A Novel Approach on Auto-Scaling for Resource Scheduling Using AWS

I. George Fernandez and J. Arokia Renjith

Abstract The mostly wide adoptions in cloud computing for businesses have the several reason; among that the elasticity in cloud computing is the virtual infrastructures which will be the dominating leader. The most important area is elasticity that allows to auto-scale the resources which are on-demand. On the other hand, Web applications typically comprise dynamic workload and are hard to predict. The cloud service providers and researchers are jointly working together to condense the cost at the same time as maintaining quality of service (QoS). AWS command-line interface (CLI): the command-line tools, which is Python written, introduce the efficient use cases for managing the AWS services through the sets precisely modest in command. Forecasting on historical data by way of inputting in order to make the up-to-date assessment which are analytical in modeling a path for forthcoming trends. Prophet which acts as a tool will be built to report such concerns, and it delivers real-world methodology in forecasting that is at scalable. From the experimental result, it is found that when predicted value is greater than 51% of CPU usage, then a new EC2 instance is created.

Keywords Amazon web services · Amazon EC2 · Auto-scaling · Prophet · Time series · Elastic load balance

1 Introduction

Scalability in cloud applications is being main reasons behind the wide adoption for cloud computing. Most of the IaaS cloud services providers (CSP) offer auto-scaling services to familiarize the VMs to the shifting demand. A virtual entity of new type called container, which allows user to proposal the loosely coupled applications

I. George Fernandez (✉)
Department of Information Technology, Jerusalem College of Engineering, Chennai, Tamil Nadu, India

J. Arokia Renjith
Department of Computer Science and Engineering, Jeppiaar Engineering College, Chennai, Tamil Nadu, India

R. J. Kannan et al. (eds.), *International Virtual Conference on Industry 4.0*,
Lecture Notes in Electrical Engineering 355,
https://doi.org/10.1007/978-981-16-1244-2_8

containing of several small building blocks. These building block such as micro-services implements a minor fixed of functions and interconnect with other micro-services. In auto-scaling, the decision-making technique provisioned to allocate the number of resources to divergent process is classified as reactive and proactive. The reactive technique regularly monitors events, namely CPU utilization, workloads, queues, etc., and it performs the elastics operation on resources based on threshold. In proactive techniques, forecasting methods are used to predict traffic from past workload.

Bitbrains, a service provider, which is specialized in business computation and managed hosting for enterprises. The requirements would come in form of request such as data transfers among the customer, and the data center through protected networks, the calculate nodes are rented as virtual machines (VM) cutting-edge data center which delivers a expectable routine and provide in elevation obtainability for consecutively of business-critical simulation.

Prophet is a technique aimed at foretelling, time series statistics which are constructed on an additive model such that nonlinear trends is appropriate with the annual or weekly and on everyday seasonality, in addition to leave effects. This work finest through the time series that invent healthy periodic effects and numerous spells of the past data.

Time series forecasting finds significant problem with various industrial applications such as the trade request estimating, economic forecasts, prediction of stream of traffic, or the climate patterns. In broad, these shows a crucial part in the industrializing commercial developments. These methods are included in the well-known auto-regression (AR), ARIMA, the classical Box-Jenkins methodology, exponential smoothing, and more precisely linear state-space models. Conversely, these approaches will not certainly accessible to enormous datasets through the lots of time series, owed the essential for the specific training. Furthermore, they will not help from the mutual temporal configurations in whole dataset although the prediction and training.

Amazon Web services (AWS): Amazon provides auto-scaling services in particular IaaS public cloud. Amazon Web services auto-scaling is precisely auto-scaling group (ASG). The auto-scaling group is a fixed of various Amazon elastic compute cloud (EC2) instance, the virtual machines (VM's) which are sharing the similar. Consequently, each VM pioneering the collection consumes the similar Amazon machine image in addition the identical hardware features. Load distribution among the virtual machines (VMs) are automated by elastic load balancer (ELB). Amazon cloud watch is performance monitoring tool, be responsible for the performance data cast-off in scaling rules.

1.1 Reactive Auto-Scaling

Reactive auto-scaling (automatic scaling) technology that enabled for automatic provisioning as well as termination of the virtual entities to familiarize the resource

capacity for changing demand. The automation is accomplished via a monitoring service to retrieve related resource utilization metrics by which the alarms and also triggers can be defined. Reactive auto-scaling is one such type of auto-scaling which will either deploy or will terminate based on predefined amount of the virtual entities as a response to change of some metrics. Therefore, reactive auto-scaling proceeds only on the basis of given parameters, which can be either provided or measured by the monitoring solution or by the cloud administrator. The number of changes will be the number of virtual entities which are allocated or terminated is encoded based set of rules.

1.2 Predictive Scaling

Predictive scaling or proactive scaling influences historical data just about the virtual infrastructure and the application which is collected via the monitored solutions. Collected historical data are in different forms, namely: logs, application traces, time series, etc. These data will be required in the derivation of the prototypes that are used to generalize, future values of the particular metrics. For example, the poised requests per second time series are used to develop a model and to forecast such as predict or extrapolate, the requests per second values for a particular service at some instant in near future.

2 Literature Review

Papadopoulos et al. [1] proposed an auto-scaling policy on performance evaluation approach constructed on a chance constrained optimization problem explained by means of scenario theory. The approach was implemented in performance evaluation framework for auto-scaling (PEAS) and tested on the several existing auto-scaling policies using 796 real workload traces. The work introduced a numerous distinct metrics to estimate the autoscaling performance with the core metrics of the average number of under provisioned resources and over provisioned resources. Ilyushkin et al. [2] proposed a set of performance metrics to estimate the auto-scaling policy. The set which includes, namely wrong-provisioning timeshare, under- and over-provisioning accuracy, instability as well as the other user-oriented metrics such as elastic slow-down, response time, wait time, average task throughput, or average number of resources. Bauer et al. [3] investigated predictive auto-scaling solution: (a) collects the monitoring data for forecasted parameters, (b) derives forecast models, (c) derives the virtual infrastructure and the application performance models, (d) derives the scaling policy to guarantee provision of virtual entities that are be able to assist forecast workload, and (e) finally executes the scaling action for forecast [4]. Anshul et al. [5] proposed experimental evaluations of the multilayered auto-scaling, performance with the mixture on virtual setup auto-scaling of Google compute engine,

AWS, and Microsoft Azure by the pods for flat auto-scaling of the Kubernete via demonstrating ScaleX based on four typical patterns of load. Benjamin et al. [6] demonstrated performance analysis [7] and for sake of comparisons and evaluated the forecasted procedures and also introduced a tool which enable the analyst to make use of their expertise reliable and forecasting business time series more practically. Fang et al. [8] analyzed that prophet and LSTM which are used in prediction for the trends of the time series data, and author studied that prediction trends can be united by the contrary neural network exemplary aimed at the prediction. Shen et al. [9] analyzed business-critical workloads hosting in distributed data center with 1,750 virtual machines workload traces of long term and large scale. Traces are also analyzed on actual resource usage as well as requested resources, in terms of CPU, memory, disk I/O, and network I/O. Walid et al. [10] proposed the optimization framework which can adaptively solve the joint VM-to-PM packing problem and VM auto-scaling. Prathanra et al. [11] investigated that the performances of the machine knowledge prototypes can forecast recital of the Jupyter notebook on the JupyterHub popular relationships of their response time.

3 System Architecture

In the proposed model, complete architecture is depicted (Fig. 1 shows proposed system architecture), which contains five components such as AWS auto-scaling group, elastic load balancer (ELB), prophet, prediction manager, and cloud watch. The various works are allocated according to the various requests by cloud users.

3.1 AWS Auto-Scaling Group

The AWS launch configuration is used to form the auto-scaling group (ASG) in Amazon Web service cloud using the user metrics. The auto-scaling group is a usual of dissimilar Amazon elastic compute cloud (EC2) instances.

3.2 Elastic Load Balancer (ELB)

Load distribution among the virtual machines (VMs) is automated by elastic load balancer (ELB). To direct the loads to the auto-scaling group, the elastic load balancer (ELB) remains added. It functions as a distinct endpoint aimed at the load generations workload, which in turn, the load is disseminated between the ASG instances.

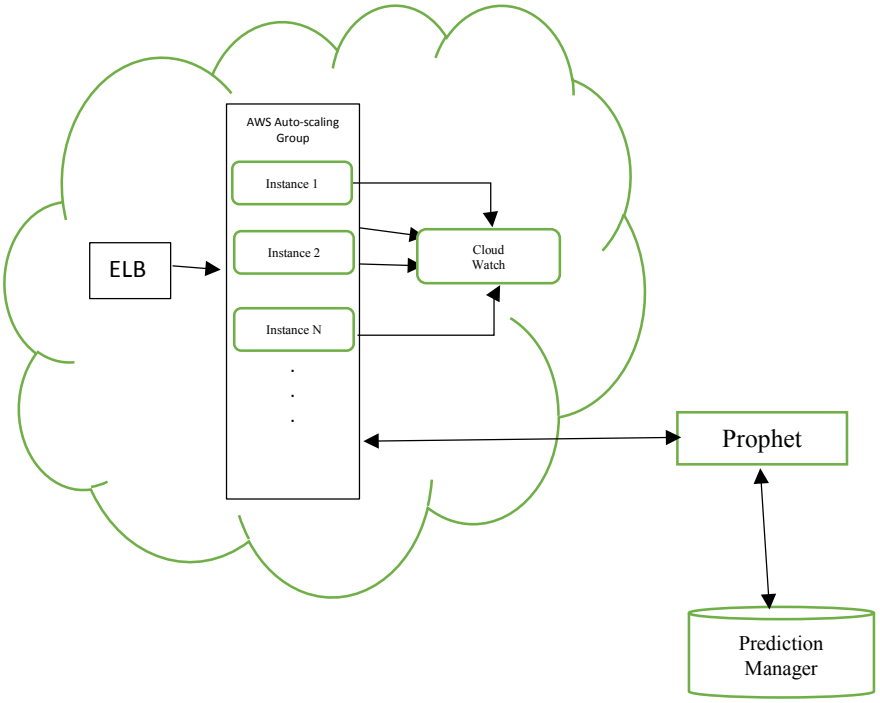

Fig. 1 Proposed system architecture

3.3 Prophet

Prophet is sklearn exemplary API, which can generate an instances of prophet class, call the fit besides prediction methods expectially. The input parameter to the prophet exists data frame has two columns, namely y and ds. The y column necessity must be numeric. The date stamp (ds) column must be of a setup likely by Pandas. It represents the magnitude that anticipated to estimate.

3.4 Prediction Manager

The prediction manager manages the prediction rules that are available to predict according to the prophet predict method to forecast.

3.5 AWS Cloud Watch

Amazon cloud watch is performance monitoring tool, be responsible for the performance data castoff in scaling rules.

4 Proposed Work

In our proposed work, we investigate the active traces explanatory intended at business-critical capabilities. Business-critical capabilities regularly comprise requests on creditworthiness areas, and these are repeatedly Monte Carlo imitation balanced marketable exposing requests. The Auxiliary requests that illustrate business-critical capabilities are database, CRM, email, and management plus collaborative amenities.

4.1 Traces Collected

In proposed work, we obtain traces which cover both actually used and requested resources, basically four types of resource such as memory, CPU, network, and disk. The traces collected from August to September 2013, the bit fastStorage which comprises of 1250 virtual machines (VM's), and fastStorage trace contains a complex section of application servers and the compute nodes.

4.2 Dataset

The records are systematized into three subdirectories in terms of month that each metrics remain documented. The setup specification of individually file is racket wise; each such racket signifies an opinion for concert metrics such as timestamp in milliseconds, CPU usage in percentage, the CPU's are in cores, the CPU's capability in MHZ, memory's capability in kilobyte (KB), the memory's usage in KB, the disk read throughput in kilobytes per second, the disk write throughput in kilobytes per second, network transmitted throughput in kilobyte per sec network received throughput in kilobytes per second.

4.3 AWS Auto-Scaling

Boto3 which is the AWS SDK used for Python that provides a low-level direct access and object-based API's to AWS services such as elastic compute cloud. AWS command-line Iinterface (CLI): The command-line tool which is Python programmed that familiarizes a well-organized use cases for managing the Amazon Web services through established of precise modest instructions.

4.4 Experimental Setup

We install the AWS CLI and the Python Boto3 libraries, in order to create our own consumer authorizations for the Amazon Web service solace, since AWS amenities will remain accessible programmable. After creating the user and obtaining the credentials (Access ID and Secret key). Python scripting environment is configured with credentials for managing EC2 instances. We now build a key pair for EC2 instances, which can be accessed in later stages, so that virtual machines (VMs) are launched programmatically by using the Python.

```
Ec2_instances = elasticcomputecloudec2.Create_AW instances (
node_Id = 'ami-0323c3dd2da7fb37d',
Count_MIN = 1,
Count_MAX = 3,
Type_Instances = 't1.micro',
Name_Key='auto-key pair')
```

From the above program code, Node_ID which stipulates, Amazon machine image (AMI) ID of the AWS EC2 illustration. Count_MIN and Count_MAX that are used to express, the quantity of EC2 occurrences which are to be launched. For example, Count_MIN = 1 and Count_MAX = 3, number of instances launched are said to be 3. The Typt_Instances specify instance size such as: t1.micro, t1.small, or M3.large. The Name_key is name distinct for key pair which will permit to[12] admittance EC2 instances; for example, in our proposed work, we use the name "auto-key pair" which is created in the AWS proposed scenario. The virtual machine configuration in proposed on clouds is given in Table 1. The AWS AMI image for operating system we used for the VM configuration is centos 7.

From Table 2, we encounter minimum instances and maximum instances that remain cast-off to explain, the quantity of EC2 instances which are near be launched. For example, minimum instances are 1, and maximum instances are 3; number of maximum instances launched are said to be 3. The scaling metrics are CPU utilization, and the threshold to launch a new instance is said to be 50%.

Table 1 Virtual machine configuration of proposed

Storage memory	Type_Instance	Virtual CPUs
1 GB	t1.micro	1 CPU

Table 2 Configuration of Amazon web service auto-scaling

Metrics for scaling	Threshold (%)	Minimum instance	Maximum instances
CPU usage	50	1	3

5 Results and Discussion

In proposed work, we obtain traces which are the used and requested resources, here four types of resource such as memory, CPU, network, and disk. The traces collected from August to September 2013, the trace fastStorage which consists of 1250 virtual machines (VMs).

5.1 Dataset Importing

To prophet involvement stays continually a data frame through two stakes, namely y and ds. The ds stands for date stamp stake, always in format predictable in Pandas. For the analysis in our proposed, we are using an excel file that contains a total of "CPU Usage in Percentages," traces collected from August to September 2013, the trace fastStorage which consists of 1250 virtual machines (VMs).

5.2 Converting the Dataset to Prophet Compliant

We can convert the historical dataset to be prophet compliant. Now convert the given data into the formats that are desired by prophet from Table 3. It is renamed as the date: ds and the CPU Usage (%): y.

Table 3 Converted dataset to prophet compliant

	Date	CPU usage (%)
0	2013-08-12 13:40:46	40.866667
1	2013-08-12 13:45:46	42.100000
2	2013-08-12 13:50:46	40.733333
3	2013-08-12 13:55:46	44.000000
4	2013-08-12 14:00:46	42.300000

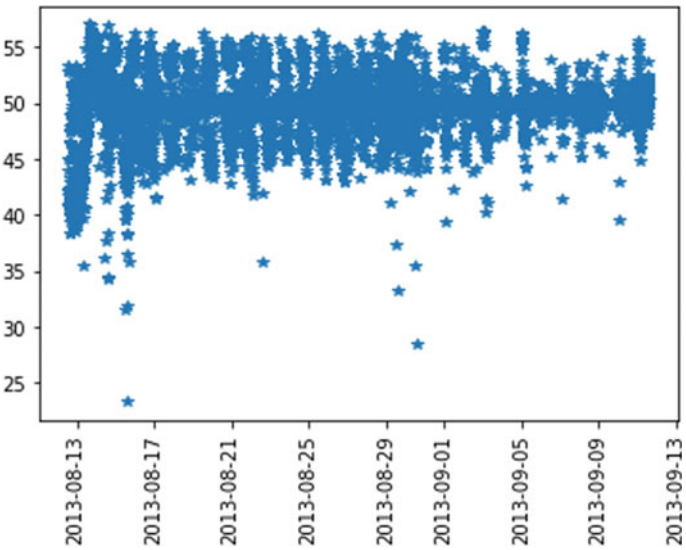

Fig. 2 Forecast plotting of CPU usage versus timestamp

5.3 The Forecasting in Prophet

To forecast, in prophet for prediction in near future. A data frame is created for the purpose of future predictions by making use of make_future_dataframe. Predict the CPU Usage (%) for upcoming 300 s. Forecast plotting using prophet: Forecast can be plotted by calling method as Prophet.plot, and it is passed to forecast data frame.

From (Fig. 2 CPU Usage versus timestamp), we can forecast the CPU Usage in percentage with timestamp, logs from August to September 2013.

5.4 The Proposed Future Prediction

When predicted value is greater than 51% of CPU usage for 300 s in next 50 min, then create new EC2 instance. The forecast technique shall allocate for, respectively, rows in the forthcoming of prediction assessment which it is named as yhat.

Algorithm for Future Prediction

```
Step 1: if len(out[out['yhat']>51])>300
Step 2: r=requests.get(url=\"http://localhost:5000/api/start\")
Step 3:data = r.json()
Step 4:print(data)
Step 5: else
Step 6: x=requests.get(url=\"http://localhost:5000/api/stop\")
Step 7:data2 = x.json()
```

```
Step 8:print(data2)
Step 9: end
```

From the (Fig. 3 future prediction), it represents the quantity we desire to forecast, and we can predict that when the CPU Usage increased above 51 percentages; then, a new instance of AWS EC2 is created.

From the (Fig. 4 CPU Utilization), the cloud watch monitoring tool of AWS experimental result found that only about 9% of CPU utilization is done.

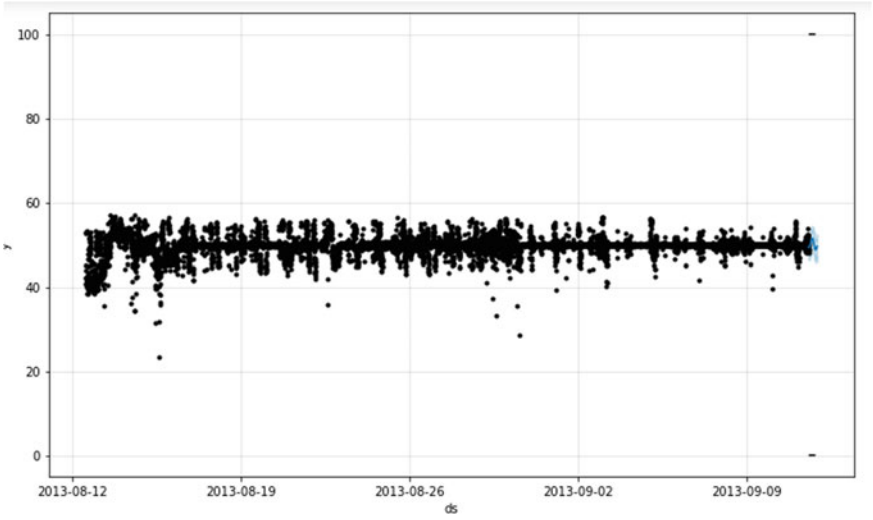

Fig. 3 Future prediction when CPU Usage above 51%

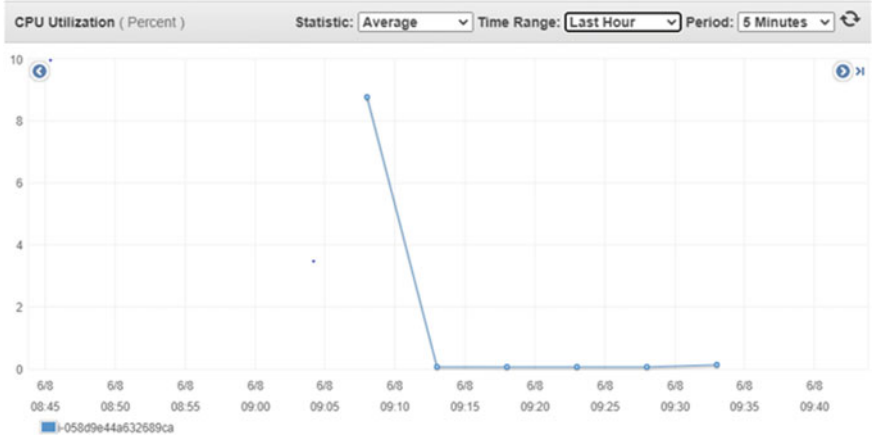

Fig. 4 CPU utilization in terms of percentage vs time interval

6 Conclusion

In this paper, the traces collected from August to September 2013, the trace fastStorage which consists of 1250 virtual machines (VMs). We use the Python, namely: Pandas, Numpy, and Matpoltlib are the modules which are used for analysis and the transformation. In our proposed work, we have converted the historical dataset to be prophet compliant. Implemented AWS auto-scaling for predicting the future workload by applying algorithm for future prediction. Prophet acts as tool, real-world methodology in forecasting that are at scalable.

The main finding of the proposed work is:

(i) When predicted value is greater than 51% of CPU usage for 300 s in next 50 min, then create new EC2 instance.
(ii) We found that only about 9% of CPU utilization is done.

References

1. Papadopoulos AV, Ali-Eldin A, Arzen KE, Tordsson J, Elmroth E (2016) PEAS: A performance evaluation framework for auto-scaling strategies in cloud applications. In: ACM transactions on modeling and performance evaluation of computing systems
2. Ilyushkin A, Ali-Eldin A, Herbst N, Papadopoulos AV, Ghit B, Epema D, Iosup A (2017) An experimental performance evaluation of autoscaling policies for complex workflows. In: Proceedings of the 8th ACM/SPEC on International conference on performance engineering, ICPE'17. L'Aquila, Italy, pp 75–86
3. Bauer A, Herbst N, Kounev S (2017) Design and evaluation of a proactive, application-aware auto-scaler: tutorial paper. In: Proceedings of the 8th ACM/SPEC on International conference on performance engineering, ICPE'17. L'Aquila, Italy, pp 425–428
4. Podolskiy V, Jindal A, Gerndt M (2019) Multilayered autoscaling performance evaluation: can virtual machines and containers co–scale. Int J Appl Math Comput Sci
5. Jindal A, Podolskiy V, Gerndt M (2017) Multilayered cloud applications autoscaling performance estimation. In: IEEE 7th International symposium on cloud and service computing
6. Sean J. Taylor, Benjamin Letham (2017) Forecasting at Scale https://doi.org/10.7287/peerj.preprints.3190v2
7. Taylor SJ, Letham B (2017) Forecasting at scale 10.7287
8. Fang W-X, Lan P-C, Lin W-R (2019) Combine Facebook prophet and LSTM with BPNN forecasting financial markets: Morgan Taiwan Index. In: International symposium on intelligent signal processing and communication systems (ISPACS)
9. Shen S, Van Beek V, Iosup A (2015) Statistical characterization of business-critical workloads hosted in cloud datacenters
10. Guo Y, Stolyar A, Walid A (2018) Online VM auto-scaling algorithms for application hosting in a cloud. IEEE Trans Cloud Comput 1–1
11. Prathanrat P, Polprasert C (2018) Performance prediction of Jupyter notebook in JupyterHub using machine learning. ICIIBMS
12. https://aws.amazon.com/sdk-for-python