An Effective Workload Prediction with Rnn-Lstm For Efficient Resource Autoscaling In Private Cloud Environments

Narek Badjajian[*1], Sandy Montajab Hazzouri[2]

[1] University of Debrecen, Department of Mathematical and Computational Science, Debrecen, Hungary

[2] Faculty of Informatics Engineering , Albaath University, Syria

Emails: badjajiann6math@gmail.com; Samonhaco1994@gmail.com

Abstract

The research focuses on an accurate workload prediction approach for auto-scaling resources in the Private Cloud using improved Time-Series models. Although many factors still result in dynamic workloads of cloud systems, an accurate forecast becomes vital for service quality and cost. The chapter discusses a Proactive Prediction Engine (PPE) framework using Auto Regressive Integrated Moving Average (ARIMA) and Recurrent Neural Network Long Short-Term, to forecast CPU utilization. Real-time datasets of OpenStack private cloud and Amazon AWS were used for experimental evaluation. The analyses show that the RNN_LSTM model performs far better than ARIMA by reducing the MAE and RMSE values by roughly 40 percent in each set. This has further reinforced that RNN_LSTM can model non-linearity and handle correlation issues in the workload data. Automated scaling of the instances with the Open Stack based on the predicted CPU load is made possible by the integration of RNN_LSTM prediction with OpenStack, supported by Terraform. This strategy reduces times of service outages and enables the efficient use of resources in the network. Regarding accuracy and automation, the proposed method can be a relevant solution for workload management for private cloud infrastructure. In this respect, the results support the implementation of deep learning-based predictive models to optimize the performance of autoscaling.

Keywords: Proactive Prediction Engine (PPE); Workload Prediction; ARIMA; RNN_LSTM; Hybrid Cloud; Deep Learning and Dynamic Autoscaling.

1. Introduction

Auto-scaling is one of the maximum significant topographies of the next generations of cloud computing, allowing systems to rule up and down to meet the difficulties of workloads. This capability provides the ultimate in performance, dependable user satisfaction, and economic advantage for applications in cloud settings. [1] There are several types of cloud deployment; among them, the private cloud has received much attention because of its improved security, control, and compliance. Nevertheless, handling resources in a private cloud has its challenges especially when it comes to achieving the aspect of cost-effective while delivering [2] high-quality services.

At its core, autoscaling can be categorized into two primary types: vertical scaling which involves the scaling both up and down by the modification of capacities of the current resources, and horizontal scaling which involves scaling both in and out by adding or eliminating instances. One kind of scaling is horizontal [3] scaling, which is common in cloud architecture; it comes with the option to add or release VMs to or from the system as needed. This dynamic approach guarantees the availability of the required resources during high use and the absence of any unused extra resources during low utilization.

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

The cost factor turns out to be central to the strategy of autoscaling. For resource over-provisioning there is a problem in wasted resources leading to high operational costs [5] For resource under-provisioning, the impact is that it leads to violation of SLA and low value of provision for the consumers. In private clouds resources as compared to the public cloud, are limited therefore getting a balance between cost and performance is crucial. To be able to achieve this balance, one has to incorporate predictive algorithms, as well as automated tools in the autoscaling.

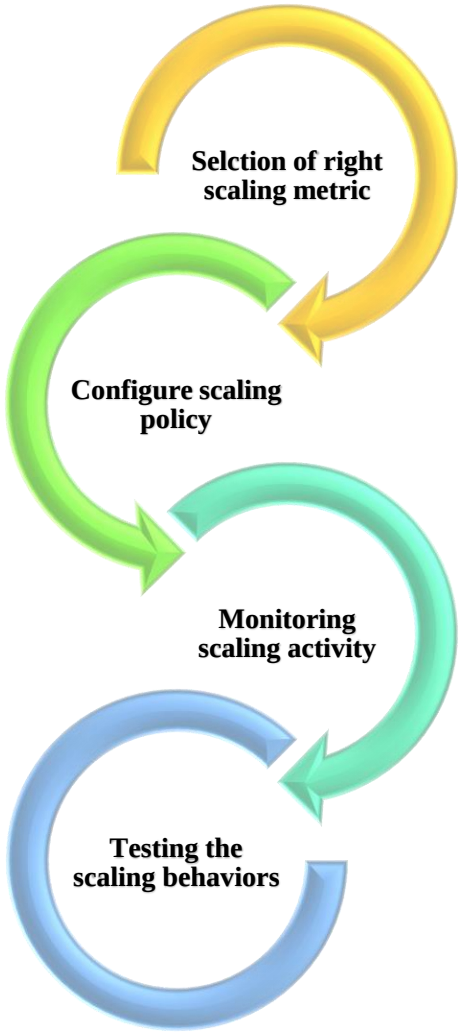

Figure 1: Illustration of Auto scaling for cost efficiency

1.1 Need for Cost Optimization

Service cost, the unique characteristic of cloud computing that is distinguished by public clouds' pay-as-you-use approach, demonstrates the financial risks of resource overprovisioning. As for the private cloud, the unshared service model [7] is not subscription-based, however, low resource utilization results in high CapEx and OpEx. Stopped virtual machines take up power and processing capabilities while they are not utilized to enhance application functionality, which is unprofitable.

Cost-optimized autoscaling tries to solve these issues by gaining a proper estimation of the workloads to provision the corresponding resources. [9] This encompasses a process of predicting future volumes of work based on past data to prepare for workloads in advance. Using predictive analytics, private cloud systems do not have to over-provision or under-provision a system to ensure that the applications will always run properly, consequently, [11] keeping costs to a minimum.

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

1.2 Predictive analytics and Autoscaling

Earlier conventional techniques of autoscaling involve proactively scaling resources, in which additional resources are added once a certain threshold is reached, for instance, once CPU or Memory usage goes up. Nevertheless, reactive scaling serves only partly and may be slow, thus creating temporary performance issues or squandering resources during volatile load increases.

While, on the other hand, predictive analytics must use [13] complex algorithms and analyze resource requirements before a demand appears. The Workload prediction can be done through time-based models like ARIMA and the more advanced techniques of machine learning like RNN_LSTM. These models consider the antique use data to establish an approximate pattern in resource use to factor in future expectations.

The analysis of data in private clouds can be predictive and its application can save a huge amount of money. Through predictive fabrics, organizations can [15] make provision for resources that are likely to be in high demand to prevent over-provisioning costs but still maintain the necessary capacity that with face during busy periods. In addition, the predictive models can be built to include the load-shaping metrics that are nonlinear and dynamic in real-life handling of workload.

1.3 Automation and Integration

This is the essence of cost-effective autoscaling: Automation. Some of the examples include Terraform where companies have an infrastructure-as-code [16] platform to scale resources such as VMs. Terraform will work well with other private cloud solutions such as OpenStack to enable the creation of a sound framework for predictive autoscaling.

In this context, a typical autoscaling workflow involves the following steps:

- Data Collection: Using Telegraph, much information is gathered from the cloud infrastructure including CPU utilization, memory usage, and traffic.
- Data Storage and Visualization: Metrics aggregated are stored in time series databases such as Influx DB while visualizations of the metrics are done using tools such as Grafana for the display of patterns.
- Prediction: In this case, [17] models like the RNN_LSTM use the stored data to feed into the model and get an approximate of future workload.
- Autoscaling Execution: Going by the predictions, terraform auto-scales resources where resources in the form of VMs are added or removed.

The combination of these components makes for accurate integration to enhance efficiency and decrease the number of human interactions needed, expedite response time, and cut costs.

Nonetheless, the implementation of cost-optimized [18] autoscaling comes with some challenges. Workload prediction is dependent on high-quality data and sophisticated algorithms that can conveniently capture the intricacies and dynamics of workloads. Secondly, the use of predictive models and automation tools can only be done by personnel with adequate knowledge of how the models and tools work and how they should be set in a way that works harmoniously.

The second important issue is the choice between more [19] accurate calculations and less computational complexity. Even in cases, where deep learning models like RNN_LSTM provide better accuracy in the results, the amount of computation needed to train and execute the model might be very high. One must understand that there are always pros and cons, and it depends on the case that an organization needs to solve or achieve.

This type of autoscaling levels up cost optimization [20] along with behavioral analysis to create a perfect equilibrium between the usage of cloud resources and their cost. This is especially evident in private cloud environments where resources are quite scarce to meaningfully address dynamic workloads. If it is recognized by organizations, tools including Terraform, predicted model, and visualization platform can make cloud infrastructure better, and guarantee scalability for cloud cost in the future.

2. LITERATURE REVIEW

Preliminary studies focus on contemplating [21] weaknesses of existing autoscaling systems related to cloud computing. It especially focuses on approaches and methods to scale resources in hybrid cloud environments. This work seeks to build the premises for developing new ideas on

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

how to further advance cost optimality, resource management, and capacity to [22] accommodate users' set budget limits and temporal restrictions.

Author(s)	Year	Title	Methodology	Key Findings	Relevance
Arabnejad et al. [4]	2016	Fuzzy Rule-Based Controller for Autoscaling	Fuzzy logic integrated with Q-learning	Implemented active supply apportionment to enhance the scalability of decision-making based on reinforcement learning.	Stresses the likely possibilities of enhancing autoscaling effectiveness in cloud systems by applying AI techniques.
Nilabja Roy et al.	2017	Predictive Model for Workload Forecasting [6]	ARIMA and Look-ahead optimization techniques	Fewer SLA violations and efficient cost control of resources by predicting workloads with the help of past data.	Introduces the idea that a proper approach towards cloud resource management requires certain acts in anticipation of its occurrence.
Bauer [8] et al.	2019	Chameleon: Hybrid Proactive Autoscaling Mechanism	Box-Cox and ARMA models for prediction	Better resource management but did not include cost models to adapt the workload during short-term fluctuations.	Illuminates combined strategies for resource management.
Hirashima [10] et al.	2016	Proactive-Reactive Autoscaling for Unpredictable Loads	ARIMA and Threshold-Based Rules	High SLA compliance for the hybrid approach but there is a challenge with dealing with short-term alterations.	Stresses the benefits of using both the systems approach and the event-oriented approach.
Chudasama et al.	2020	Bidirectional LSTM for Resource Prediction [12]	LSTM integrated with queuing theory	Improved the accuracy of SLA violation predictions and resource utilization optimization.	Supports methods of deep learning for predicting resources in hybrid clouds.

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

Pham et al.	2020	Cloud Bursting Strategy for Hybrid Clouds	OMNI module for VM [14] migration between clouds	Lower synchronization times about the movement of resources for effective hybrid cloud bursting.	Explains cloud bursting which is particularly useful in hybrid cloud scenarios where peak loads need to be shared between private and public clouds.

When it comes to cloud autoscaling, the survey points to remarkable improvements from raw methods of scaling via simple threshold rules to [23] predictive scaling with methods like ARIMA and machine learning models. Nevertheless, some problems are still inherent, one of which is a shortage of perfect solutions to provide an [24] adequate distribution of resources while considering the level of workload. This study also highlights the need for strong and cheap autoscaling solutions that take advantage of the capability and versatility of a hybrid cloud for efficient response to future needs.

3. RESEARCH OBJECTIVE

In the following, a cost-efficient autoscaling approach aimed toward private cloud infrastructures will be designed using predictions and automated technologies as [25] major components. The key objectives are:

- Design a Proactive Prediction Framework: Private clouds must employ sophisticated forecasting models, ARIMA or RNN_LSTM, to forecast the optimum workload in the cloud.
- Optimize Resource [26] Utilization and Costs: Don't over-provision or under-provision any component, at the same time ensure that the service level agreements are achieved.
- Integrate Prediction with Automation Tools: Integrate such models at the operational level with various automation tools, such as Terraform for dynamic growth in OpenStack.
- Evaluate and Validate Performance: Discuss how to compare the accuracy of a predictive model by employing differing measures including MAE and RMSE and also prove the usefulness of the framework on realistic datasets.

4. PROPOSED WORK

The proposed solution involves predictive analytics and automation to achieve a simple and cheap autoscaling strategy in the context of private clouds. The framework mainly deals with workload prediction through time series model namely ARIMA and RNN_LSTM and works in conjunction with an autoscaling tool namely Terraform.

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

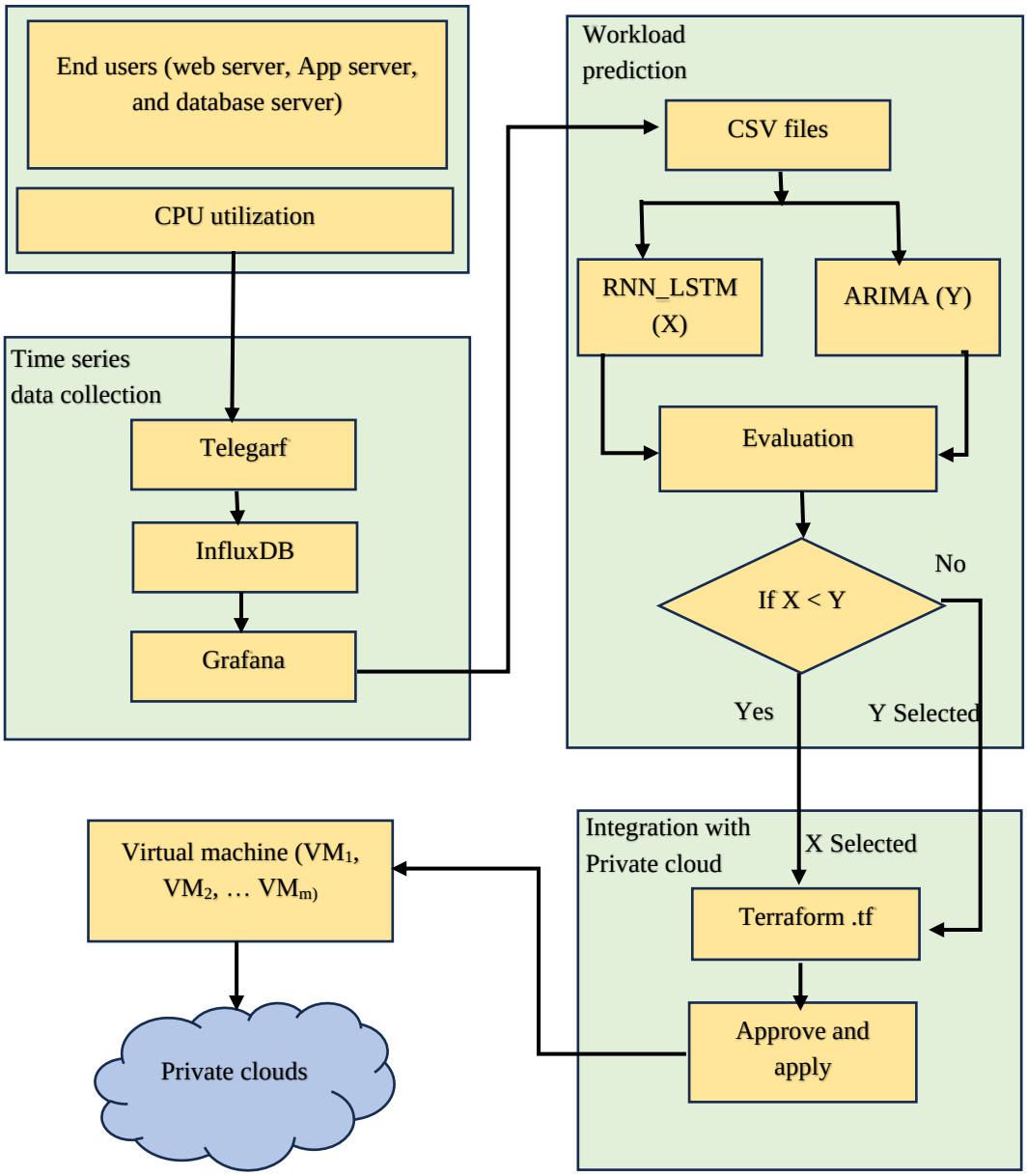

Figure 2: Proactive Prediction Engine Framework

Figure 2 shows the structure of the Proactive Prediction Engine (PPE) and its suggested methodology. There are two phases to adopt PPE. The first stage is to determine which of the two techniques, ARIMA and RNN_LSTM, is more suitable for demand prediction. Step two involves using the estimated amount from an effective approach acquired in step one to determine whether to increase or decrease the instances' virtual machines (VMs).

4.1 Telegraf

An application's website server experiences an exponential growth in demand if more people visit it simultaneously, leading to a rise in CPU and RAM utilize proportional to the number of requests. Because of this, the server goes down. Regular monitoring of these usage parameters is possible with the help of Telegraf. Telegraf is an InfluxDB-stored metric collecting component that takes measurements like CPU usage and saves them from many sources. Due to its nature as an integral part of any input stream, Telegraf never depends on any third-party libraries or applications.

4.2 Influxdb

Built and operated by InfluxData, InfluxDB is a free database for time series information implemented in the Go programming language. To monitor, analyze, and evaluate applications, InfluxDB stores and retrieves time series information in a highly available manner using a SQL-

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

based programming syntax. InfluxDB stores all Telegraf's use information, both historical and present, so it can be retrieved and queried later.

4.3 Grafana

As a control panel, you may use Grafana, a freely available graphing tool, to see how different indicators to be utilized over time are trending. Grafana gets information from the element that is being backed up. You can see InfluxDB and the outcomes of SQL searches on information on the dashboard. Information may be exported from the Grafana display in many forms, such as CSV, XML, and JSON.

4.4 ARIMA

Auto-Regressive Integrated Moving Average is a time series forecasting model useful for linear data in a series. It combines various aspects of autoregression, differencing or the moving average model to follow trends in sequential data. In workload prediction, ARIMA predicts the trends of CPU usage that help in performing an early adjustment of resources in the cloud.

In this context, the best-known and most established techniques involve the use of ARIMA models for analysing and estimating time series data. The model is characterized by three parameters: m, n, and o:

m: The interdependent variable order of autoregression (AR) which explains the number of lagged observations incorporated in the model.

n: Specifically, it is the extent of differencing needed to season the data.

o: The order of moving average (MA), The number of lagged forecast errors in the model.

ARIMA is also denoted as ARIMA (m, n, o) able to deal with trends, seasonality, and the noise in the time series data after pre-processing.

The Autoregressive component for example captures the value of a certain time period with the previous values of the same time period. They employ the use of lagged observations when estimating future points in a set. For instance, AR (1) work with the previous value, and coefficients define these past observations as the bearing on the current value.

$$b_t = c + \sum_{k=1}^{m} \phi_k b_{t-k} + \epsilon_k \quad (1)$$

It is a method of transforming a time series so as to eliminate trends or seasonality with a view of making it stationary. It determines the first difference of current observation ($yt-yt-1$) so as to stabilize the mean and variance to enable easier model estimation and prediction of the accurate values.

$$b_t' = b_t - b_{t-1} \quad (2)$$

The Moving Average component is about how a value is related with past forecast errors. It deals with lagged error terms in forecasting. The system corrects forecast errors. For instance, MA (1) employs the error at the time step t-1 into the current forecast to minimise the noise and overall residual errors.

$$b_t = c + \sum_{k=1}^{0} \theta_k \, \epsilon_{t-k} + \epsilon_t \quad (3)$$

The combined ARIMA equation is:

$$b_t = c + \sum_{k=1}^{m} \phi_k b_t + \sum_{l=1}^{0} \theta_l \epsilon_{t-l} + \epsilon_t \quad (4)$$

Where b_t = current value, ϕ = coefficients for lagged observations, ϵ = Error term.

Algorithm 1: ARIMA for Workload Prediction

1. Input: Time-series information (CPU utilization).
2. Preprocess the information:
   Check stationarity using ADF test.
   Apply differencing if required.
3. Identify m, n and o.
   Use ACF/PACF plots to determine parameters.
4. Fit ARIMA model:
   Train using the identified parameters.

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

5. Forecast future values:
   Generate predictions for the test dataset.
6. Evaluate performance:
   Calculate MAE and RMSE for model accuracy.
7. Output: Forecasted CPU utilization values.

ARIMA offers a good approach towards forecasting linear workload through understanding data patterns of CPU utilization. ARIMA forecasts future resource requirements and helps in advance scaling in cloud environment by using preprocessing, parameter estimation and iterative fitting. Despite being sufficient for stationary and linear workloads, characteristic of low efficiency in confronting nonlinear data show the requirements for an alternative like RNN_LSTM in other circumstances.

4.5 RNN_LSTM

RNN_LSTM is one of the deep learning algorithms which are essentially suitable for working with sequential data and long-term memory knowledge. Through this, it removes the drawbacks of traditional RNN for example, vanishing gradients thus allowing for forecasting of complex non-linear patterns. In workload prediction, RNN_LSTM consistently predicts the future tendency of CPU utilization, so that resource allocation can be done in advance in the different cloud environments.

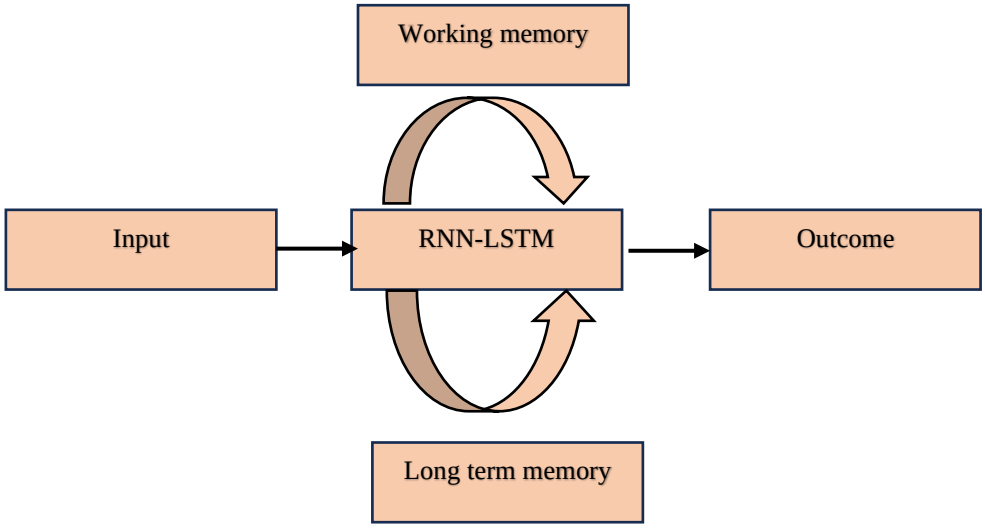

Figure 3: Illustration of RNN_LSTM

A network of linked long short-term memory (LSTM) cells processes information in a sequential fashion:

Forget Gate: Chooses which state information should be deleted in the past example state. It employees a sigmoid function for getting the values between 0 (forget) and 1 (retain).

$$FG_t = \sigma(w_{FG} \cdot [h_{t-1}, a_t] + y_{FG}) \quad (5)$$

Input Gate: Establishes what part of the new knowledge is to be retained. A sigmoid layer to select updates and tanh layer for candidate values are used in this work.

$$IG_t = \sigma(w_{IG} \cdot [h_{t-1}, a_t] + y_{IG}) \quad (6)$$

$$\tilde{CS}_t = tanh(w_{CS} \cdot [h_{t-1}, a_t] + y_{CS}) \quad (7)$$

Cell State Update: Recurrently modifies the memory by receiving information that was retained in the past (forget gate) along with a selected input information (input gate).

$$CS_t = FG_t \cdot CS_{t-1} + IG_t \cdot \tilde{CS}_t \quad (8)$$

Output Gate: Outcomes the final concluding result containing both the output and the hidden state while employing sigmoid function to measure the significance of the output and tanh for scaling the new cell state.

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

$$OG_t = \sigma(w_{OG} \cdot [h_{t-1}, a_t] + y_{OG}) \quad (9)$$

$$h_t = OG_t \cdot \tanh(CS_t) \quad (10)$$

Where FG_t = forget gate, IG_t = input gate, CS = cell state update, $\tilde{C}S_t$ = candidate values of cell state, w = weights, h_{t-1} = previous hidden state, a_t = current input, y = bias.

Algorithm 2: RNN_LSTM for Workload Prediction

1. Input: Time-series information.
2. The normalization of this data should be employed followed by splitting the data into training and testing data bases.
3. Outline the LSTM approach:
   The size of the input layer = number of timesteps.
   Add LSTM layers and neurons.
4. Now get the approach compiled with the loss function and the already defined optimizer.
5. Train the approach:
   Epochs: Number of repetitions.
   Batch size: Number of samples used in each training iteration; it is the number of samples processed to update the model parameter.
6. It will be used to test the approach on unseen information.
7. Tell out how well the approach worked
8. Outcome: Autoscaling predicted workload values.

There is a need to adopt RNN_LSTM models notably proffer an enhanced and robust computational solution for workload prediction in cloud computing scenarios. This characteristic makes them suitable for managing sequential data and learning these long-term temporal dependencies, making them efficient in dynamic and complex scenarios in clouds; the resource usage can be optimized with minimized cost.

4.6 Terraform

Terraform is an opensource Infrastructure-as-Code Devops tool which is used to manage and provision the infrastructure. It enables dynamic autoscaling by running prescribed scripts to enable private cloud systems adapt more resources, depending on the expected workload. This approach promotes efficiency in cost of resource usage and quality of application returned.

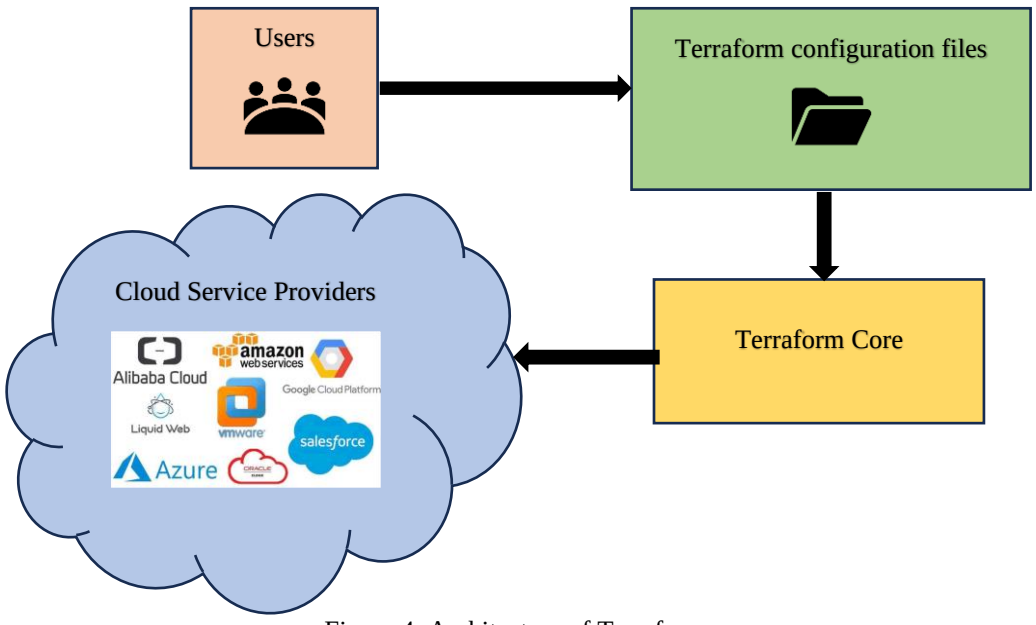

Figure 4: Architecture of Terraform

Terraform Workflow for Autoscaling

1. Create Configuration Files

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

Terraform is implemented using the Hashi Corp Configuration Language (HCL).
These files describe what the desired setup of a cloud solution is, for example virtual machines (VMs), networks, and autoscaling.

2. Initialize Terraform
   The mentioned command is the terraform init which sets up the workflow by finding and installing the required plugins for cloud providers including OpenStack.
3. Plan Infrastructure Changes
   The terraform plan command produces an execution plan that informs users of the changes that Terraform will perform.
4. Apply Changes
   Terraform performs the configurations using the terraform apply command the change provisions or deprovisions virtual machines in response to the configuration.

Auto scaling occurs when the CPU usage forecasts are over or below particular thresholds, such as 60%. The instance_count parameter also gets updated in the Terraform configuration depending on a forecasted value.

$$IC = \frac{Predicted\ workload}{Threshold} \tag{11}$$

Autoscaling with the help of Terraform is the optimized way to manage cloud resources by including in this process the mechanism that takes decisions about how more resources are needed as a result of the predicted workloads. By incorporating best practices of predictive models and IaC, it enables cost optimization of cloud operations making it suitable for private and hybrid clouds.

5. Result

The results section provides a detailed assessment of the proposed workload prediction and autoscaling framework. By comparing the two models, namely ARIMA and RNN_LSTM, the efficacy of the models in predicting the workload patterns from real-world datasets is evaluated. Evaluation criteria, which indicate how accurately the models predict patients, MAE, RMSE. To illustrate how the solutions operate and show that the process of scaling up resources will be performed by the system whilst keeping operating costs low, the integration of outputs from the prediction process with Terraform for autoscaling is examined.

Further, the results show that RNN_LSTM performs better for nonlinear and dynamic workload patterns than ARIMA for linear trends. To support the above hypothesis, the cost implications of the implementation, system efficiency enhancements as well as system response are considered in the real-world private cloud environments. As a result, this evaluation offers valuable suggestions about the practicability and comprehensiveness of the proposed solution for realistic setups.

Accuracy: It is calculation of correct observation out of the total observation made on the model. It gives a general indication of a model's accuracy as a single pass prediction.

$$Acc = \frac{True\ Positive + Tegative\ Negative}{Total\ Obseravtion} \tag{12}$$

Precision: It refers to the sum of true positives to the numeral of positive predictions made. It is concerned with the quality of positive forecasts.

$$Pre = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{13}$$

Recall: It is calculated as the number of true positive observed to the total number of positive actual remarks. It focuses on the potential for accessing all positive results.

$$Rec = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{14}$$

F1 Score: It is designing for evaluating classification model, most commonly in an environment where the false positive and false negative decisions bear different costs.

$$FS = 2 * \frac{Pre * Rec}{Pre + Rec} \tag{15}$$

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

MAE: It finds the average of size of errors in predictions regardless to direction of error. It gives an absolute value as to how accurate a given forecast is to real worth.

$$MAE = \frac{1}{m}\sum_{t=1}^{m}|b_t - \hat{b}_t| \quad (16)$$

RMSE: The square root of the mean of the square of the difference among actual and predicted value is determined. It assigns higher weight to big errors and that is why it is not resistant to outliers.

$$RMSE = \sqrt{\frac{1}{m}\sum_{t=1}^{m}(b_t - \hat{b}_t)^2} \quad (17)$$

Where b_t = Actual value at time t, $\hat{b}_t$ = Predicted value at time t, m = total number of predictions.

Table 1: Analysing the Current Method in Light of the Proposed One

Model	Accuracy %	Precision %	Recall %	F1 Score %
Proposed RNN_LSTM	0.95	0.93	0.97	0.95
ARIMA	0.84	0.8	0.85	0.82
SVM	0.88	0.85	0.87	0.86
Random Forest	0.89	0.83	0.88	0.85

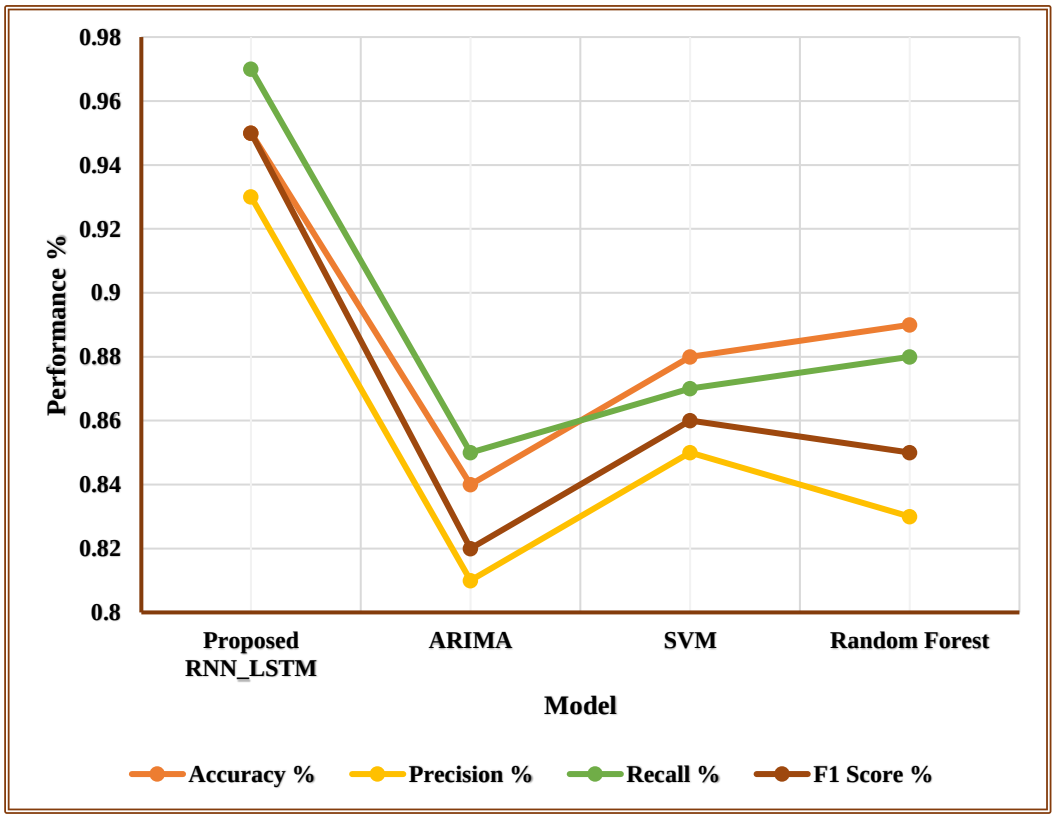

Figure 5: A visual illustration of the proposed technique compared to the current approach

Generally, only the RNN_LSTM-based model has better results, with high precision with great Recall value, which leads to a high F1 Score. Specialized for non-sequential and constantly changing workloads in cloud computing applications. Comparing to linear workloads, ARIMA demonstrates better performance, but in overall it outperforms proposed approach in accuracy, precision, recall and F1 Score because it does not provide efficient tools for handling non-linear data. The recognition performance of the Support Vector Machine (SVM) approach is rather

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

satisfactory but still worse overall compared to RNN_LSTM, notably in terms of recall. Although its shows good results for binary classification problems, it does not successfully address workload prediction in the dynamic cloud environment. Consequently, Random Forest splits give a good model balance, but the F1 Score is a bit lower than that of RNN_LSTM, to point out the possible weakness of Random Forest in featuring the long-term dependency of time series. The statistics have shown that the suggested model, RNN_LSTM, outperforms the current models in all the metrics of better predicting the workloads and thus improving on the optimum scale of resource cloud.

Table 2: Compare the performance of current models with proposed model with respect to MAE and RMSE

Model	MAE %	RMSE %
Proposed RNN_LSTM	0.06	0.24
ARIMA	0.11	0.35
SVM	0.09	0.3
Random Forest	0.08	0.28

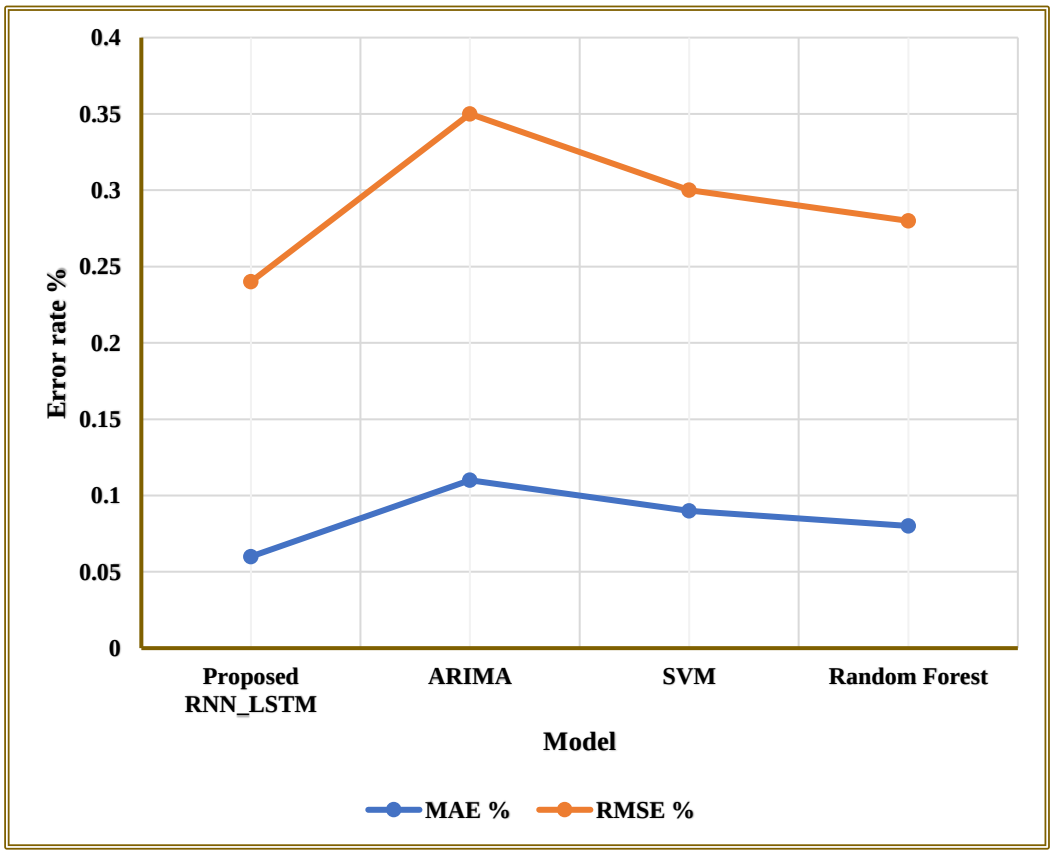

Figure 6: Visual representation of Evaluate the suggested model's performance in comparison to the existing models' MAE and RMSE metrics

The proposed RNN_LSTM state that shows the MAE and RMSE of different models to predict workload is represented as follow, In which the RNN_LSTM proposed approach clearly outperforms other existing approaches like ARIMA, SVM and Random Forest. The Proposed RNN_LSTM model records the lowest MAE which is 0.06% and the lowest RMSE which is 0.24 emphasizing highly efficient prediction with little room for error. This is favorable for complex, dynamic workloads typical for and distinctive to cloud environments owing to its capacity in the detection of non-linear patterns and long-term dependencies in time-series data. We observe that ARIMA yielded the highest MAE score of about 0.11 % and RMSE of about 0.35 %. This is due

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

to the weakness of ARIMA to job interstate and non-sinusoidal workload pattern since ARIMA models fit for linear and homogeneous data. Two other models we used, namely SVM and Random Forest give smaller MAE = 0.09%, RMSE = 0.30% and MAE = 0.08 %, RMSE = 0.28 % respectively than that of ARIMA. Though, it is lower than that obtained by the RNN_LSTM network as shown in the previous table. The disadvantage of SVM is that it has problem in capturing sequential dependencies while Random Forest also lacks a temporal modeling as compare to RNN_LSTM. The findings reveal that as an outcome of the RNN_LSTM the high accurate model for forecasting workload and hence best suitable for dynamic autoscaling of private cloud as it not only minimizes the average errors (MAE) and large deviations (RMSE) but also outperform the other models in our experiments.

6. Conclusion

This project effectively provided the architectural direction on how an autoscaling framework for private clouds can be acquired at an optimum cost while applying analytics for cloud resource management. The paper contrasted two time-series models used for workload prediction, ARIMA, and RNN_LSTM, using workload data to show the enhanced capability of RNN_LSTM to manage non-linear and dynamic workloads. The combination of forecasting with automation tools such as Terraform and OpenStack showed how to apply the results of the model to efficiently allocate resources thereby avoiding over-provisioning and subsequent operational costs. Experimental analysis carried out on real data sets supported the idea of using the proposed framework for ensuring service level agreements (SLAs) without compromising the usage of the resources. Using the resource demand prediction, the Framework scales virtual machines with strategic distributions in a private cloud thereby freeing them from reliance on reactive means such as thresholds for resource allocation. The above causes lead to a substantial increase in system efficiency as well as cost reduction on the system. Finally, the project offers a complete solution to address all the challenges related to the management of private cloud resources and workload control to maximize the efficiency of subsequent processes while keeping costs low and ensuring fast and seamless scalability of the infrastructure. The proposed solution can be seen as a step toward the future in improving cloud infrastructures management, and adapting it to Enterprise-scale application requirements.

The project can be extended in the following ways:

- Incorporate Multi-Cloud Scaling: Expand the autoscaling framework to support cloud hybrid and multi-cloud environments and therefore to allow for the consolidation of resources from the public and private clouds.
- Support for Diverse Workload Types: Optimization of other parameters such as memory usage and I/O operations needs to be incorporated in the predictive models to increase the accuracy in resource workload allocation.
- Energy Efficiency Optimization: Implement efficient resource scaling policies and control to minimize private cloud resources' environmental footprint while minimizing resource management costs.
- Real-Time Adaptive Models: Create models that can learn simultaneously with much potential of responding to performance change without major retraining.

Reference

[1] S. Sotiriadis, N. Bessis, C. Amza, and R. Buyya, "Elastic load balancing for dynamic virtual machine reconfiguration based on vertical and horizontal scaling", IEEE Trans. Services Comput., vol. 12, pp. 319-334, Mar. 2019.

[2] W. Zhang, B. Li, D. Zhao, F. Gong, and Q. Lu, "Workload prediction for cloud cluster using a recurrent neural network", Proc. Int. Conf. Identificat. Inf. Knowl. Internet Things (IIKI), pp. 104-109, 2016.

[3] A. Gandhi, P. Dube, A. Karve, A. Kochut and L. Zhang, "Providing performance guarantees for cloud-deployed applications", IEEE Trans. Cloud Comput., vol. 8, no. 1, pp. 269-281, Jan. 2020.

[4] Mohammed Hasan Aldulaimi, Ibrahim Najem, Tabarak Ali Abdulhussein, M. H. Ali, Asaad Shakir Hameed, M. Altaee, Hatira Günerhan, Intelligent Load Identification of Household-Smart Meters Using Multilevel Decision Tree and Data Fusion Techniques, Journal of

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

Intelligent Systems and Internet of Things, Vol. 9 , No. 1 , (2023) : 24-35 (Doi : https://doi.org/10.54216/JISIoT.090102)

[5] A. Madhuri, Veerapaneni Esther Jyothi, S. Phani Praveen, Mustafa Altaee, Ibrahim N. Abdullah, Granulation-Based Data Fusion Approach for a Critical Thinking Worldview Information Processing, Journal of Intelligent Systems and Internet of Things, Vol. 9 , No. 1 , (2023) : 49-68 (Doi : https://doi.org/10.54216/JISIoT.090104)

[6] N. Roy, A. Dubey & A. Gokhale. "Efficient Autoscaling in the Cloud using Predictive Models for Workload Forecasting", IEEE 4th International Conference on Cloud Computing, pp. 500-507, 2017.

[7] Heba Mohammed Fadhil,Mohamed Elhoseny,Baydaa M Mushgil, Protecting Medical Data on the Internet of Things with an Integrated Chaotic-GIFT Lightweight Encryption Algorithm, Journal of Journal of Cybersecurity and Information Management, Vol. 12 , No. 1 , (2023) : 50-66 (Doi : https://doi.org/10.54216/JCIM.120105)

[8] Vandana Roy, An Effective FOG Computing Based Distributed Forecasting of Cyber-Attacks in Internet of Things, Journal of Journal of Cybersecurity and Information Management, Vol. 12 , No. 2 , (2023) : 08-17 (Doi : https://doi.org/10.54216/JCIM.120201)

[9] B. Joseph, Awotunde, K. Hrudaya K. Tripathy, A. Bandyopadhyay, Hybrid Particle Swarm Optimization with Firefly based Resource Provisioning Technique for Data Fusion Fog-Cloud Computing Platforms, Fusion: Practice and Applications, Vol. 8, No. 2, (2022): 25-35, (Doi: https://doi.org/10.54216/FPA.080203)

[10] Y. Hirashima, K. Yamasaki & M. Nagura. "Proactive-Reactive Auto-Scaling Mechanism for Unpredictable Load Change", 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI), PP. 861-866, 2016, IEEE.

[11] J. Liu, S. Wang, A. Zhou, J. Xu, and F. Yang, SLA-driven container consolidation with usage prediction for green cloud computing, Front. Comput. Sci. Sel. Publ. Chin. Univ., vol. 14, no. 1, pp. 42–52, 2020.

[12] Jayakaran P.,Jeffery matthew S.,Litheeswaran S.,Mohamed Arshad,Mahajan S.,R. Priya, Lip Print Scanner, Journal of Journal of Cognitive Human-Computer Interaction, Vol. 5 , No. 1 , (2023) : 46-52 (Doi : https://doi.org/10.54216/JCHCI.050105)

[13] C. Vivek,M. Indu,N. Nandhini, Speech Recognition Using Artificial Neural Network, Journal of Journal of Cognitive Human-Computer Interaction, Vol. 5 , No. 2 , (2023) : 08-14 (Doi : https://doi.org/10.54216/JCHCI.050201)

[14] B. Pham, R. C. Jones & M. Shaalan. "Analysis of Cloud Bursting from the Open Stack Infrastructure to AWS", 2020 IEEE Cloud Summit, pp.114-118, 2020, IEEE.

[15] C. Xia, M. Zhang, and J. Cao, "A hybrid application of soft computing methods with wavelet SVM and neural network to electric power load forecasting", J. Elect. Syst. Inf. Technol., vol. 5, no. 3, pp. 681-696, Dec. 2018.

[16] S. Phani Praveen, Balamuralikrishna Thati, Ch Anuradha, S. Sindhura, Mohammed Altaee, M. Abdul Jalil, "A Novel Approach for Enhance Fusion Based Healthcare System in Cloud Computing, Journal of Intelligent Systems and Internet of Things", Vol. 9, No. 1, (2023): 84-96 (Doi: https://doi.org/10.54216/JISIoT.090106)

[17] Harith Yas , Manal M. Nasir, Quality of Service in Mobile Adhoc Networks with Non-Saturation Conditions, International Journal of Wireless and Ad Hoc Communication, Vol. 5 , No. 2 , (2022) : 64-76 (Doi : https://doi.org/10.54216/IJWAC.050205)

[18] Amel Ali Alhussan , Hassan K. Ibrahim Al-Mahdawi , Ammar Kadi, Spam Detection in Connected Networks Using Particle Swarm and Genetic Algorithm Optimization: Youtube as a Case study, International Journal of Wireless and Ad Hoc Communication, Vol. 6 , No. 1 , (2023) : 08-18 (Doi : https://doi.org/10.54216/IJWAC.060101)

[19] C. An and J. T. Zhou, Resource demand forecasting approach based on generic cloud workload model, in Proc. IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), Guangzhou, China, 2018, pp. 554–563.

[20] Tarek Gaber , Chin-Shiuh Shieh , Yuh-Chung Lin , Fatma Masmoudi, Modified Flower Pollination Algorithm based Resource Management Model for Clustered IoT Network, International Journal of Wireless and Ad Hoc Communication, Vol. 4 , No. 2 , (2022) : 97-106 (Doi : https://doi.org/10.54216/IJWAC.040205)

[21] A. Anwar, M. Mohamed, V. Tarasov, M. Littley, L. Rupprecht, Y. Cheng, N. Zhao, D. Skourtis, A. S. Warke, H. Ludwig, et al., Improving docker registry design based on

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025

production workload analysis, in Proc. 16th USENIX Conf. File and Storage Technologies, Oakland, CA, USA, 2018, pp. 265–278.

[22] V. Roy et al., "Network Physical Address Based Encryption Technique Using Digital Logic", International Journal of Scientific & Technology Research, Vol. 9, No. 4, 2020, Pp no.- 3119-3122.

[23] O. Poppe, Q. Guo, W. Lang, P. Arora, M. Oslake, S. Xu, and A. Kalhan, Moneyball, Proc. VLDB Endow., vol. 15, no. 6, pp. 1279–1287, 2022.

[24] M. Sumithra, Kiruthika.S, Nithya S, Poornima B, DharanyaS, Enhancement Of Cloud User Data Access Security Entrusted to AI Face Recognition Techniques, Journal of Cognitive Human-Computer Interaction, Vol. 2, No. 2, (2022): 60-64 (Doi: https://doi.org/10.54216/JCHCI.020204)

[25] A. Newell, D. Skarlatos, J. Fan, P. Kumar, M. Khutornenko, M. Pundir, Y. Zhang, M. Zhang, Y. Liu, L. Le, et al., RAS: Continuously optimized region-wide datacenter resource allocation, in Proc. ACM SIGOPS 28th Symp. on Operating Systems Principles, Virtual Event, Germany, 2021, pp. 505–520.

[26] R. Vandana et al., "Detection of sleep apnea through heart rate signal using Convolutional Neural Network", International Journal of Pharmaceutical Research, Vol. 12, No. 4, Oct-Dec 2020, Pp No. 4829-4836.

Doi: https://doi.org/10.54216/IJAACI.070105
Received: October 09, 2024 Revised: December 22, 2024 Accepted: January 17, 2025