International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 2, Issue 2, 73-83, 2021
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V2I2P110

Original Article

Predictive Autoscaling for Kubernetes Using Multivariate Time-Series Forecasting

Chaithanya Kumar Reddy Nala Obannagari[1], Parameswara Reddy Nangi[2]
[1,2]Independent Researcher, USA.

***Abstract** - Applications built on Kubernetes clusters tend to be cloud-native and have to deal with highly dynamic loads whilst being constrained under a rigorous performance and cost-efficiency benchmark. Historical reactive autoscaling tools, including the Horizontal Pod Autoscaler (HPA) deliver a response that largely depends on the real-time resource consumption indicators and are not always able to react to abrupt workload changes. This is constraining, performance is affected, service-level agreement (SLA) is not met and resources are not used efficiently. A new trend of predictive autoscaling has come into picture as a format to rise above these issues by predicting future requirement of resources and preempting pooling of resources. The current paper has provided a detailed model of predictive autoscaling in Kubernetes systems based on multivariate time-series forecasting algorithms. The approach proposed will combine historical workload data, system level metrics, and application-specific data in order to come up with predictions of future resource requirements that are accurate. The system predicts the patterns of workload in advance with the help of machine-learning-based forecasting models, such as Long Short-Term Memory (LSTM) networks, Autoregressive Integrated Moving Average (ARIMA), and Multivariate Regression to adjust cluster resources accordingly. The framework is deployed as a planetary controller connected to the Kubernetes control plane, which would not create an issue with integration with the existing infrastructure. Large-scale experiments were done on benchmark workloads and real-world traces. The findings indicate that the suggested strategy can spend much less time to reply, make compliance with SLA significant, and increase the efficiency of resources as a whole as compared to traditional methods of reactive autoscaling. The results affirm that multivariate time-series forecasting offers a strong basis of intelligent and adaptable resource management in existence in cloud-native platforms. The research takes a step towards the development of autonomous cloud systems and offers some useful recommendations to implement the predictive autoscaling systems in the production setting.*

***Keywords** - Kubernetes, Predictive Autoscaling, Time-Series Forecasting, Cloud Computing, Resource Management, Lstm, Machine Learning, Container Orchestration, Sla Optimization.*

1. Introduction

1.1. Background

The quickening of microservices and container-based building has radically changed contemporary application developmental and implementation exercises as it has made it more modular, scalable, and easy to showcase. In this regard, Kubernetes has become the default standard platform to manage workloads implemented with containers due to the strong association to automated deployment, service discovery, fault tolerance, and horizontal scaling. [1-3] The Kubernetes is increasingly used by organizations of different industries in managing complex distributed systems and supporting high-availability applications. But, with larger and more complicated clusters, the management of resources becomes a challenge of critical importance in Kubernetes. In the large scale setting, workloads tend to have very dynamic and erratic behavior, making it challenging to assign computing resources in an optimum way that will not affect the performance or unnecessarily add to the operation cost. Autoscaling is also important in overcoming this dilemma by dynamically scaling the computing resources to changing workload requirements. The main mechanisms in Kubernetes, to support reactive autoscaling, are Horizontal Pod Autoscaler (or HPA) and Vertical Pod Autoscaler (or VPA), that may scale application replicas or resource constraints, based on real-time performance metrics, such as CPU utilization or memory utilization. These methods are quite easy to deploy and easily blend with the Kubernetes architecture. Nevertheless, they are based on the existing or previous system states and lack prediction features. Due to this fact, reactive autoscaling usually experiences lagging behind in the event of a sudden surge of work or a flash traffic, causing temporary shortages of resources and the deterioration of performance. These delays can lead to longer response times, lower throughput and service-level agreement violations. In turn, reactive autoscaling is not able to succeed in highly volatile workloads, which underscores the necessity of more intelligent and active approaches to managing the resources and in the stabilized and moderately dynamic environment.

1.2. Emergence of Predictive Autoscaling

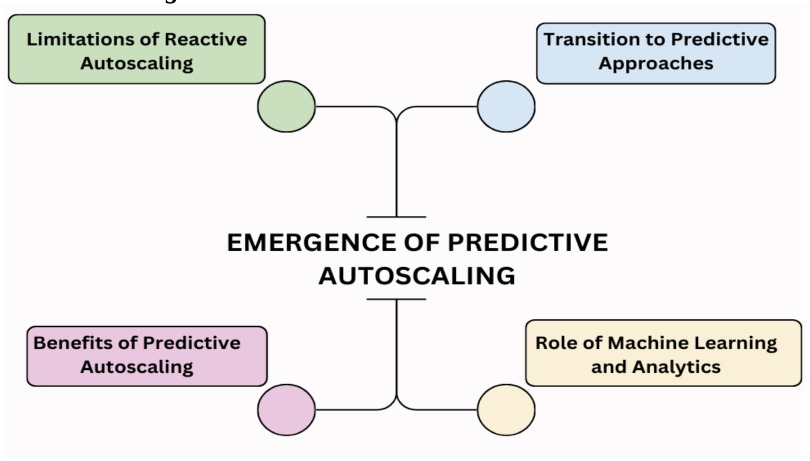
Limitations of Reactive Autoscaling
Transition to Predictive Approaches
EMERGENCE OF PREDICTIVE AUTOSCALING
Benefits of Predictive Autoscaling
Role of Machine Learning and Analytics

Fig 1: Emergence of Predictive Autoscaling

1.2.1. Limitations of Reactive Autoscaling

The traditional reactive autoscaling systems (like Kubernetes HPA and VPA) increasing or decreasing the resources according to the current or historical metrics of the system performance. Though they are easy to use and commonly used, these means react only when the adjustment of workload has already affected the performance of systems. Reactive strategies can be ineffective in timely delivering resources when there is suddenly a high increase in demand or unexpected changes in demand. The effect of this delay may be response-time increase, service interruptions, and breach of service-level contracts. Besides, the frequent scaling of actions caused by the short-term fluctuations of metrics can cause the instability of the system and the waste of the resources.

1.2.2. Transition to Predictive Approaches

Predictive autoscaling has become one of the advanced strategies of managing resources to overcome the limitations of reactive approaches. Predictive autoscaling uses past history as well as real-time monitoring data to predict the future workload trends. The system can maintain performance by predicting future demand to either allocate or release resources beforehand hence avoiding performance breakdown. This move signifies the change of making decisions based on rules to the following data-driven and intelligent automation. Predictive strategies allow cloud platforms to be more dynamic and sensitive to shifts in the workloads.

1.2.3. Role of Machine Learning and Analytics

Predictive autoscaling is made possible with the roles of machine learning and advanced analytics. The patterns and correlations and workload data are identified using the statistical models which include ARIMA, machine learning methods which include regression, support vectors machine, and deep learning networks. Specifically, the deep models, including LSTM and GRU models have shown their powerful potential in capturing the relationship over time and nonlinear dynamics. These methods enable predictive systems to adapt to previous performances and maintain their forecasting quality at a high standard thus making them applicable to dynamic and complicated cloud-based environments.

1.2.4. Benefits of Predictive Autoscaling

Such implementation of predictive autoscaling has a number of important benefits over these more conventional methods of reactive implementation. Predictive systems can save time in response latency, minimize service disruption, and increase user experience by provisioning resources in advance to them. Higher accuracy in prediction causes better use of resources and also less operational costs are caused by preventing both under-provisioning and over-provisioning. Moreover, predictive autoscaling assists in the long-term planning of capacity and allows utilizing infrastructure resources much more sustainably. With the ever-increasing size and complexity of cloud applications, predictive autoscaling has become a new part of smart and autonomous cloud management controls.

1.3. Limitations of Reactive Autoscaling

Reactive as well as autoscaling systems mainly work on the principle of preset threshold-based policies object by which actions of scaling are initiated once performance metrics under observation leading to CPU utilization or memory consumption have passed or have gone beyond a defined limit. [4,5] Although this strategy is easy to execute and popular among the cloud platforms, it has various points of limitation that limit its application to complex environment where it is active and large-scale. Delay in scaling decisions is also one of the biggest weaknesses of reactive autoscaling. Because these systems do not provide resources until a specified threshold is reached in terms of utilization of resources, a lag will always exist between when an increased workload starts and when additional resources are provided. Some performance degradation, in as much as response time and throughput, are possible during this period, causing poor user experience and possibly breaching the service-level agreements. The implications of reactive mechanisms as well as their failure to manage sudden and unforeseen bursts of workload is also another major weakness. Flash crowd, promotion, and viral content are some situations in which traffic can skyrocket in a very brief time. Often reactive systems can not respond to such rapid variation in a timely manner, responding to such rapid changes has to be based on long-term metric violation across multiple monitoring intervals. As a result, the system could face some overload, which would lead to the disruption of the services and the failure of the apps. Moreover, reactive autoscalers can be also sluggish in disMessage resources once the workload has declined thus causing longer durations of over-provisioning. nReactive autoscaling systems also tend to have oscillating resource allocations, which is often called thrashing. Repetitive scaling up and scaling down activities that occur in a short period may be as a result of frequent changes of workload parameters that are adjacent to threshold figures. These vibrations add overhead to the system, and consume more computer resources as well as could have adverse effects to application performance. They also make capacity planning harder, and ultimately less stable. Moreover, the use of reactive autoscaling usually results in the inefficient use of costs. As a means of reducing the risk of under-provisioning, administrators can set conservative values of the threshold in ways that are biased towards over-provisioning. Although this method enhances reliability, it causes waste of resources and unwarranted costs of operation. On the other hand aggressive thresholds lower prices at the expense of reduced performance. Consequently, reactive autoscaling cannot find an optimal balance between the service quality and the cost-effectiveness. These shortcomings underscore the importance of smarter, more proactive approaches to autoscaling that will be able to match the current features of cloud workloads.

2. Literature Survey

2.1. Traditional Autoscaling Techniques

The initial systems of cloud autoscaling were mainly based on fixed provisioning and rule based scaling to control the computational resources. [6-9] Services like Amazon EC2 Auto Scaling and Google app engine put and took away resources based on pre-established threshold policies, by which system metrics exceeded or fell below a particular threshold. Although these methods were easy to design and implement, they were not intelligent and capable of accommodating dynamic workload statistics. Consequently, they tended to cause under-provisioning in situations of sudden demand increase and over-provisioning when the demand was low. Kubernetes Horizontal Pod Autoscaler (HPA) is a similar paradigm extension that maintains metrics on the use of resources like CPU and memory and continuously checks. Even though HPA is reactive and more responsive, it is still essentially reactive and prone to changes in short-term metrics, which are likely to trigger frequent and unwarranted scaling responses.

2.2. Time-Series Forecasting Models

Coastal prediction has been widely applied in forecasting the future work load trends in cloud computing. Classical statistical models (Auto regression- AR, moving average- MA, autoregressive integrated moving average- ARIMA and seasonal autoregressive integrated moving average- SARIMA) are mathematical models used to identify the temporal and seasonality relationships in the past through mathematical programming. The models are appreciated due to their mathematical nature and are interpretable so that the system administrators can get insight into the workload behavior. Nevertheless, they also make the assumption of linear correlation and stationary data and thus are not very effective in cloud systems that are highly dynamic and complex. Moreover, they perform poorly in situations where workload changes abruptly, on an intermittent nature, or where the relationships are nonlinear as seen in real-life situations.

2.3. Machine Learning-Based Approaches

The methods based on machine learning have become more popular with the growing complexity of cloud workloads thanks to autoscaling and the workload prediction. Support Vector Regression (SVR) and Random Forests (RF) are among the techniques that have been widely embraced because they are able to capture nonlinear relationship and a high dimensional data. These techniques have the ability to gain knowledge of complicated patterns based on previous measurements and evolve with a new environment. Its usefulness, however, highly relies on selectiveness of features and engineering, which cannot be done without domain experience. Over the past few years, deep learning models specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks have demonstrated better performance with regard to long-term dependencies and temporal dynamics modelling. Although they are very accurate in prediction, these models are intensive in computation and training needs a very large dataset which might not be effective when resources are limited.

2.4. Multivariate Forecasting in Cloud Systems

Multivariate forecasting techniques take a combination of several performance measures at the same time to enhance precision in predicting the result. Within cloud environments, a workload behavior is caused by a number of interconnected measures, such as CPU usage, memory usage, network bandwidth, disk I/O, and request arrival rates. Multivariate models have the capacity of representing system dynamics and relations of resources by combining these variables. Research has shown that these models are more efficient than univariate ones that apply single metrics particularly in heterogeneous and multidimensional cloud environments. Multivariate methods allow making scaling choices more knowledgeably as they allow one to see the entire picture of the system performance. Nonetheless, they also bring increased calculational challenges and need the consideration of correlated and redundant qualities.

2.5. Comparative Summary

A comparison between the existing autoscaling and workload prediction methods presents the strengths and limitations to them. Threshold-based reactive autoscaling techniques are easy to deploy and involve minimum computation resources, however, they do not respond quickly and cannot be predictive. ARIMA and SARIMA are statistical time-series models with interpretability and theory grounding however having problems with nonlinear and non-stationary data. Methods based on machine learning, such as SVR or Random Forests, allow more flexibility and adaptability but require more feature engineering and parameter tuning. LSTM and GRU are deep learning models that are characterized by high forecasting accuracy obtained based on learning the complex temporal patterns; but it requires a large number of computational resources and a vast amount of training data. Hence, a choice of a suitable approach has to be a trade-off between the accuracy of prediction, cost of computation, and requirements of a system.

3. Methodology

3.1. System Architecture

The similarity of the proposed predictive autoscaling framework consists [10-12] of four significant elements that collaborate to provide the ability to manage resources on the clouds intelligently and proactively.

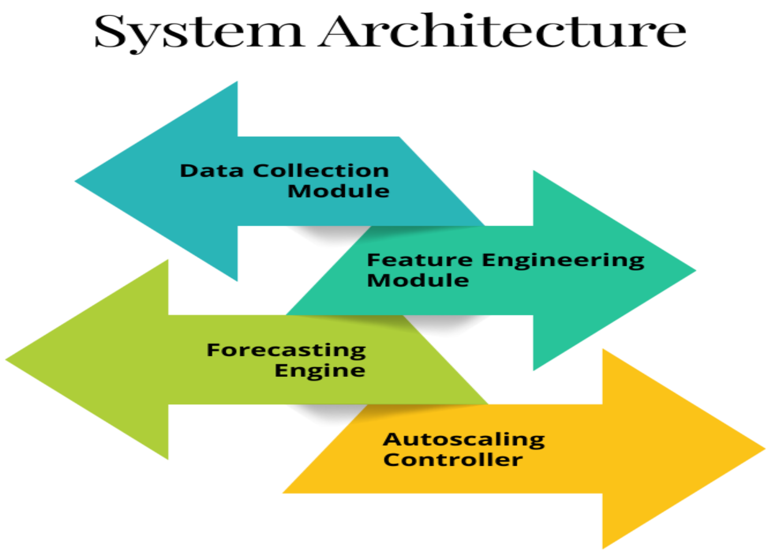

Fig 2: System Architecture

3.1.1. Data Collection Module

The Data Collection Module will have a role of nonstop monitoring and collecting system performance metrics of the cloud infrastructure. It gathers real-time and past information like CPU making use, memory usage, disk communications, network traffic, and rate of incoming requests of the virtual machines, containers, and application services. Such data is received with the help of monitoring tools, system logs, and cloud-native services. The information gathered is then taken into a central repository where additional processing will be done. The collection of data should be accurate and timely since it is the basis of accurate predictions of workload and autoscaling.

3.1.2. Feature Engineering Module

The Feature Engineering Module works on the raw data received at the collection layer and converts it into meaningful data that is used as input features to the forecasting models. This module cleans data, normalizes data and/or fills in missing values to enhance the quality of data. It also retrieves statistical and temporal characteristics of the data including moving averages, trend, seasonality signals and the lag variables. These features are chosen and are built by this module to increase the learning ability of the prediction models and minimize noise that subsequently increases the accuracy of the forecasting of the model.

3.1.3. Forecasting Engine

Forecasting Engine dominates the predictive autoscaling structure. It applies machine learning and deep learning prototypes to measure previous and current data and produce future predictions pertaining to workloads. Models ARIMA, SVR, LSTM, or GRU can be used to make predictions of resource demand in both the short-term and long-term using different models depending on the requirements of the system. Periodically the engine will be retrained to take into account the variation of workload patterns and systems behavior. Incremental scaling that is preemptively implemented during accurate forecasting will help avoid the pollution of the performance and wastage of resources.

3.1.4. Autoscaling Controller

The workload predictions are then converted into tangible resource management executions by the Autoscaling Controller. Depending on the results of the forecasting engine, it will be able to decide the number of virtual machines, containers, or pods needed to sustain the desired performance. This component utilizes the pre-established policies and optimization techniques to optimize the quality of service and operation cost. It communicates with cloud orchestration environments e.g, Kubernetes or OpenStack, to auto-spin resources or auto-spin them down. The proactive approach to the situation instead of the reactive one allows the controller to make the systems stable, shorter the response time, and enhance the overall use of the resources.

3.2. Data Collection and Preprocessing

A predictive autoscaling system requires efficient data collection and preprocessing to develop an accurate and reliable system. [13-15] The system metrics in the proposed framework are collected with help of Prometheus and Kubernetes Metrics Server that enables real-time and historical monitoring of a containerized environment. These tools allow one to monitor resource consumption and performance of application in all nodes and pods in the cluster. Data obtained are stored as time-series database, which could be easily accessed to help in analysis, training a model, and prediction. Before advancing any further processing, proper preprocessing is used to eliminate inconsistencies, eliminate noise and maintain the quality of data.

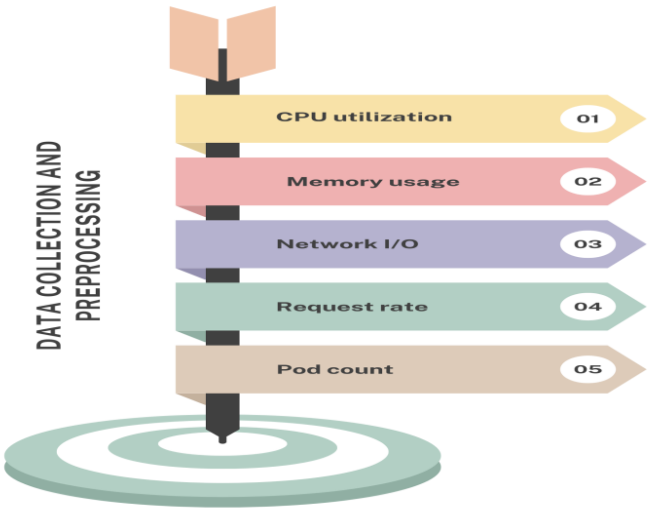

Fig 3: Data Collection and Preprocessing

3.2.1. CPU Utilization

CPU utilization is a form of percentage of the processor resources that are used to run the applications and system processes. It is a serious gauge of calculative load and it ranks the requirement of the services used directly. A high CPU usage is usually an indicator of high user activity or calculation intensive activities whereas low is an indication of unused resources. Within this framework, usage data of the CPU is sampled with some constant interval and normalized in order to consider the differences between the nodes and pods. This preprocessing is necessary to be consistent and allow fair comparison between the elements of the system.

3.2.2. Memory Usage

Memory usage can be defined as the use of main memory by applications and containers that are executing. A lack of memory allocation can cause reduced performance, high latency or even crashes in applications as a result of out-of-memory errors. Thus, it is important to regulate memory consumption in order to ensure the stability of systems. The obtained memory measures are filtered with transient spikes, and it is averaged with moving averages. This assists in the capturing of the usage over time and enhances the strength of workload predictions.

3.2.3. Network I/O

Network I/O is used to measure data flow between services, nodes and external clients. It indicates communication overhead and requests intensity of the user in the cloud environment. Increased network traffic is typically linked with applications that are data-intensive, distributed processing, and high loads to requests. During preprocessing, network input and output rates are averaged across some time window in order to eliminate short-term fluctuations. This consolidation assists in the determination of significant traffic pattern and enhances the accuracy of the forecasting model.

3.2.4. Request Rate

Request rate is a measure of the number user or client requests that are sent to this application per unit time. It is one of the closest predictors of work load intensity and user demand. Subsequent surges in request rates can create resource contention and performance bottlenecks in the event they are not addressed appropriately. In the proposed system, the data on the request rate will be derived out of the application logs and monitoring endpoints. These are then synchronized on other system measures and standardized to give it a temporal coincidence with which to do the multivariate analysis.

3.2.5. Pod Count

The number of active pods that are operational in the Kubernetes cluster at any particular moment, is known as the pod count. It indicates the existing scaling level of the system and the number of instances of the application that can be used to work with the workloads. Counting of pods is relevant towards relation of resources allocation and performance. In the preprocessing stage, correlation between pod counts information, and workload metrics are performed to relate scaling effectiveness. This data is learned to predict forecasting models and optimize future scale choices.

3.3. Forecasting Models

The suggested predictive autoscaling system uses the various forecasting models to represent the various workload features. These models have been chosen with regard to the capacity to deal with the linear trends, nonlinear patterns, and multivariate relationships of cloud workload data. In particular, predicting accurate and reliable workload is performed with the help of ARIMA, LSTM, and Multivariate Regression models.

3.3.1. Arima Model

ARIMA is an effective statistical technique with prior utilization in time-series prediction. It is expressed as ARIMA(p, d, q) whereby, p refers to the number of past observations to be used in making a prediction, d refers to how many times the data have to be differentiated in order to bring the data to a stationary level and finally, q is the number of past error terms to be included in the model. In this model, current value of the time series is represented as a constant value, weighted sum of the past values and weighted sum of the past prediction errors in addition to a random noise term. The autoregressive component takes into consideration the driving factor of the past values whereas the moving average component takes into consideration the driving factor of the past forecasting error. A mixture of these elements makes ARIMA suitable in modeling linear time-dependent relationships in the workload data.

3.3.2. LSTM Network

Long Short-Term Memory (LSTM) networks are a kind of recurring neural network that is capable of learning the long-term connection in a sequential data. They are memory cells and three primary gates namely the input gate, the forget gate and the output gate. The internal memory of the network is the cell state, which is changed during every time step. This update is carried out by choosing the extent of past information one wishes to remember with the forget gate and the extent of additional information one wishes to add with incoming information to it using the input gate. The previous cell state combined with the new candidate memory to create the updated state is then used. The mechanism enables LSTM networks to filter out redundant data and retain relevant patterns successfully. Consequently, LSTMs have great success in the ability to record nonlinear and long-term workload variations in cloud systems.

3.3.3. Multivariate Regression

Multivariate regression or, multiple linear regression, is a statistical method of modeling a phenomenon of one dependent variable and more than one independent variable. Under this method the forecasted value is represented as a constant value plus weighted values of the input features in the approach with a value of error that shows the variation which is not explained. All the coefficients represent the extent of effect of a particular input variable to the predicted workload. As an example, input variables can be measurements like CPU utilization, memory usage, and network traffic that can be used to predict the future demand of the resources. This model is also easy, interpretable, and computationally efficient, hence is applicable in a situation where fast predictions are needed. Its performance, however, can be affected in a negative manner whenever the relationship between variables is not linear.

3.4. Autoscaling Decision Engine

The Autoscaling Decision Engine is at the core of converting the workload needs projected into tangible resource allocation steps. [16-18] After the forecasting models have estimated the resource requirements in the future, the decision engine makes the right call to decide the number of pod replicas to be used to deal with the expected workload effectively. It is done in accordance with a mathematical correlation that translates the forecasted demand into the units of computing, which can be scaled. The number of required pods in this formulation is determined by dividing the forecasted work load requirement by the single processing capacity of a single pod and rounding the value to the nearest whole integer. This rounding off operation permits that there would always be adequate resources that can be supplied to satisfy service demands even when the calculated figure is not an integer. In regular lingo, the formula represents the fact that the system estimates the degree of future load of computations in the near future. This projected demand is the sum of total processing, memory or service capacity that is going to be needed to get a good performance. Information on the average capacity of one pod is also stored in the system and this is used to show the amount of workload that one pod is capable of carrying under normal operating conditions. The engine can calculate the number of pods theoretically needed to handle the workload by dividing the forecast of the demand by this per- pod capacity. This is because we cannot deploy a fraction of a pod and the value needs to be rounded up to ensure that there is sufficient supply. This is an preventive measure that allows the system to distribute resources even before it starts deteriorating. The decision engine proposed works in contrast to the conventional strategies of reactive autoscaling which only becomes effective when resources are already saturated. Instead, the strategy suggests addressing the needs before they arise, ensuring that the infrastructure is already prepared. Due to this, it minimizes response time, eliminates service level agreement (SLA) breaches, as well as, enhances overall system reliability. Moreover, through proper estimation of the demystification level of pod, the engine will not engage in over-provisioning, thus it will reduce operation costs and energy consumption. Moreover, the Autoscaling Decision Engine includes ready-made policies and safety guard so that the system behavior is stable. An example is maximum and minimum pod limits to avoid extreme scaling behaviour and cooldown periods are imposed to avoid rapid oscillations. These are the mechanisms that ensure inequality between responsiveness and stability. Altogether, the autoscaling decision engine allows managing the resources efficiently, reliably and cost-effectively in any cloud-native surroundings with systematic translations of workload predictions into optimal scaling decisions.

3.5. Predictive Autoscaling Workflow

The predictive autoscaling workflow specifies a series of processes by which the system tracks workloads, process data and resource adjustments are proactively operated. This workflow will guarantee that scaling decisions rely on correct information and dependable projections, which are suitable to establish effective and consistent management of resources in a cloud setup.

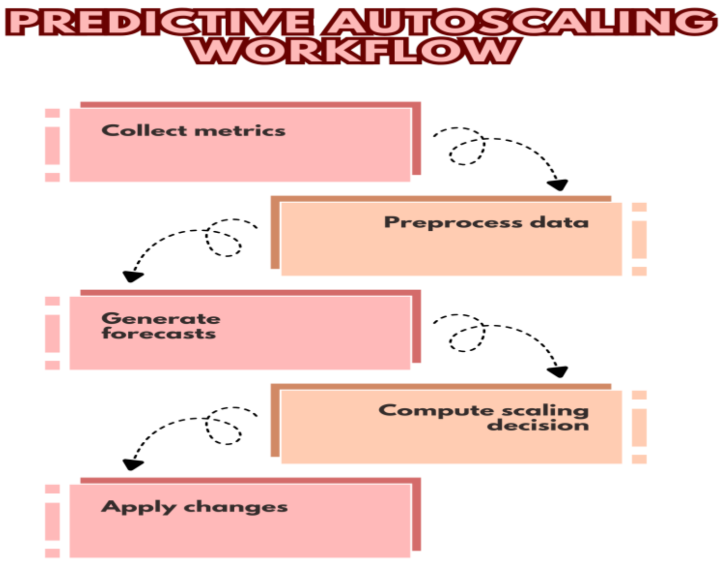

Fig 4: Predictive Autoscaling Workflow

3.5.1. Collect Metrics

The initial activity in the labor process is the ongoing gathering of the system and application performance measurements. Prometheus and Kubernetes Metrics Server are monitoring tools, which receive real-time data connected informing about CPU utilization, memory consumption, network traffic, request rates, and pod status. These measures are taken periodically to realize workload fluctuations and behavior of the system with a period of time. Regular and overall measure gathering forms the basis of all the further processing and forecasting operations.

3.5.2. Preprocess Data

The data collection stage is followed by the preprocessing phase whereby the raw metrics are delivered to the processing stage in order to enhance their usefulness. The step entails the cleaning of the data by eliminating noise, outliers, and missing records. After interpolation, statistical methods are used to deal with missing values whereas normalization and scaling are used to make all metrics similar. Temporal alignment also is carried out to align streams of data of different sources. Good preprocessing will increase reliability and accuracy of forecasting models.

3.5.3. Generate Forecasts

During the forecasting phase, one enters the preprocessed data into the choice of prediction models, which include ARIMA, LSTM or multivariate regression. These models evaluate the historical trends and present trends in order to project the future demand of the work load. The forecasting engine will give short term and long term forecasts depending on system requirements. The retraining of these models is carried out on a regular basis in order to cope with how the resulting workload patterns change and still retain its prediction capacity. Proper forecasting makes it possible to scale up and not scale down.

3.5.4. Compute Scaling Decision

After the generation of workload predictions, the system then calculates the correct scaling action. Estimated demand is plotted against the number of replicas of pods required based on pre-existing mathematical models and policy requirements. Some of the factors which are taken into account include minimum and maximum pod limits, service-level objectives and availability of resources during this process. The decision engine makes sure that the chosen scaling action is based on the balance between performance needs and cost-effectiveness and stability of the system.

3.5.5. Apply Changes

The last part of the working process is the execution of the calculated scaling decision in the cloud infrastructure. Autoscaling controller communicates with the orchestrating platforms like Kubernetes and creates or destroys pods as needed. There is gradual implementation of scaling to prevent disruption and instability of the services. System feedback following deployment is constantly followed to ascertain the efficiency of the scaling operation. This feedback-like process allows the optimization of the continuous process and management of the available resources.

4. Results and Discussion

4.1. Experimental Setup

The experimental assessment of the proposed predictive autoscaling framework was performed in a Kubernetes-based cloud system that was supposed to model the real-everyday operational environment. The testbed was a cluster comprising of ten worker nodes and a separate master node to orchestrate and control. All the worker nodes were equipped with identical hardware specifications featuring multi-core processors, adequate memory, and high-speed network connectivity between them, so that their performance in the cluster is uniform. Docker was used to deploy containerized applications, which were governed by Kubernetes to allow the allocation of resources dynamically and automated making of workloads distribution. This system was designed to offer a scalable and adjustable platform on which the behaviour of the autoscaling system could be studied in both different load conditions. In order to create controlled and repeatable workloads, industry standard benchmarking tools (Apache JMeter and Locust) were using. Apache JMeter was operated to replicate the behavior of a high number of users simultaneously and estimate response times, throughput and error rates of applications. It allowed developing a variety of test cases such as steady-state loads, rapid surge in traffic and incremental workload assignments. A load testing tool based on the patterns of realistic user behavior and dynamically regulating the request rate, called Locust, was used to simulate realistic user behavior. All these tools combined enabled the analysis of the system performance in terms of synthetic and semi-realistic workloads thoroughly. Besides synthetic benchmarks, workload traces provided by real-life settings were also used to ensure the practical usefulness of the proposed approach. The traces were gathered using publicly available datasets, including cloud usage logs and web traffic histories, which are reflections of actual access patterns of the users and resource consumption patterns. The experiments comprised the traits of complex workloads, such as periodic variation, bursty traffic, and long-term trends through the use of real-world data. Any and all dataset was preprocessed to fit in the forecasting models as well as the experimental environment. In addition, several experimental executions were chosen with the same conditions applied to achieve statistical reliability and knowledge reproductiveness. Response time, resource utilization, scaling latency and cost efficiency performance metric parameters were recorded and analyzed. This elaborate experimental design allowed a detailed and fair analysis of the postulated predictive autoscaling model under both controlled and realistic conditions in a cloud.

4.2. Performance Evaluation

Four major measures, which include Mean Absolute Percentage error (MAPE)., Service Level Agreement (SLA) Violation Rate, CPU Utilization, and Cost Reduction, were used to assess the performance of the suggested predictive autoscaling framework. These measures were chosen in order to measure the accuracy of prediction, reliability of the service, resource economy, and financial gains. The proposed methodology was contrasted with the default Kubernetes Horizontal Pod Autoscaler (HPA) to show how the predictive approaches, such as ARIMA-based, regression-based, and LSTM-based models, have made improvements.

Table 1: Performance Evaluation

Method	MAPE (%)	SLA Violation (%)	CPU Utilization (%)	Cost Reduction (%)
HPA	18.6	9.4	62.3	0.0
ARIMA-Based	12.2	6.1	71.5	8.4
Regression	10.8	5.3	74.2	11.6
LSTM-Based	6.4	2.1	82.7	19.3

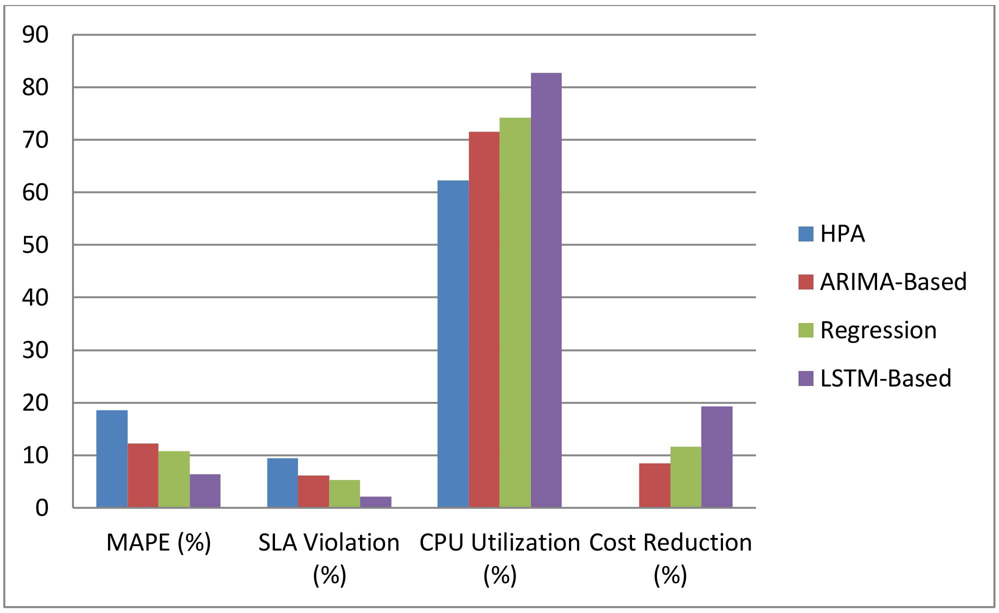

Fig 5: Graph representing Performance Evaluation

4.2.1. Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) is a standard that evaluates the quality of workload prediction by retrospecting the mean difference between predicted demand and actual demand in percentages. When MAPE is lower, forecasting performance is higher. The results of the experiments showed that the HPA reactive approach had the worst MAPE of 18.6% because it could not predict future work loads. ARIMA model also enhanced the accuracy of predictions through a MAPE of 12.2, which proves the usefulness of statistical time-series modeling. The regression based model minimized the error further to 10.8% by using several system measurements. The LSTM model presented the lowest MAPE of 6.4% which illustrates that the model has a higher ability to capture complex temporal and nonlinear patterns. The findings verify that the deep learning models are more precise in workload prediction than customary tactics.

4.2.2. SLA Violation Rate

The rate of SLA violation is the percentage of time when the system did not satisfy the performance requirements established in the agreement i.e. performance thresholds like response time and availability. Reduced rates of SLA violations signify good reliability in services. The reactive HPA strategy had a moderate violation of 9.4% which was primarily attributed to low scaling response to sudden workload increase. ARIMA-based approach had minimized violations to 6.1% as it allowed the partial proactive scaling. This regression based model also enhanced the reliability of the services up to 5.3% violation. LSTM-based method had the lowest violation rate at 2.1, which indicates that it would be effective in owing to its capabilities to avoid the degradation of performance by making sound and prompt scaling decisions.

4.2.3. CPU Utilization

CPU utilization is a measure of the efficiency of the available computationOn in using the computing resources. An increase in the values of utilization means that resources are managed well as long as the performance of the systems does not reduce. The reactive HPA strategy recorded an average CPU usage of 62.3, implying that the CPU was underutilized when the demand was low. The ARIMA based model enhanced the use of resources to 71.5% through closer alignment of the resources allocation to the anticipated demand. The regression based model further improved the use to 74.2 one that showed superior workload-resource fit. The LSTM-oriented model reached the maximum level of utilization equal to 82.7 that means the model is able to use the resources in the most efficient way without adversely affecting the performance of the system. This enhancement can testify to the benefit of predictive autoscaling to prevent idle resources.

4.2.4. Cost Reduction

Cost reduction is a percentage measure of operation reduction realized due to efficient provisioning of resources, in comparison to the baseline approach of HPA. Reactive HPA model failed to accrue any cost savings because often it over-provisioned resources to meet peak demands leading to unnecessary resource expenditure. The ARIMA-based method had succeeded in reducing the cost by a margin of 8.4 per cent with the ARIMA technique enhancing the accuracy of scaling. The model that was regression-based brought the savings further to 11.6% by means of enhanced use of system measures. The highest cost minimization rate of 19.3 was in LSTM-based approach because it was able to reduce not only the over-provisioning but also the under-provisioning. These findings indicate that proper workload prediction results in a high level of economic gains in cloud set ups.

4.3. Discussion

As the outcome of the experiment shows, the proposed predictive autoscaling framework is rather effective, especially employing the LSTM-based predictive forecasting framework. Of all the analyzed methods, the autoscaler based on the LSTM model has obtained the smallest prediction error and minimum SLA violation rate, which proves that it is a more competent method in forecasting workload changes and proactively distributing resources. The LSTM model was able to forecast the resources before the demand spikes by properly identifying long-term dependencies and nonlinearities in the workload data. This dynamic preemptive provisioning greatly minimized the turnaround time when it is maxing out and had the effect of maintaining consistent application performance and better user experience. Besides, timely scaling operations helped to avoid overloading of the service and reduced the likelihood of system overload and thus improved the overall system reliability. The other significant conclusion drawn during the experiments is the significant decrease in the wastage of resources by predictive autoscaling. As compared to reactive domain where the model is likely to over-serve the unexpected spikes in work demands, the LSTM-based model was closely aligned with the actual demands and the allocation of resources. In this exact fit, more CPU was used, and idle space was reduced thus significant cost savings were achieved. The increased efficiency of the resources also lead to a less energy usage and enhanced cloud infrastructure sustainability. These results emphasize the convenience of applying deep learning to the cloud resource management systems. Other statistical models (ARIMA) showed a moderate improvement in performance compared to default reactive HPA mechanism. These models were also capable of giving limited predictive and scaling delays to some degree by taking into account the past workload trends. They were however limited in performance in highly dynamic and unpredictable environments by their dependence on linear assumptions and stationary data. ARIMA-based methods were unable to adapt to the sudden shifts in workload and the erratic patterns of traffic, which increased the number of errors in the predictions and the cases of the SLA violation. Correspondingly, regression-based models performed reasonably well, using a number of system measures, but their linear form cannot be used to represent complex interaction between variables. However, the LSTM-based system generally adjusted to the changing patterns of workloads and even performed consistently in various situations.

5. Conclusion

In this paper, a fully operational predictive autoscaling framework of Kubernetes environments using time-series forecasting methods with multivariates was introduced. The proposed system, incorporating both a statistical and machine learning-based model, such as an ARIMA, multivariate regression, and a LSTM network, is efficient in predicting the changes in the workload in the future and allows proactive resource provisioning. In contrast to the classical reactive means of autoscaling, which enables resources reallocation only after they become saturated, the proposed structure relies on both historical and real-time performance metrics which may be used to forecast future demand and prior adjustments to resource distribution. This active mode of action increases a lot of responsiveness of the system, slows down services and increases the reliability of the whole application. The results of the experiment effectively show that predictive autoscaling is more successful in relation to its advantages in prediction accuracy, SLA compliance, resource use and also cost efficiency in its operation as compared to the conventional threshold-based technologies. Specifically, the LSTM-based model was more successful in terms of finding deep temporal correlations, nonlinear workload profiles, and thus the model is very appropriate to the contemporary cloud-native apps.

Also, the combination of multivariate monitoring data, including CPU load, memory usage, network traffic, the rate of requests, and the number of pods, made it possible to learn more about the system behavior. The framework could produce more credible predictions and make a decision on scaling by taking into account a variety of correlated parameters. This holistic approach reduced over-provisioning and under-provisioning and hence there was wastage of resources and the efficiency of energy was enhanced. The high flexibility of the proposed system in form of its modular architecture also supports the ease of deployment, maintenance and extension to the already existing Kubernetes infrastructures. Consequently, the framework is flexible and can be configured to develop applications in diverse application fields and workloads with minimal configuration.

Although the proposed approach has shown promising outcomes, there are still a number of opportunities that can be used to further advance the approach. The introduction of reinforcement learning techniques and their integration into future work is one of such directions. Reinforcement learning agents are able to dynamically optimize scaling strategies through a sequence of learning on system feedback and environmental transformations and further optimise performance over long term horizons. The other possible extension is the implementation of the framework in the multi-cluster and multi-cloud setup, where the workloads are spread into the data centers that are geographically separated. Such environments would be supported and could increase the scalability, fault tolerance and load balancing capabilities.

Moreover, enhancing the interpretability and transparency of the deep learning models is an essential research problem. Explainable artificial intelligence (XAI) methods on workload prediction that can be explained and understood by the system administrator can assist to gain additional trust over automated scaling processes. The future research could also investigate hybrid models based on deep learning and domain knowledge in order to have high accuracy and interpretation. Altogether, the offered predictive autoscaling system is a good basis to intelligent cloud resources management and creates various research and practical development opportunities.

References

[1] Lorido-Botrán, T., Miguel-Alonso, J., & Lozano, J. A. (2012). Auto-scaling techniques for elastic applications in cloud environments. Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09, 12, 2012.
[2] Ghani, N., Ahmad, I., Mohsin, M., & Khan, M. A. (2018). Predictive autoscaling for cloud resources using machine learning techniques. Journal of Cloud Computing: Advances, Systems and Applications, 7(1), 1–18.
[3] Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20(3), 273-297.
[4] Mao, M., & Humphrey, M. (2012, June). A performance study on the vm startup time in the cloud. In 2012 IEEE Fifth International Conference on Cloud Computing (pp. 423-430). IEEE.
[5] Box, G. E., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). Time series analysis: forecasting and control. John Wiley & Sons.
[6] Hyndman, R. J., & Athanasopoulos, G. (2018). Forecasting: principles and practice. OTexts.
[7] Abdi, H., & Williams, L. J. (2010). Principal component analysis. Wiley interdisciplinary reviews: computational statistics, 2(4), 433-459.
[8] Masdari, M., & Khoshnevis, A. (2020). A survey and classification of the workload forecasting methods in cloud computing. Cluster Computing, 23(4), 2399-2424.
[9] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444.
[10] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.
[11] Nikravesh, A. Y., Ajila, S. A., & Lung, C. H. (2015, May). Towards an autonomic auto-scaling prediction system for cloud resource provisioning. In 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (pp. 35-45). IEEE.
[12] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.
[13] Alshamrani, A., & Malek, S. (2019). A survey on performance prediction and auto-scaling techniques for cloud-based services. Journal of Systems and Software, 156, 110–124.
[14] Nikravesh, A. Y., Ajila, S. A., & Lung, C. H. (2015, May). Towards an autonomic auto-scaling prediction system for cloud resource provisioning. In 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (pp. 35-45). IEEE.
[15] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
[16] Karim, F., Majumdar, S., Darabi, H., & Chen, S. (2017). LSTM fully convolutional networks for time series classification. IEEE access, 6, 1662-1669.
[17] Baldan, F. J., Ramirez-Gallego, S., Bergmeir, C., Herrera, F., & Benitez, J. M. (2016). A forecasting methodology for workload forecasting in cloud systems. IEEE Transactions on Cloud Computing, 6(4), 929-941.
[18] Imdoukh, M., Ahmad, I., & Alfailakawi, M. G. (2020). Machine learning-based auto-scaling for containerized applications. Neural Computing and Applications, 32(13), 9745-9760.
[19] Yin, S., Liu, L., & Hou, J. (2016). A multivariate statistical combination forecasting method for product quality evaluation. Information sciences, 355, 229-236.
[20] Lu, S., Lu, H., & Kolarik, W. J. (2001). Multivariate performance reliability prediction in real-time. Reliability Engineering & System Safety, 72(1), 39-45.
[21] Chen, L., Bahsoon, R., & Yao, X. (2018). Self-adaptive and self-aware autoscaling for cloud resource management: A survey. ACM Computing Surveys, 51(3), 1–41.
[22] Imdoukh, M., Ahmad, I., & Alfailakawi, M. G. (2020). Machine learning-based auto-scaling for containerized applications. Neural Computing and Applications, 32(13), 9745-9760
[23] Obannagari, C. K. R. N., & Nangi, P. R. (2020). Deep Learning-Driven Compliance Automation for Continuous Monitoring of Security Controls in Regulated Cloud Systems. International Journal of Artificial Intelligence, Data Science, and Machine Learning, 1(3), 21-32. https://doi.org/10.63282/3050-9262.IJAIDSML-V1I3P104
[24] Obannagari, C. K. R. N., & Nangi, P. R. (2020). Advanced Data Science Frameworks for Predictive Cyber-Risk Assessment and Adaptive Security Policy Optimization in Zero Trust Networks. International Journal of Emerging Trends in Computer Science and Information Technology, 1(4), 67-78. https://doi.org/10.63282/3050-9246.IJETCSIT-V1I4P108