision must be $DoNothing$ (line no. 15), otherwise, we continue with the decision of reactive auto-scaling that was $ScaleDown$.

The predictive auto-scaling used for the short-term predictable workload and the reactive auto-scaling is applied for less predictable fluctuations in the incoming workload. The state under-provisioning and over-provisioning of the resources is known as resource oscillation in cloud computing. The predictive approach ready the VMs before incoming workload and reactive technique overcome the prediction error by taking immediate scaling decisions in case of resource oscillation in the cloud environment.

Algorithm 5 The pseudo code of hybrid planning approach (Proposed)

1: Input: A^u, A^{rt}, U^{lowThr}, U^{upThr}, RT^{lowThr}, RT^{upThr}, $\hat{w}_{t+1}$
2: Output D
3: /* Reactive Scaling */
4: **if** $A^u > U^{upThr}$ and $A^{rt} > RT^{upThr}$ **then**
5: $\quad D \leftarrow ScaleUp$
6: $\quad$ return D
7: **else if** $A^u < U^{lowThr}$ and $A^{rt} < RT^{lowThr}$ **then**
8: $\quad D \leftarrow ScaleDown$
9: **end if**
10: /* Proactive Scaling */
11: $VM^F \leftarrow QueuingModel(\hat{w}_{t+1}, \mu, RT^{SLA})$ ▷ Queuing model $M/M/m$ estimates the future capacity
12: **if** $VM^{AL} < VM^F$ **then**
13: $\quad D \leftarrow ScaleUp$
14: **else if** $VM^{AL} == VM^F$ **then**
15: $\quad D \leftarrow DoNothing$
16: **end if**
17: return D

4.5 Execution

The execution phase action depends upon the interpretation of the planner. This phase takes the final decision for the scaling up, down or do nothing, and put a request to CP. The default executor selects the machines randomly for scale-up/down from the resource pool. This phase cross-validate the on-demand virtual machines limit before the scale-up decision, if the limit-exceeding, it will deny the scale-up deci-

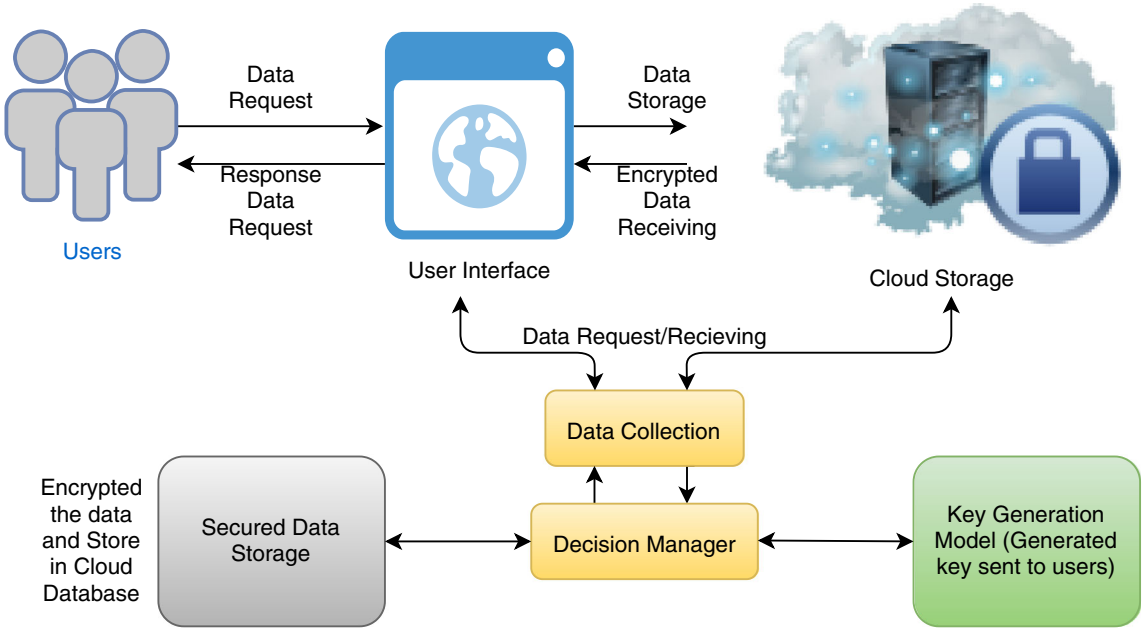


Fig. 5 Secured architecture for cloud applications [25]

sion. Similarly, if the on-demand limit reaches zero no further request will be entertained.

4.6 Security of cloud applications

The features in cloud architecture provides various facilities for a web application to acquire the resources in demand and better cost for operations. However, these features arise numerous concerns about security. As the cloud is a multi-domain environment, so the users can have different trust and security requirements. The cloud provides this facility through Service Oriented Architecture (SOA) to compose the services as per the requirement. In this section, we used the Chinese Remainder Theorem (CRT) [66] based privacy-preserving and secured storage for web application as shown in Fig. 5.

Ganapathy et al. [25] proposed a new privacy-preserving and secured storage model. This model performance is best among the state-of-the-art algorithms. In this work, we have used that algorithm to secure the storage and communication of web applications in cloud data centers. There are six components in this architecture includes key generation model, secured data storage model, decision manager, data collection module, cloud database, and user interface as shown in Fig. 5. The user interface module serves as a primary component of data request/response and storage. The data of the user either store as simple or ciphertext. The data collected from the decision manager and user interface is performed by the data collection module. The system architecture components are controlled by the decision manager. Furthermore, the encryption of data accomplished at the data storage module. The collection module encrypts or decrypts data to store in a cloud database. Although, the user can request through the data collection module forwarded for key generation. The decision manager is responsible for the distribution of keys to the users. The specific user is then able to access the required data from the cloud database. The detailed working can be referred from the article [25].

5 Performance evaluation

The goal of the present study is to develop the robust auto-scaling approach for application provider by hybrid auto-scaling approach with essential security features. The experiment has been performed in summarized as follows:

- The prediction models linear regression (LR), support vector regression (SVR), autoregression (AR), moving average (MA), autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), and TASM implemented in *R* tool and comparison made for 1 min short term prediction for real workload traces.
- The existing and proposed techniques implemented in Cloudsim and observation made for auto-scaling decisions.
- The performance metrics: response time and CPU utilization are observed during the experiment.

Table 3 Summary of datasets information

Dataset name	ClarkNet	NASA
File name	Aug28log, access1	accesslogAug95
File size	171.0 MB, 172.5 MB	167.8 MB
Length (s)	3, 328, 587(14days)	3, 461, 612 (28 days)
Timestamp resolution	1 second	1 second
Sampling interval	1 min	1 min
Number of samples	2880 (2 days)	2880 (2 days)

– The comparative analysis performed on VM allocation, SLA violation, renting cost of proposed and existing methods to validate the robustness and efficiency.

5.1 Experimental setup

The comparison of the proposed model is performed on the CloudSim toolkit. The prediction model is implemented in the R tool. R caller library is used to integrate R API in the CloudSim toolkit. The various existing auto-scaling approach mentioned as RightScale threshold rules-based technique [1], AR [41], SVR [54], ARIMA [7], TASM [63] compared with proposed approach in terms of response time, number of VMs allocated and resource utilization. The RHAS provides maximum resource utilization percentage and minimum response time.

The workload used in the experiments are mentioned in Table 3. The experiment is conducted with the cloud provider, application provider and end-user entities.

5.1.1 Cloud provider

CloudSim provides the ability for the cloud provider. The classes are added for resource rental and cost management. Time-shared scheduling is used for the experiment purpose.

5.1.2 Application provider

Application providers host the application in cloud infrastructure. The on-demand resources have a significant delay in the startup of virtual machines [33]. The interesting study is conducted and show that VM startup delay varies depending on the factors such as VM request time, VM size and day of the week. However, in simulation, some considered as the normal distribution or fixed number. In our experiment, we considered the VM startup delay as fixed for 5 min. Accordingly, hybrid auto-scaler took scaling decisions after every 10 min with the highest priority to reactive scale-up.

5.1.3 End user

The emulation of ClarkNet and NASA workload incorporate to AP for the web application requests. These workloads are used by various auto-scaling performance evaluations [4,42, 45,50]. The input workload is mentioned in Table 3.

6 Results and discussion

The performance evaluation is performed against 6 metrics as follows:

6.1 Prediction accuracy of time series models

The cloud infrastructure parameters such as CPU utilization, response time and request arrival rate are collected. The prediction of future requests performed using the TASM method on the basis of classification model [63]. The sliding window is used to capture the latest request in the history and on the basis of the classification approach, prediction models are applied. Linear and non-linear models are applied to capture the various types of trends in the incoming workload. The proposed model shows the less difference in incoming workload and predicted workload, thus feasible to implement in a cloud environment for web applications. In the previous work, we tested the prediction accuracy of 10 minutes discrete series. In this experiment, we have tested the discrete series of 1 min shown in Fig. 6 for the ClarkNet series and Fig. 7 for the NASA series. So, that highest in predicted workload in 1 min from the upcoming scaling interval choose, which can decrease the SLA violation and proper resource utilization.

The accuracy of the prediction model is calculated using Root-Mean-Square-Error (RMSE) and Mean-Absolute-Percentage-Error (MAPE). The standard metric for RMSE and MAPE as specified in Eqs. 6 and 7 respectively mention in Table 4 for ClarkNet series and Table 5 for NASA series. It has been observed that the prediction model gives better accuracy with long-term workload forecasting. During the experiment, when we performed the short-term prediction (e.g. 1 min), most of the prediction models fail to give ade-

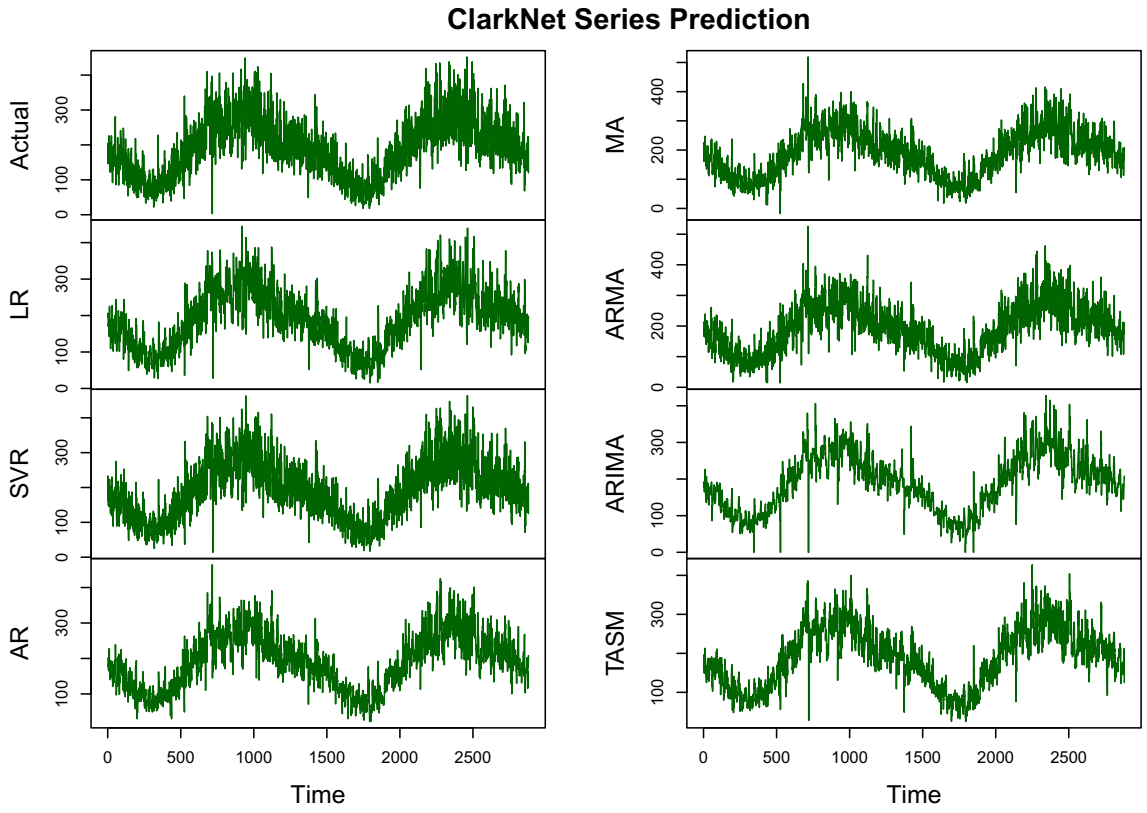


Fig. 6 ClarkNet workload prediction using LR, SVR, AR, MA, ARMA, ARIMA and TASM

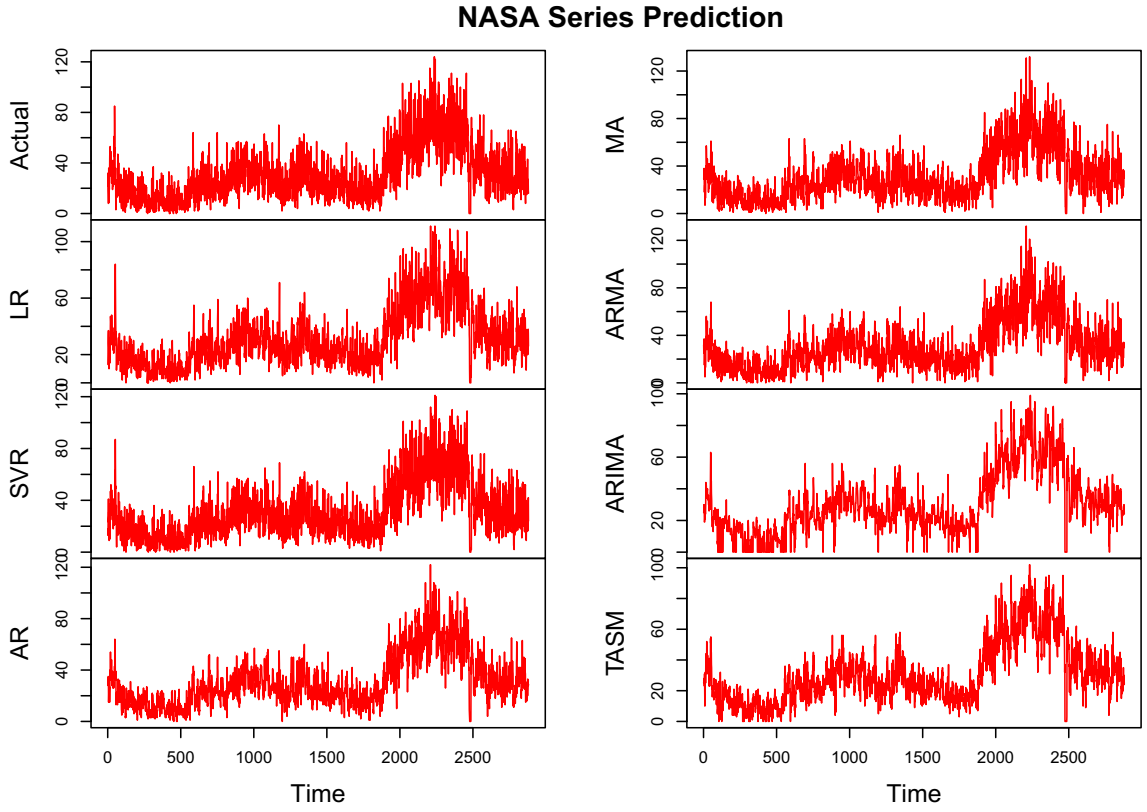


Fig. 7 NASA workload prediction using LR, SVR, AR, MA, ARMA, ARIMA and TASM

Table 4 Accuracy of prediction models for ClarkNet workload

Prediction model	RMSE	MAPE
LR	64.37	34.52
SVR	64.67	32.95
AR	54.60	30.42
MA	59.24	33.14
ARMA	59.25	32.98
ARIMA	51.24	26.64
TASM	42.24	20.63

Table 5 Accuracy of prediction models for NASA workload

Prediction model	RMSE	MAPE
LR	16.56	66.77
SVR	17.0	67.93
AR	14.09	56.32
MA	15.72	62.40
ARMA	15.23	60.37
ARIMA	13.37	52.68
TASM	11.01	39.21

quate accuracy. Thus, the need for hybrid auto-scaling arises, which can further evaporate the prediction errors using the reactive technique.

$$RMSE = \sqrt{\frac{1}{n}\sum_{s=1}^{n}\left(actualWorkload_s - predictedWorkload_s\right)^2} \tag{6}$$

$$MAPE = \frac{1}{n}\sum_{s=1}^{n}\left|\frac{actualWorkload_s - predictedWorkload_s}{actualWorkload_s}\right| \times 100\% \tag{7}$$

6.2 VM allocation

The difference between the number of resources estimated and consumed are calculated to analyze the performance of proposed hybrid analysis and planning phase. The experiment is performed on the first two days discrete model of the ClarkNet and NASA time series. The analysis phase performance is shown in the accuracy of the prediction model and found the TASM gives better performance for web applications workload prediction as compared to other prediction models. The planning phase experiment has been conducted on two datasets: ClarkNet and NASA. The first experiment is conducted with the proactive auto-scaling using TASM with ClarkNet shown in Fig. 8 and proposed model result shown in Fig. 9. The second experiment is conducted on NASA workload with proactive TASM and proposed auto-scaling approach shown in Figs. 10 and 11. The TASM prediction model is able to predict future demand in peak hours and normal hours. Still, resource oscillation is there due to prediction error. The AP renting VMs as per demand and release with mitigation of workload. The calculation performed in minutes from demand to release of VMs. The elevated length indicates provisioning stability. The stabilization is observed in the robust planning approach.

The third and fourth experiments are conducted on the 6th and 7th day of ClarkNet and NASA time series for scaling overhead. The experiment results are shown in Figs. 12 and 13. The proposed planning algorithm further reduces the over-utilization and under-utilization up to 16%.

6.3 CPU utilization

The CPU utilization is referred to as the work handled by the CPU of VMs deployed for the web application in a cloud environment. The performance of RHAS has been tested with CPU utilization metrics. This is the percentage of current CPU usage during the processing of user's requests. The CPU utilization is tested against the threshold-based and proactive technique. The experiment results shown in Fig. 14 for ClarkNet series and Fig. 15 for NASA series. The threshold (TR) based technique is able to achieve higher CPU utilization in the ClarkNet workload model, but due to the scaling interval of 10 min the performance of the TR technique is just 42%. The proposed technique is the mixed approach of reactive and proactive technique, thus able to give a consistent performance of 90% CPU utilization.

6.4 Response time

Response time is the elapsed time between the request submitted to the request-response. The response time of the web application must be quick. In this experiment, the desired response time has agreed to 1 s as per SLA. The minimum response time is a parameter of quality of experience (QoE) of each user request. Figures 16 and 17 are the result of average response time for ClarkNet and NASA workload respectively. The proposed method has a relatively low response time and showing the RHAS is adapting the dynamic workload.

6.4.1 SLA violation

The SLA agreement is an important aspect which ensures the maximum availability of cloud services to the end-users. The CP has to pay a penalty in case of an SLA violation. The SLA violation is calculated as per equation Eq. 8.

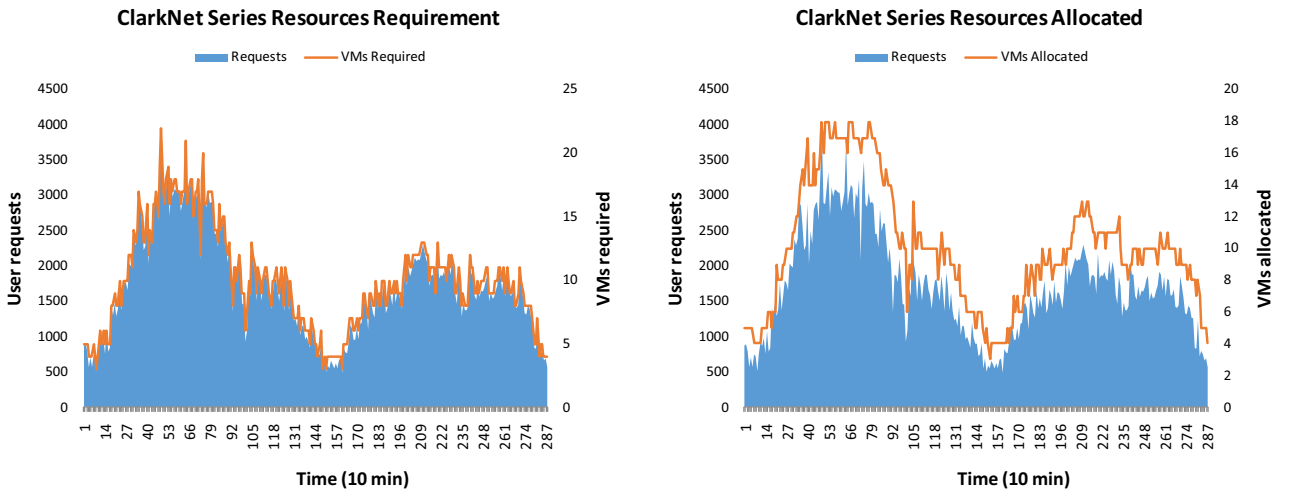


Fig. 8 ClarkNet series VM required and allocated using proactive scaling

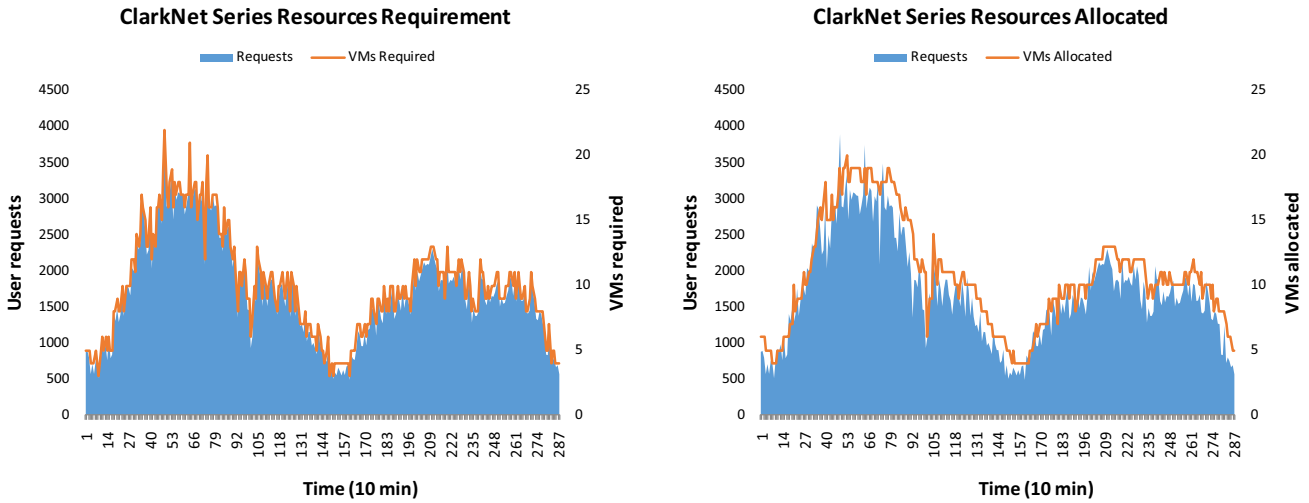


Fig. 9 ClarkNet series VM required and allocated using proposed RHAS

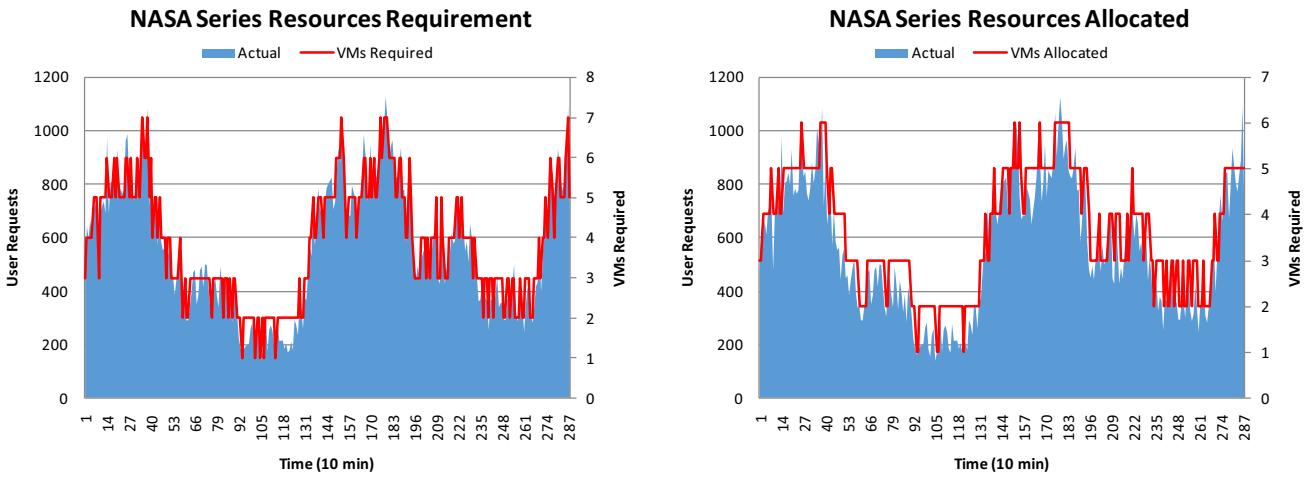


Fig. 10 NASA Series VM required and allocated using proactive scaling

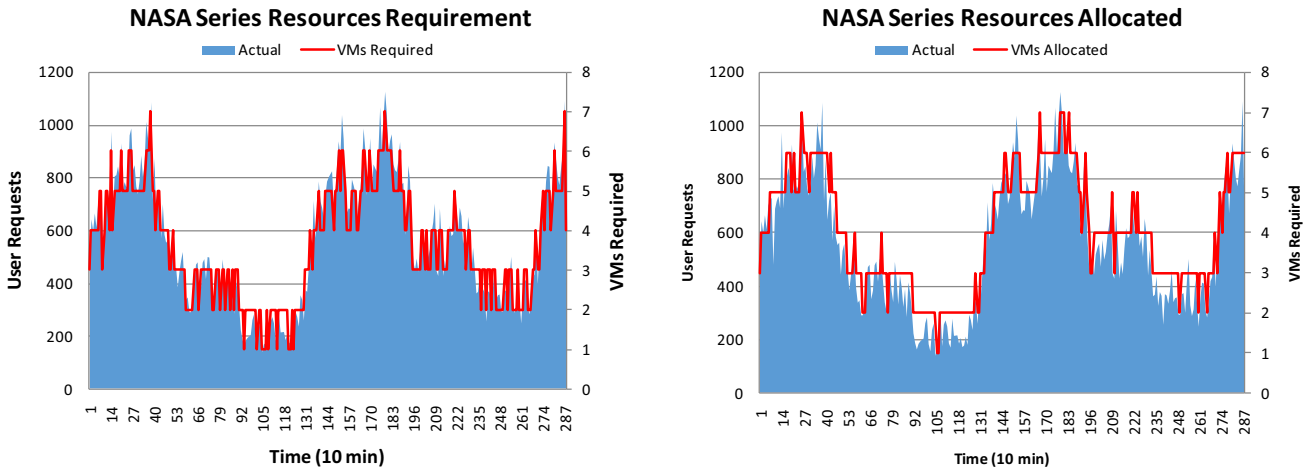


Fig. 11 NASA series VM required and allocated using proposed RHAS

Fig. 12 ClarkNet series scaling overhead

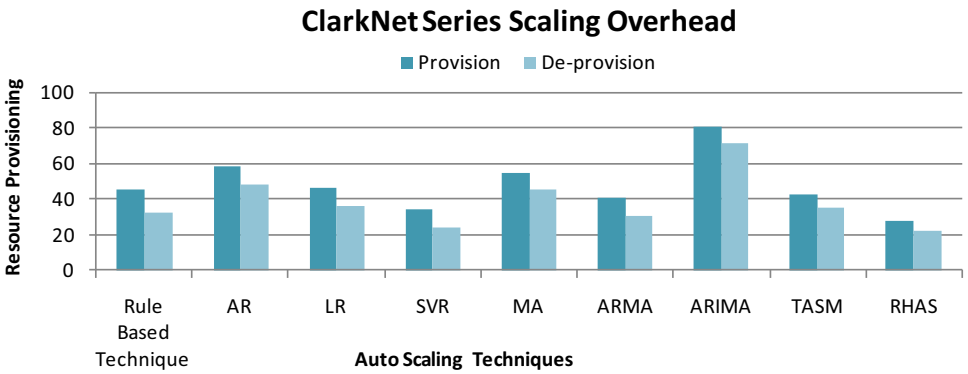


Fig. 13 NASA series scaling overhead

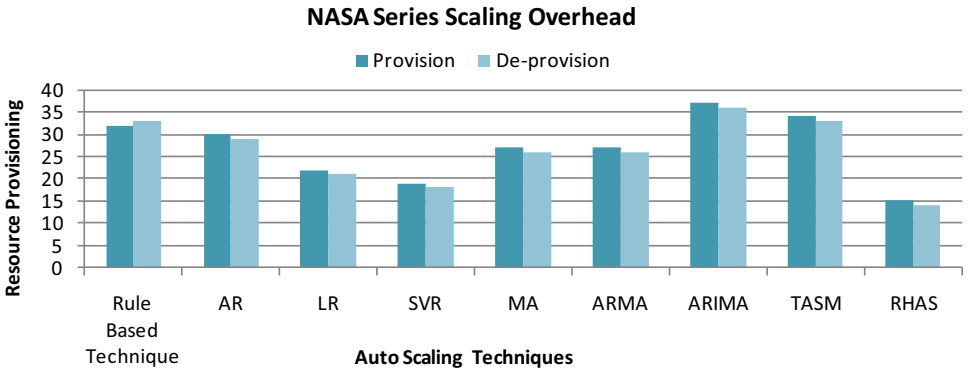
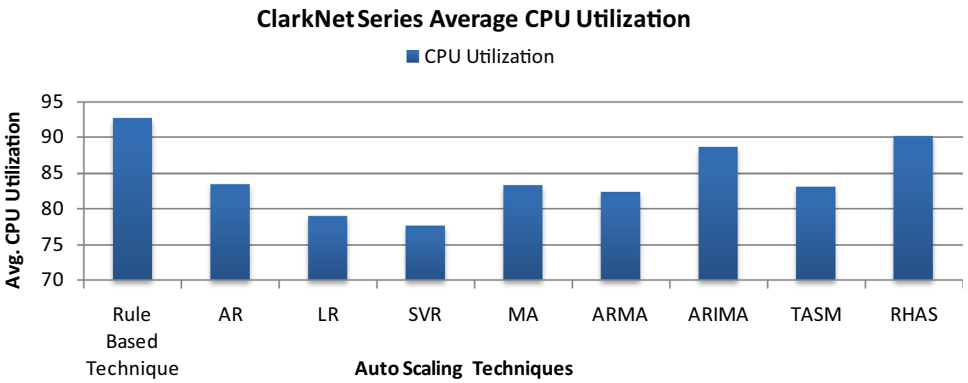


Fig. 14 ClarkNet series average CPU utilization



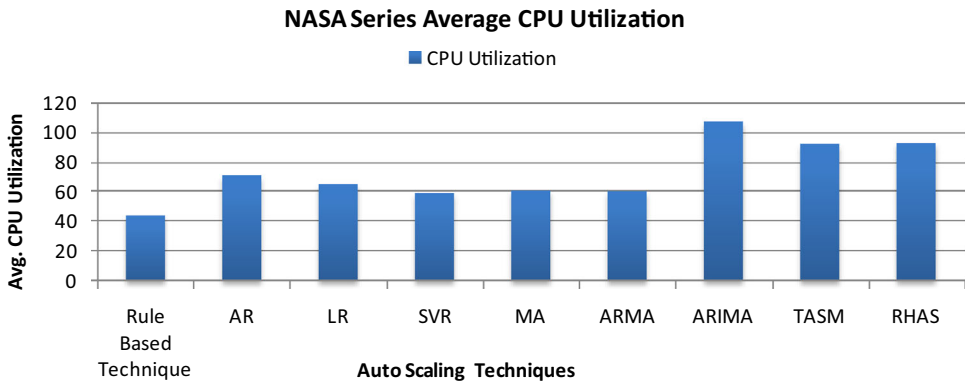


Fig. 15 NASA series average CPU utilization

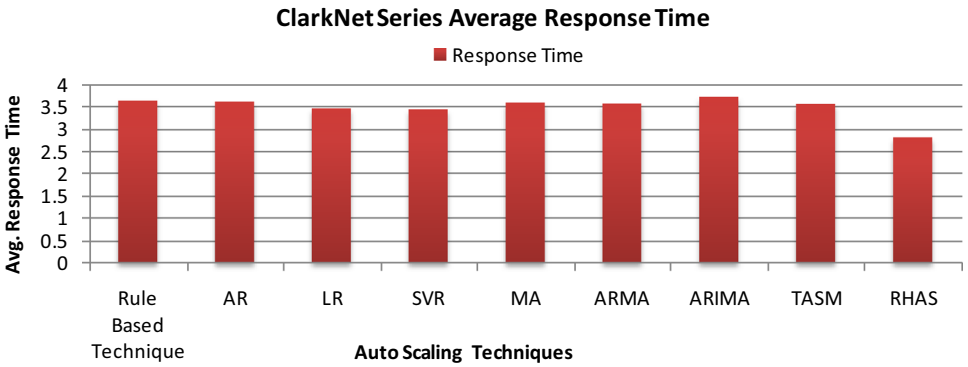


Fig. 16 ClarkNet series average response time

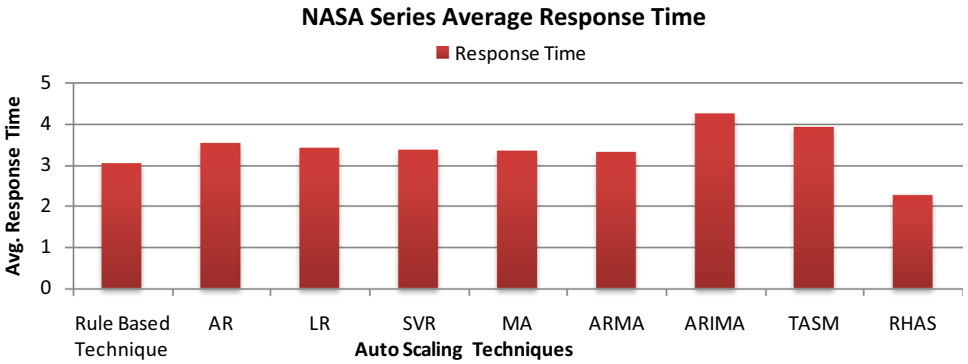


Fig. 17 NASA series average response time

$$SLAV = \sum_{i=1}^{w} RT_i - S^{rt} \tag{8}$$

The SLA violation is of ClarkNet and NASA series experiment results shown in Figs. 18 and 19. The experiment results show that the proposed technique is experienced a 1% SLA violation. The end-user will experience the QoS for their application accessibility.

6.4.2 Renting cost

The on-demand resources are rented as pay per use principal. The cost of renting is calculated hourly basis as per Amazon EC2. It is the sum of all the resources after ceiling each VM utilization hours. The different rates of penalty are defined for different QoS requirement [28]. The total cost is calculated as per Eq. 9.

$$Cost = Rentingcost + SLApenaltycost \tag{9}$$

$$Rentingcost = ExecutionTime_i \times Price \tag{10}$$

$$Delaytime = Estimatedfinishtime - Actualfinishtime \tag{11}$$

$$PenaltyCost = \sum_{i=1}^{C} (Penalty_{minimm} + PenaltyRate \times |DelayTime|)_i \tag{12}$$

The experiment results are shown in Fig. 20 for ClarkNet series renting cost and 21 for the NASA series renting cost. The SLA violation leads to the penalty to the service provider.

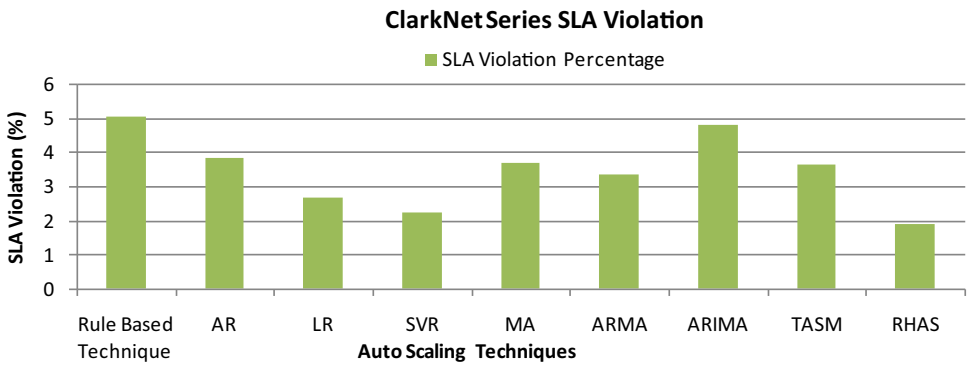


Fig. 18 ClarkNet series SLA violation

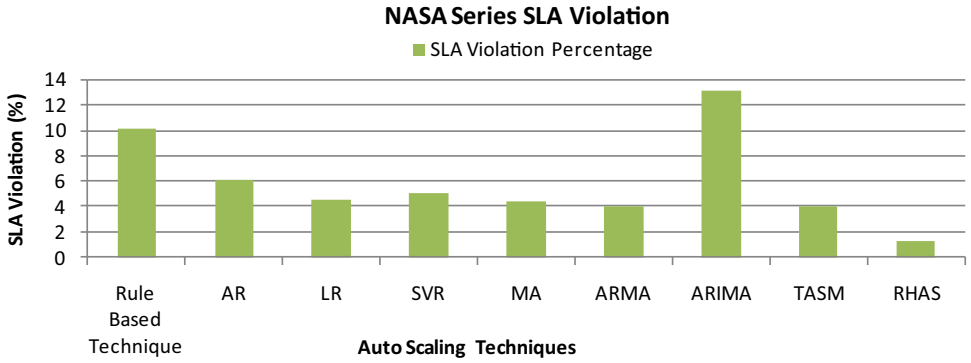


Fig. 19 NASA series SLA violation

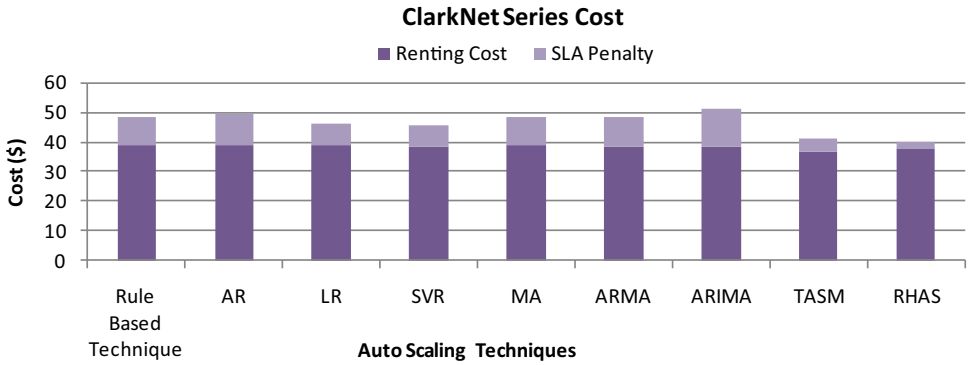


Fig. 20 ClarkNet series overall cost

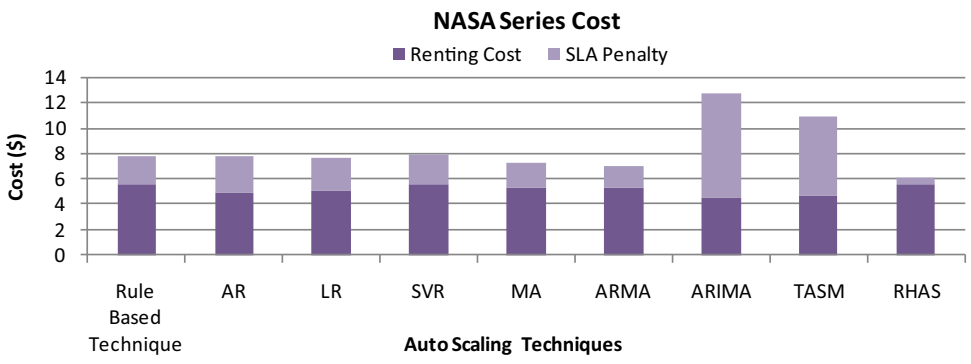


Fig. 21 NASA series overall cost

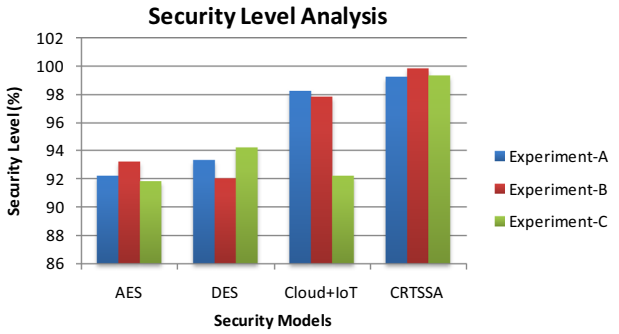


Fig. 22 Security level analysis

The proposed model is cost-efficient for the AP in terms of renting cost and minimum SLA penalty.

6.5 Security analysis

The security level analysis is conducted with AES, DES, cloud with IoT model [16] and CRT-based Secure Storage Algorithm (CRTSSA) [25]. The size of the file varies to perform the different experiments on these algorithms. Figure 22 shows the security level analysis of various algorithms. It has been observed that the CRT key generation and CRTSSA gives an outstanding performance. The encryption process designed such a way that it makes our proposed RHAS model robust by raising the security level of web applications in cloud data centers.

It is important to mention that the proposed RHAS auto-scaling technique is a robust approach, which gave an efficient and fair amount of QoS to the end-users, and equally provides cost benefits to the AP. It minimizes the renting cost with reduction in SLA penalty. The end-users get QoE with justified response time as per SLA.

7 Conclusions and summary

In the cloud environment, the web application providers face the issue of irregular load fluctuation, this further leads to the uncertain scaling decision. In this paper, we designed the robust auto-scaling technique (RHAS) with hybrid analysis and planning approaches for web applications in cloud infrastructure. The proposed technique provides benefits to the auto-scaling with cost and QoS parameters. The simulation result shows the benefits in renting cost and reduction in SLA violation. It also gives a fair amount of CPU utilization. As a result, it has been clear that apart from response time, other parameters such as the number of requests and CPU utilization are also equally important in scaling decisions. The CRT-based secure storage provides better security to the user's request and response for web applications. The experiment results demonstrates that the proposed RHAS auto-scaling approach reduce the cost upto 14%, response time upto 18%, and SLA violation upto 25%. Furthermore, it also achieves the rise in security level from 2% to 4%. The robust hybrid scaling approach can provide benefits to application providers in terms of cost and equally gives QoE to the end-user.

7.1 Future directions and open challenges

In the future, we shall explore the applicability of the present model/technique and any potentially needed extensions in the following main directions.

1. **Dynamic scalability** Further, the dynamic scalability can be incorporated [14], which can provide operational capabilities to improve performance of cloud computing applications in a cost-effective way, yet to be fully exploited [33].In addition, the vertical scaling strategy can be designed to overcome the challenge of startup delay of VMs [70].
2. **Mobile cloud computing** In future, implement dynamic/real-time offloading techniques for energy conservation in hybrid Fog-Cloud setups using RHAS. Further, there is a need to perform offloading using Mobile Cloud Computing for mobility and energy-aware based offloading and task scheduling decision.
3. **Edge and fog computing** We can extend our model from cloud computing to another emerging computing models such as fog and edge to reduce the latency and response time of cloud applications dynamically [29].
4. **Internet of Things** Extending auto-scaling across the compute continuum from IoT devices to cloud in order for timely execution/processing of scaling decisions is a challenging problem that needs to be addressed [29].
5. **Blockchain** Auto-scaling utilizing Blockchain capabilities such as Smart SCs in order for addressing first security concerns and second unsolved monetization problem in federated cloud for services provided in collaborative computing is a new challenge [29].
6. **Adaptive prediction model** In future work, the AI-based adaptive prediction model can be designed to predict the number of the requests as per the desired response time of SLA within the scaling interval [29].
7. **Artificial intelligence (AI)** Self-correction prediction and multi-objective auto-scaling using AI for the trade-off between performance and cost can be accomplished using Deep Learning. Recently, deep learning, i.e. automation of predictive analytics-a subset of AI-has gain more attention for solving the problems which have not been yet solved [29,64].

Acknowledgements We would like to thank the editor, area editor and anonymous reviewers for their valuable comments and suggestions to help and improve our research paper.

References

1. Adler, B.: Building Scalable Applications in the Cloud: Reference Architecture & Best Practices. Rightscale inc, Santa Barbara (2011)
2. Amiri, M., Mohammad-Khanli, L.: Survey on prediction models of applications for resources provisioning in cloud. J. Netw. Comput. Appl. **82**, 93–113 (2017)
3. Aslanpour, M.S., Dashti, S.E.: Proactive auto-scaling algorithm (pasa) for cloud application. Int. J. Grid High Perform. Comput. **9**(3), 1–16 (2017)
4. Aslanpour, M.S., Ghobaei-Arani, M., Toosi, A.N.: Auto-scaling web applications in clouds: a cost-aware approach. J. Netw. Comput. Appl. **95**, 26–41 (2017)
5. Bodik, P., Griffith, R., Sutton, C., Fox, A., Jordan, M.I., Patterson, D.A.: Automatic exploration of datacenter performance regimes. In: Proceedings of the 1st workshop on Automated control for datacenters and clouds, pp. 1–6. ACM (2009)
6. Bodík, P., Griffith, R., Sutton, C., Fox, A., Jordan, M.I., Patterson, D.A.: Statistical machine learning makes automatic control practical for internet datacenters. HotCloud **9**, 12 (2009)
7. Calheiros, R.N., Masoumi, E., Ranjan, R., Buyya, R.: Workload prediction using arima model and its impact on cloud applications' qos. IEEE Trans. Cloud Comput. **3**(4), 449–458 (2015)
8. Caron, E., Desprez, F., Muresan, A.: Pattern matching based forecast of non-periodic repetitive behavior for cloud clients. J. Grid Comput. **9**(1), 49–64 (2011)
9. Casalicchio, E.: A study on performance measures for auto-scaling cpu-intensive containerized applications. Clust. Comput. **22**(3), 995–1006 (2019)
10. Casalicchio, E., Lundberg, L., Shirinbab, S.: Energy-aware auto-scaling algorithms for cassandra virtual data centers. Cluster Computing **20**(3), 2065–2082 (2017)
11. Chandra, A., Gong, W., Shenoy, P.: Dynamic resource allocation for shared data centers using online measurements. In: International Workshop on Quality of Service, pp. 381–398. Springer (2003)
12. Chen, G., He, W., Liu, J., Nath, S., Rigas, L., Xiao, L., Zhao, F.: Energy-aware server provisioning and load dispatching for connection-intensive internet services. NSDI **8**, 337–350 (2008)
13. Computing, A., et al.: An architectural blueprint for autonomic computing. IBM White Paper **31**, 1–6 (2006)
14. Coulson, N.C., Sotiriadis, S., Bessis, N.: Adaptive microservice scaling for elastic applications. IEEE Internet Things J. **7**(5), 4195–4202 (2020)
15. Coutinho, E.F., de Carvalho Sousa, F.R., Rego, P.A.L., Gomes, D.G., de Souza, J.N.: Elasticity in cloud computing: a survey. Ann. Telecommun. **70**(7–8), 289–309 (2015)
16. Cui, H., Yi, X., Nepal, S.: Achieving scalable access control over encrypted data for edge computing networks. IEEE Access **6**, 30049–30059 (2018)
17. De Grande, R.E., Boukerche, A., Alkharboush, R.: Time series-oriented load prediction model and migration policies for distributed simulation systems. IEEE Trans. Parallel Distrib. Syst. **28**(1), 215–229 (2017)
18. Dutta, S., Gera, S., Verma, A., Viswanathan, B.: Smartscale: Automatic application scaling in enterprise clouds. In: Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, pp. 221–228. IEEE (2012)
19. EC2", A.: "spot instances" (2018). https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html
20. Elumalaivasan, P., Kulothungan, K., Sannasi, G., Arputharaj, K.: Trust based ciphertext policy attribute based encryption techniques for decentralized disruption tolerant networks. Aust. J. Basic Appl. Sci **10**, 18–26 (2016)
21. Erradi, A., Iqbal, W., Mahmood, A., Bouguettaya, A.: Web application resource requirements estimation based on the workload latent features. IEEE Trans. Serv. Comput. (2019)
22. Fallah, M., Arani, M.G., Maeen, M.: Nasla: novel auto scaling approach based on learning automata for web application in cloud computing environment. Int. J. Comput. Appl. **113**(2), 18–23 (2015)
23. Fang, W., Lu, Z., Wu, J., Cao, Z.: Rpps: a novel resource prediction and provisioning scheme in cloud data center. In: Services Computing (SCC), 2012 IEEE Ninth International Conference on, pp. 609–616. IEEE (2012)
24. Fernandez, H., Pierre, G., Kielmann, T.: Autoscaling web applications in heterogeneous cloud infrastructures. In: Cloud Engineering (IC2E), 2014 IEEE International Conference on, pp. 195–204. IEEE (2014)
25. Ganapathy, S., et al.: A secured storage and privacy-preserving model using crt for providing security on cloud and iot-based applications. Comput. Netw. **151**, 181–190 (2019)
26. Garí, Y., Monge, D.A., Mateos, C., Garino, C.G.: Learning budget assignment policies for autoscaling scientific workflows in the cloud. Clust. Comput. **23**(1), 87–105 (2020)
27. Ghanbari, H., Simmons, B., Litoiu, M., Iszlai, G.: Exploring alternative approaches to implement an elasticity policy. In: Cloud Computing (CLOUD), 2011 IEEE International Conference on, pp. 716–723. IEEE (2011)
28. Gill, S.S., Chana, I., Singh, M., Buyya, R.: Chopper: an intelligent qos-aware autonomic resource management approach for cloud computing. Clust. Comput. **21**(2), 1203–1241 (2018)
29. Gill, S.S., Tuli, S., Xu, M., Singh, I., Singh, K.V., Lindsay, D., Tuli, S., Smirnova, D., Singh, M., Jain, U., et al.: Transformative effects of IoT, blockchain and artificial intelligence on cloud computing: evolution, vision, trends and open challenges. Internet of Things **8**, 100118 (2019)
30. Gong, Z., Gu, X., Wilkes, J.: Press: predictive elastic resource scaling for cloud systems. In: 2010 International Conference on Network and Service Management, pp. 9–16. IEEE (2010)
31. Han, R., Ghanem, M.M., Guo, L., Guo, Y., Osmond, M.: Enabling cost-aware and adaptive elasticity of multi-tier cloud applications. Fut. Gener. Comput. Syst. **32**, 82–98 (2014)
32. Hashmi, K., Malik, Z., Erradi, A., Rezgui, A.: Qos dependency modeling for composite systems. IEEE Transactions on Services Computing **11**(6), 936–947 (2016)
33. Hu, S., Smith, J.E.: Reducing startup time in co-designed virtual machines. In: 33rd International Symposium on Computer Architecture (ISCA'06), pp. 277–288. IEEE (2006)
34. Huang, J., Li, C., Yu, J.: Resource prediction based on double exponential smoothing in cloud computing. In: Consumer Electronics, Communications and Networks (CECNet), 2012 2nd International Conference on, pp. 2056–2060. IEEE (2012)
35. Iqbal, W., Dailey, M.N., Carrera, D.: Low cost quality aware multi-tier application hosting on the amazon cloud. In: 2014 International Conference on Future Internet of Things and Cloud, pp. 202–209. IEEE (2014)
36. Iqbal, W., Dailey, M.N., Carrera, D.: Unsupervised learning of dynamic resource provisioning policies for cloud-hosted multitier web applications. IEEE Syst. J. **10**(4), 1435–1446 (2015)
37. Iqbal, W., Dailey, M.N., Carrera, D., Janecek, P.: Adaptive resource provisioning for read intensive multi-tier applications in the cloud. Fut. Gener. Comput. Syst. **27**(6), 871–879 (2011)
38. Islam, S., Keung, J., Lee, K., Liu, A.: Empirical prediction models for adaptive resource provisioning in the cloud. Fut. Gener. Comput. Syst. **28**(1), 155–162 (2012)

39. Kavin, B.P., Ganapathy, S., Karman, A.: An intelligent task scheduling approach for cloud using ipso and a* search algorithm. In: 2018 Eleventh International Conference on Contemporary Computing (IC3), pp. 1–5. IEEE (2018)
40. Kim, H., el Khamra, Y., Jha, S., Parashar, M.: An autonomic approach to integrated hpc grid and cloud usage. In: e-Science, 2009. e-Science'09. Fifth IEEE International Conference on, pp. 366–373. IEEE (2009)
41. Kupferman, J.: Scaling into the cloud. CS270 Advanced Operating Systems, 2009 (2009)
42. Li, J., Su, S., Cheng, X., Song, M., Ma, L., Wang, J.: Cost-efficient coordinated scheduling for leasing cloud resources on hybrid workloads. Parallel Comput. **44**, 1–17 (2015)
43. Lim, H.C., Babu, S., Chase, J.S.: Automated control for elastic storage. In: Proceedings of the 7th international conference on Autonomic computing, pp. 1–10. ACM (2010)
44. Lin, W.: Study on the design and application of the user information resources of track and field web course based on software programming method. Clust. Comput. **22**(6), 15295–15303 (2019)
45. Liu, J., Zhang, Y., Zhou, Y., Zhang, D., Liu, H.: Aggressive resource provisioning for ensuring qos in virtualized environments. IEEE Trans. Cloud Comput. **1**, 1–1 (2015)
46. Lorido-Botran, T., Miguel-Alonso, J., Lozano, J.A.: A review of auto-scaling techniques for elastic applications in cloud environments. J. Grid Comput. **12**(4), 559–592 (2014)
47. Mao, M., Humphrey, M.: 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, pp. 1–12. IEEE (2011)
48. Mao, M., Humphrey, M.: A performance study on the vm startup time in the cloud. In: 2012 IEEE 5th international conference on Cloud Computing (CLOUD), pp. 423–430. IEEE (2012)
49. Maurer, M., Breskovic, I., Emeakaroha, V.C., Brandic, I.: Revealing the mape loop for the autonomic management of cloud infrastructures. In: Computers and Communications (ISCC), 2011 IEEE Symposium on, pp. 147–152. IEEE (2011)
50. Messias, V.R., Estrella, J.C., Ehlers, R., Santana, M.J., Santana, R.C., Reiff-Marganiec, S.: Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure. Neural Comput. Appl. **27**(8), 2383–2406 (2016)
51. Mi, H., Wang, H., Yin, G., Zhou, Y., Shi, D., Yuan, L.: Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers. In: 2010 IEEE International Conference on Services Computing (SCC), pp. 514–521. IEEE (2010)
52. Muthurajkumar, S., Ganapathy, S., Vijayalakshmi, M., Kannan, A.: Secured temporal log management techniques for cloud. Proc. Comput. Sci. **46**, 589–595 (2015)
53. Muthurajkumar, S., Vijayalakshmi, M., Kannan, A., Ganapathy, S.: Optimal and energy efficient scheduling techniques for resource management in public cloud networks. Natl. Acad. Sci. Lett. **41**(4), 219–223 (2018)
54. Nikravesh, A.Y., Ajila, S.A., Lung, C.H.: Towards an autonomic auto-scaling prediction system for cloud resource provisioning. In: Proceedings of the 10th international symposium on software engineering for adaptive and self-managing systems, pp. 35–45. IEEE Press (2015)
55. Padhy, N., Singh, R., Satapathy, S.C.: Cost-effective and fault-resilient reusability prediction model by using adaptive genetic algorithm based neural network for web-of-service applications. Clust. Comput. **22**(6), 14559–14581 (2019)
56. Park, S.M., Humphrey, M.: Self-tuning virtual machines for predictable escience. In: Proceedings of the 2009 9th IEEE/ACM international symposium on cluster computing and the grid, pp. 356–363. IEEE Computer Society (2009)
57. Prodan, R., Nae, V.: Prediction-based real-time resource provisioning for massively multiplayer online games. Future Generation Computer Systems **25**(7), 785–793 (2009)
58. Qu, C., Calheiros, R.N., Buyya, R.: Auto-scaling web applications in clouds: a taxonomy and survey. arXiv:1609.09224 (2016)
59. Roy, N., Dubey, A., Gokhale, A.: Efficient autoscaling in the cloud using predictive models for workload forecasting. In: 2011 IEEE International Conference on Cloud computing (CLOUD), pp. 500–507. IEEE (2011)
60. Sedaghat, M., Hernandez-Rodriguez, F., Elmroth, E.: 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, p. 6. ACM (2013)
61. Shen, Y., Chen, H., Shen, L., Mei, C., Pu, X.: Cost-optimized resource provision for cloud applications. In: High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), 2014 IEEE Intl Conf on, pp. 1060–1067. IEEE (2014)
62. Shen, Z., Subbiah, S., Gu, X., Wilkes, J.: Cloudscale: elastic resource scaling for multi-tenant cloud systems. In: Proceedings of the 2nd ACM Symposium on Cloud Computing, p. 5. ACM (2011)
63. Singh, P., Gupta, P., Jyoti, K.: Tasm: technocrat arima and svr model for workload prediction of web applications in cloud. Clust. Comput. **22**(2), 619–633 (2019)
64. Singh, S., Chana, I.: A survey on resource scheduling in cloud computing: issues and challenges. J. Grid Comput. **14**(2), 217–264 (2016)
65. Subbulakshmi, P., Sumathi, V., Ganapathy, S.: Cloud based pos system for secured smart shopping cart using rfid. J. Adv. Res. Dyn. Control Syst. **9**(Sp–14), 2764–2777 (2017)
66. Vijayakumar, P., Bose, S., Kannan, A.: Chinese remainder theorem based centralised group key management for secure multicast communication. IET Inf. Secur. **8**(3), 179–187 (2014)
67. Xiang, S., He, J.: Database authentication watermarking scheme in encrypted domain. IET Inf. Secur. **12**(1), 42–51 (2017)
68. Yang, R., Hu, C., Sun, X., Garraghan, P., Wo, T., Wen, Z., Peng, H., Xu, J., Li, C.: Performance-aware speculative resource oversubscription for large-scale clusters. IEEE Trans. Parallel Distrib. Syst. **31**(7), 1499–1517 (2020)
69. Yazdanov, L., Fetzer, C.: Lightweight automatic resource scaling for multi-tier web applications. In: 2014 IEEE 7th International Conference on Cloud Computing, pp. 466–473. IEEE (2014)
70. You, G., Wang, X.: A server-side accelerator framework for multicore cpus and intel xeon phi co-processor systems. Clust. Comput. pp. 1–18 (2020)

Parminder Singh is working as Associate Professor in the School of Computer Science and Engineering, Lovely Professional University, Punjab, India. He received his Ph.D. from Lovely Professional University in 2019. He has received M.Tech degree in Computer Engineering from Punjab Technical University, India. His research interests include Machine Learning, Deep Learning, Blockchain, Cloud/Fog/Edge Computing, Network Security and Web services. He has more than 30 papers in SCI/SCIE,

Scopus indexed journals, conferences and book chapters. He has been session chair and advisory member for various International Conferences. He is an active member of IEEE.

Avinash Kaur is an Associate Professor in the Department of Computer Science and Engineering of Lovely Professional University with specialization in cloud computing. She received her Ph.D. in Computer Science and Engineering from Lovely Professional University, India, M.Tech in Computer Science and Engineering from GNDEC, Ludhiana, Punjab Technical University, and B.Tech in Computer Science and Engineering from GNDEC, Ludhiana. Her research interests includes cloud computing, fog computing, IoT and data mining. She has more than 15 publications in reputed journals, conferences and book chapter. She is currently supervising 3 Ph.D. candidates in the area of cloud computing. More than 5 master's thesis has been completed so far under his supervision.

Pooja Gupta received her Ph.D. in Information Technology at Lingaya's University, India. She is an Associate Professor at the Department of Computer Science Engineering, Lovely professional University. Her research interests are related to wireless networking, web mining, cloud computing. She has published various research papers at national and international journals, conference proceedings and book chapters. She is currently guiding 6 Ph.D. students.

Sukhpal Singh Gill is a Lecturer (Assistant Professor) in Cloud Computing at School of Electronic Engineering and Computer Science , Queen Mary University of London, UK. Prior to this, Dr. Gill has held positions as a Research Associate at the School of Computing and Communications, Lancaster University, UK and also as a Postdoctoral Research Fellow at CLOUDS Laboratory, The University of Melbourne, Australia. His research interests include Cloud Computing, Fog Computing, Software Engineering, Internet of Things and Healthcare. For further information, please visit http://www.ssgill.me.

Kiran Jyoti received her Ph.D. degree in Computer Engineering from Shri JJT University, Rajasthan. She is an Assistant Professor in the Department of Information Technology, Guru Nanak Dev Engineering College, Ludhiana. Her research interests are related to data mining. She has published 24 research papers at national and international journals, conference proceedings. She is currently guiding 6 Ph.D. Students and more than 18 master's thesis completed under her guidance.