Indian Journal of Science and Technology, Vol 12(8), DOI: 10.17485/ijst/2019/v12i8/141807, February 2019
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

A Hybrid Auto Scaling for Cloud Applications Using an Efficient Predictive Technique in Private Cloud

J. Fenila Naomi and S. Roobini

Department of Computer Science and Engineering, Sri Krishna College of Engineering and Technology, Kunia-muthur – 641008, Coimbatore, Tamil Nadu, India; fenilanaomij@skcet.ac.in, roobinis@skcet.ac.in

Abstract

Background/Objective: To provide an efficient predictive technique to foresee future workload as well as to handle the resources efficiently by performing hybrid auto scaling for Cloud applications. Cloud applications might expertise completely different workload at different times, automatic provisioning has to work with efficiency at any point of time. Auto scaling is a feature of cloud computing that potentially scale the resources in line on demand. Considering this expectation, they are generally categorized into Reactive scaling which adds or reduces resources based on a fixed threshold value. The predictive scaling is used provide necessary scaling actions beforehand. **Methods/Statistical Analysis**: To perform the hybrid auto scaling (reactive plus predictive auto scaling), a time series technique should be used. Auto-regressive Moving Average (ARMA) model, the Exponential Smoothing (ES) model, the Autoregressive model (AR), the Moving Average model (MA) and the Trend- Adjusted Exponential Smoothing (TAES), Auto Regressive Integrated Moving Average (ARIMA) Time-series model, Naïve bayes algorithm, Recurrent Neural Network- Long Short Term Memory (RNN-LSTM), Independent Recurrent Neural Network (IndRNN) are time series techniques used to foresee the future workload. To find the effectiveness of predictive techniques, Mean Absolute Error (MAE), Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) performance metrics are evaluated. **Findings:** Based on the evaluation, IndRNN gives the minimum error rate. IndRNN is used to predict the future resource requisites in order to ascertain adequate resource are available ahead of time. **Application:** The predicted result from IndRNN method is integrated on private cloud to autoscale the resources for cloud applications.

Keywords: Cloud Applications, Hybrid Autoscaling, Independent Recurrent Neural Network (IndRNN), Private Cloud, Workload

1. Introduction

Cloud applications are the foremost common applications in today's technology[1]. They are characterized by resource needs that fluctuate with usage, predictably or erratically. Resource usage of an application fluctuates dynamically on demand. An automatically scalable application is essential. Cloud computing permits resources to scale dynamically, and obsessed on the application-generated workload like CPU utilization and memory.

Cloud computing allows applications to dynamically scale resources on-demand. Any sort of application ought to have the potential to scale virtual resources up or down for achieving high performance. However achieving true elasticity, cost- efficacy and high performance are always challenging factors for cloud applications. Autoscaling, a feature of cloud computing has the competency to scale up or down the cloud resources according to step with demand.

Though Autoscaling provides excellent benefits, implementation may be a difficult task. Effective Autoscaling needs techniques to foresee future workload because the resources needed to handle the workload. The foremost prevalent approach for autoscaling is the Reactive Autoscaling strategy. It integrates resources when the workload exceeds whereas the other approach for autoscaling is to utilize a predictive approach that soothsays the workload of a cloud application by analyzing historical workloads.

*Author for correspondence

Predictive autoscaling is efficient when compared to reactive autoscaling which are popular for their simplicity and intuitive nature. This study fixates on the hybrid autoscaling approach where both predictive and reactive approaches are utilized. Predictive autoscaling is utilized for workload prognostication on private cloud platform and reactive autoscaling is withal used to obviate the negative impact of erroneous presages.

The time-series algorithms could be acclimated to find reiterating patterns in the input workload or to endeavor to conjecture future values. The Autoscaling quandary can be partitioned into two steps: First, soothsaying the future workload or resource utilization by utilizing time-series analysis. Second, predicated on the above prognosticated value, decide an opportune scaling action. Several approaches subsist for decision making process - set of predefined rules, solving as an optimization quandary for the resource allocation.

In the current literature, many diverse autoscaling techniques have been proposed to scale computational resources according to prognosticated workload. Several machine-learning techniques have been used to soothsay cloud workload, but cannot recollect the past events. The rest of this paper is organized as follows. Section 2 overviews cognate work in the area of automatic resource scaling. Section 3 describes the proposed mechanism for autoscaling. Observing this, Section 4 depicts the implementation and result of proposed techniques. Conclusively, Section 5 concludes.

2. Literature Survey

Today, cloud computing has evolved as far efficient, proactive and accepted technology. In[1-2], in their work described the use of virtualization technology. Auto Regressive Moving Average (ARMA) model, the Exponential Smoothing (ES) model, the Autoregressive model (AR), the Moving Average model (MA) and the Trend- Adjusted Exponential Smoothing (TAES) are utilized for time series forecasting. These methods gives better results for stationary data but fails to predict the future values incase if the data is non stationary.

In[3] conferred the cloud work prediction module for SaaS suppliers supported the Auto Regressive Integrated Moving Average (ARIMA) model. The impact of the achieved exactness evaluated in terms of resource utilization potency and QoS. Simulation results show that the model achieved a median exactness of 91 percent resulting in potency of resource utilization with lowest impact on the QoS.

In[4] applied straightforward RNN methodology for predicting the mainframe utilization accurately for brief time periods. It fails to fulfill if there's a explosive amendment in mainframe utilization. The higher than strategies for employment prediction is employed just for short-run prediction inclose if there's massive variation within the information points it fails to predict accurately.

In[5] bestowed associate car scaling system for heterogeneous forms of resources by shaping multiple levels of QoS needs. The model considers client needs also, in conjunction with work to perform scaling actions. In[6] planned repeated Neural Network with LSTM to autoscale virtual resources supported foretold values. The experiment pattern was done on CloudSim mistreatment NASA Log Traces work. Because the model depends on machine, results won't be satisfactory for long run practicality. compared to those time-series techniques, repeated Neural Network- Long Short Term Memory deep learning technique is employed to predict the long run work accurately just in case of any variance as a result of it will bear in mind the historical values. The planned approach is close to use efficient prophetical technique mistreatment history information to car scale the resources for cloud applications in camera cloud mistreatment open supply project OpenStack.

In[7] proposed Independent Recurrent Neural Network (IndRNN) method. It can be easily regulated to prevent the gradient exploding and vanishing problems while allowing the network to learn long-term dependencies. This method is used in replacement with Recurrent Neural Network- Long Short Term Memory (RNN-LSTM) method.

3. Proposed Mechanism for Autoscaling

In this section, the proposed mechanism includes the three modules data collection from OpenStack, workload prediction and Autoscaling in OpenStack. The overall use case of proposed methodology is shown in Figure 1.

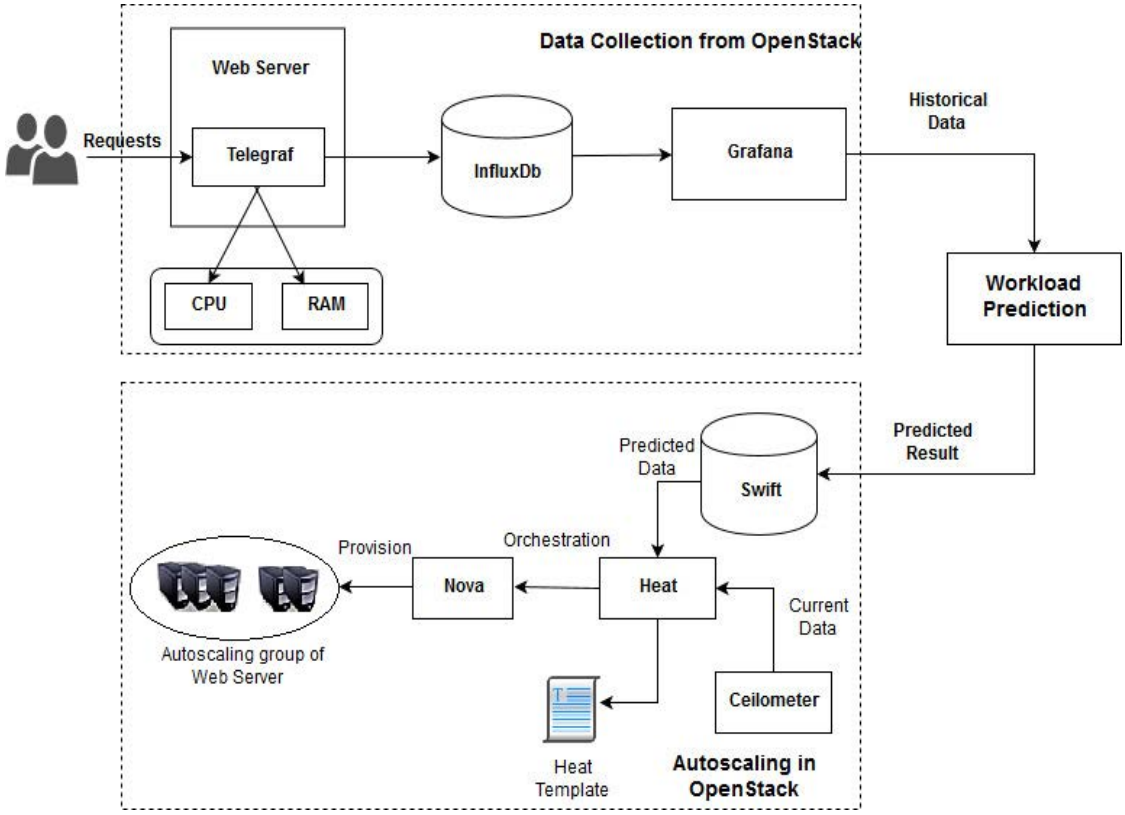

Figure 1. Overall use case of proposed methodology.

3.1 Data Collection

Telegraf, InfluxDB and Grafana are the three components utilized for amassing authentic time datasets. Each of these components is setup in separate VMs in OpenStack. When many users access a server at the same time metrics like CPU utilization, RAM utilization and the workload increases. This leads to server crash. These metrics can be perpetually monitored by utilizing a component called Telegraf. Telegraf is a metric assortment daemon which accumulates metrics from an immensely colossal array of inputs and indites them into InfluxDB. As it is plugin-driven for each assortment and output of data, it is facilely extendable. It's indited in Go, which implicatively insinuates that it is a compiled and standalone binary will be executed on any system. Hence there is no desideratum for external dependencies. InfluxDB is an open-source time series database, developed by InfluxData[8] and indited in Go language. InfluxDB is optimized for high availability storage and retrieval of time series data in operation monitoring, application evaluation and authentic-time analytics. InfluxDB is utilized to store the amassed data from Telegraf and withal query the OpenStack infrastructure metrics that are accumulated at the OpenStack infrastructure. Grafana is an open-source, dashboard and graph composer, which runs as a cloud application. It fortifies InfluxDB as backend. It is most generally utilized for visualizing time- series data for infrastructure and application analytics. The real time data accumulated from Telegraf are stored in InfluxDB. The data from the InfluxDB is visualized in graphical representation in Grafana dashboard. Grafana dashboard can export data in a required format like CSV, XSL or JSON. These components have been setup in OpenStack either in single VM or different VM's. The metrics are perpetually monitored predicated on the request that hits the server.

3.2 Workload Prediction

Workload prediction is done by analyzing the dataset collected from Grafana by using various predictive techniques. To predict the future workload, ARMA, the Exponential Smoothing (ES) model, the Autoregressive model (AR), the Moving Average model (MA) and the

Trend- Adjusted Exponential Smoothing (TAES), ARIMA Time-series model, Naïve bayes algorithm, RNN-LSTM and IndRNN are used and compared.

3.3 Autoscaling in OpenStack

The predicted results are stored in Swift component. It is a storage component in OpenStack which stores data with high availability. Ceilometer, Nova and Heat are also the components in OpenStack. Ceilometer is used for monitoring and metering the VMs. The current data from the cloud server is collected from ceilometer. The heat component provides orchestration service in OpenStack. A heat template is created which compares the current data monitored by ceilometer and predicted data stored in the Swift.

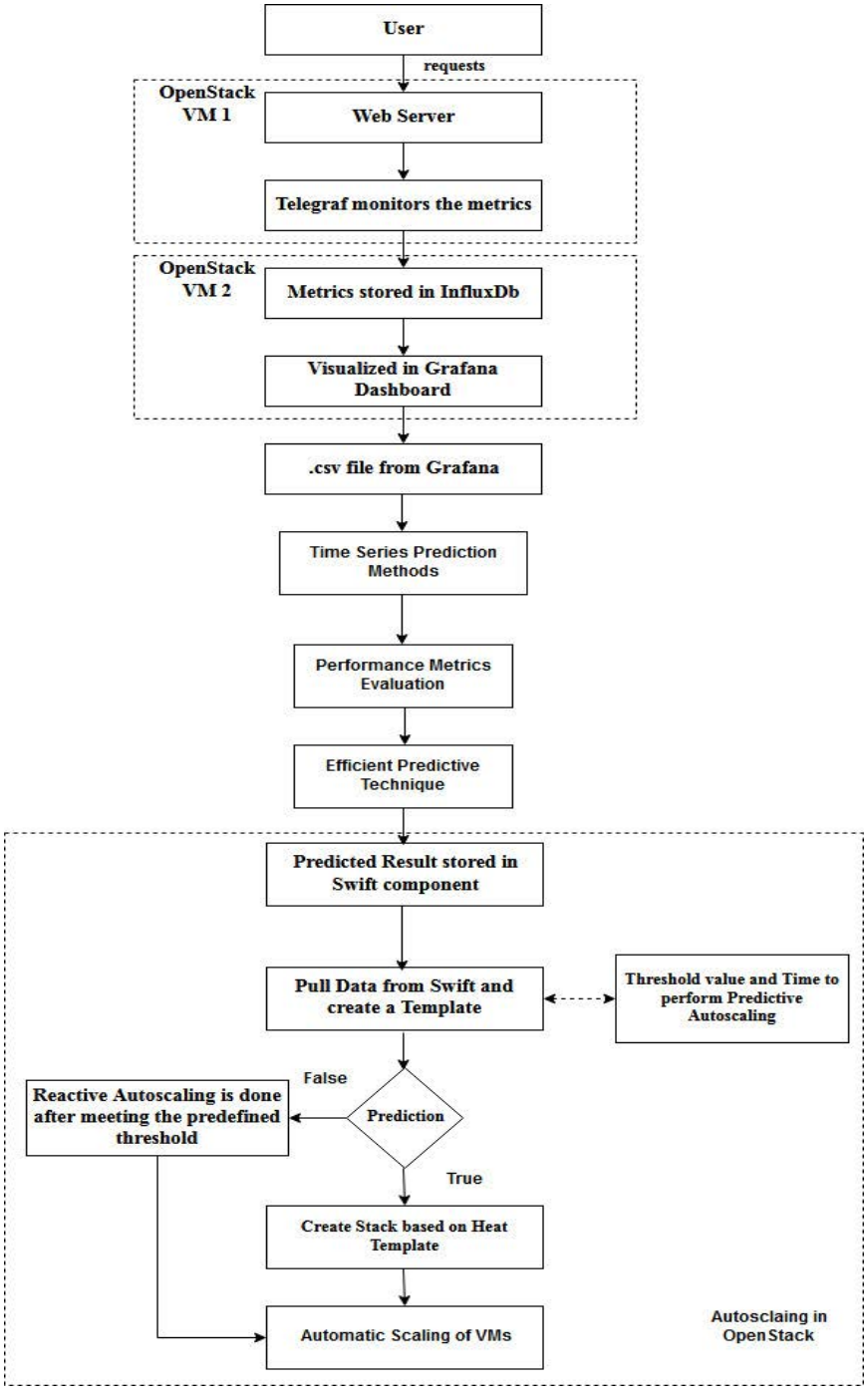

Figure 2. Workflow of proposed method.

Predicated on that, Nova component provisions the number of VMs, by Hybriding the reactive autoscaling and predictive autoscaling method.

4. Implementation and Result

The complete workflow of the proposed method is shown in Figure 2. When the number of requests from different users hit the same cloud server, CPU utilization gets magnified. A hybrid autoscaling is done on an OpenStack Mirantis version. OpenStack is setup by using five physical machines. Each machine has Intel® Core™ i3-2328M, 8 GB RAM and 64 bit OS. A virtual cluster that is OpenStack cluster is created consisting of several VMs. Apache Cloud Server and Telegraf are deployed in one VM. Telegraf monitors the number of requests in cloud server and stores it in InfluxDB. InfluxDb and Grafana are deployed in one VM. The Grafana dashboard visualizes the metrics in graphical illustration. The export choice in Grafana helps to gather the info in .csv format.The .csv file collected from Grafana contains 24 hours with four months of real-time data that is from 00:00:00 Dec 1, 2017 to 23:59:00 Mar 31, 2018 as shown in Figure 3.

The dataset collected from grafana dashboard is given as input file to time sries prediction methods. To evaluate the effectiveness of predictive techniques, Mean Absolute Error (MAE), Mean Squared Error (MSE) and Mean Absolute Percentage Error (MAPE) performance metrics are evaluated[9]. MAE is that the average over amassed dataset samples. It offers absolute variation between prediction and actual observation. Root Mean Square Error (RMSE) is a quadratic grading rule measuring the average magnitude of the error. It's the square root of the average of squared variations between prediction and actual observation.

MAE value should be less than or equal to RMSE. MAPE is used to measure the accuracy of prediction method. Typically, the smaller these values are, the better the prediction are[10]. The Table 1 shows the comparison of effectiveness of various time series prediction techniques. Based on the result obtained from evaluating the performance metrics for each prediction technique, IndRNN gives the minimum error rate compared to other prediction techniques.

Table 1. Comparison of effectiveness of time series prediction techniques

Predictive Techniques	MAE	RMSE	MAPE
IndRNN	0.007	0.0056	6.78%
RNN-LSTM	0.0128	0.0336	15.8%
ARIMA	0.46	1.245	32.6%
ARMA	2.678	2.964	46.7%
Exponential Smoothing Model	2.987	3.087	49.1%
Naive	3.87	4.33	52%

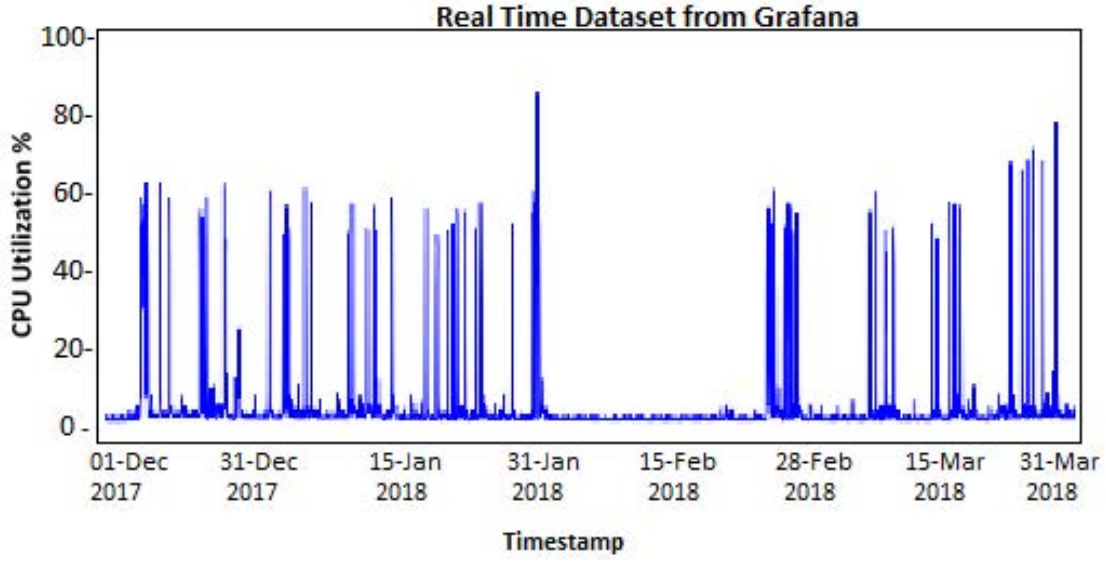

Figure 3. Dataset collection from grafana dashboard.

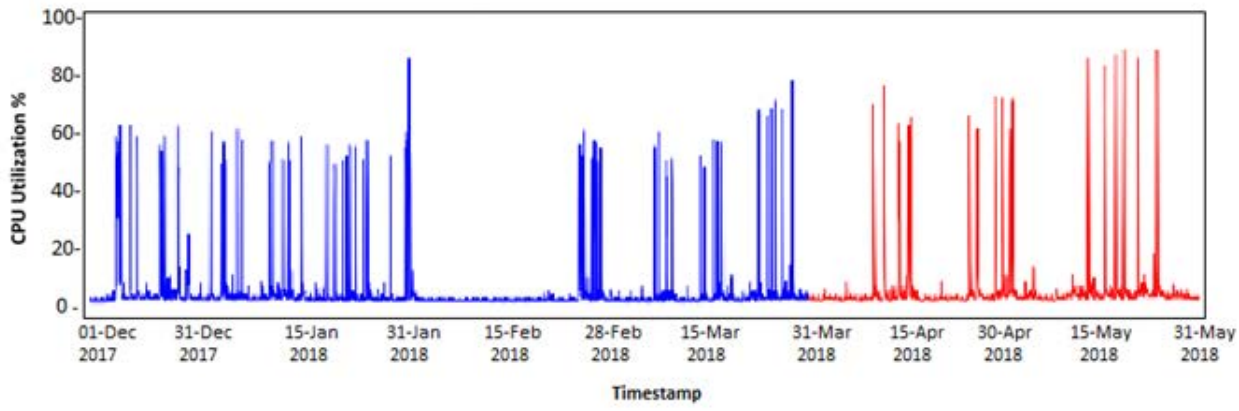

Figure 4. Future prediction using IndRNN.

In IndRNN, the dataset is preprocessed by using MinMaxScaler. Keras libraries are used to build the IndRNN. IndRNN is easily regulated to prevent the gradient exploding and vanishing problems which allows the network to learn long-term dependencies. In Keras library, predict () function is used to predict the future values. The Figure 4 shows the future predicted result.

The predicted result from IndRNN method is stored in Swift. The data is pulled from the Swift and a heat template is created. Current data of running cloud server is also collected from Ceilometer. Both the data specifies Timestamp and CPU Utilization features. In heat template, the predicted data is compared with current data. The threshold value is initialized to 50%. If CPU Utilization is above 50%, for example the CPU Utilization is above 50% from 09:49:00 to 10:49:00. Then automatically a VM is spinned for specified time period. Reactive scaling is also done when the CPU Utilization exceeds the threshold value ie 50%. It is used to prevent negative impact of wrong prediction and also to prevent server crash in case of overload.

5. Conclusion

In this study, the real time dataset collected from the components setup in OpenStack are given as an input to time series prediction techniques. The performance metrics is evaluated for each technique to choose the effective predictive technique that gives minimum error rate. Based on the evaluation, IndRNN deep learning technique gives the minimum error rate compared to other prediction models. So the predicted result from IndRNN technique is used to perform hybrid autoscaling approach for cloud applications in OpenStack. Future work is to perform prediction based autoscaling technique for heterogeneous application in cloud environment.

6. References

1. OpenStack Workload Reference Architecture: Web Applications. Date accessed: 2018. https://www.openstack.org/assets/software/mitaka/OpenStack-WorkloadRefArchWebApps-v7.pdf.
2. Chen CC, Chen SJ, Yin F, Wang WJ. Efficient Hybriding Auto-Scaling for OpenStack Platforms. IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity); 2015. p. 1079–85. https://doi.org/10.1109/SmartCity.2015.212.
3. Calheiros RN, Masoumi E, Ranjan R, Buyya R. Workload prediction using ARIMA model and its impact on cloud, IEEE Transactions on Cloud Computing. 2015; 3(4):449–58. https://doi.org/10.1109/TCC.2014.2350475.
4. Duggan M, Mason K, Duggan J, Howley E, Barrett E. Predicting Host CPU Utilization in Cloud Computing using Recurrent Neural Networks. 12th International Conference for Internet Technology and Secured Transactions (ICITST); 2017. p. 67–72. https://doi.org/10.23919/ICITST.2017.8356348.
5. Fernandez H, Pierre G, Kielmann T. Autoscaling Web Applications in Heterogeneous Cloud Infrastructures IEEE International Conference in Cloud Engineering; 2014. p. 195–204.
6. A Automatic Cloud Resource Scaling based on Long Short-Term Memory Recurrent Neural Network. Date accessed: 12.01.2017. https://arxiv.org/abs/1701.03295.
7. Independently Recurrent Neural Network (IndRNN): Building a Longer and Deeper RNN. Date accessed: 22.05.2018. https://arxiv.org/abs/1803.04831.

8. InfluxDB to Grafana: Visualizing Time Series Data in Real Time. Date accessed: 06.03.2017. https://www.codementor.io/ashish1dev/influxdb-to-grafana-visualizing-time-series-data-in-real-time-5hxhaq0uj.
9. Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)? –Arguments against avoiding RMSE in the literature. Date accessed: 30.06.2014. https://www.geosci-model-dev.net/7/1247/2014/.
10. MAE and RMSE - Which Metric is Better? Date accessed: 23/03/2016. https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d.