LSTM-based Forecasting for Non-linear Workloads in On-premises Software Systems

Rafael Xavier
Federal Center of Minas Gerais
Dep. of Technology Information
Belo Horizonte, Brazil
rafael.xavier@cefetmg.br

Bruno Cafeo
University of Campinas
Institute of Computing
Campinas, Brazil
cafeo@unicamp.br

Baldoino Fonseca
Federal University of Alagoas
Institute of Computing
Maceió, Brazil
baldoino@ic.ufal.br

Davy Baia
Federal University of Alagoas
Educational Unit of Penedo
Penedo, Brazil
davy.baia@penedo.ufal.br

Elder Cirilo
Federal University of São João del-Rei
Department of Computer Science
São João del-Rei, Brazil
elder@ufsj.edu.br

ABSTRACT

Cloud computing has become a commonly adopted solution to increase business operations efficiency and scalability. However, costs related to cloud infrastructure are increasingly prohibitive for some organizations in long-term deployment scenarios. This expense barrier has led to the emergence of a cloud repatriation trend, where existing software systems are moved back to on-premises environments. Although this initiative enhances cost control management and autonomy, it reintroduces technical challenges. This work investigates, through an empirical study, the applicability of Long Short-Term Memory (LSTM) networks for workload prediction to support auto-scaling decisions in on-premises environments. Initially, we contrasted the prediction accuracy of LSTM with classical methods, utilizing both standard benchmark datasets and access logs collected from an industrial-grade software system. In the second phase, we examined how varying temporal resolutions affect precision and computational expense. Results show the LSTM consistently outperforms classical methods, reducing MAPE by up to 18.02% and RMSE by 8.77% on benchmark datasets. Considering the in-field dataset, it outperforms in long-term predictions, especially regarding MAE and MAPE. Throughout all analyzed temporal resolutions, the 10-minute resolution optimally balanced accuracy with efficiency. Moreover, we noted that LSTM requires less training time than classical methods, emphasizing its applicability for real-world scenarios. Therefore, the results indicate that LSTM emerges as a viable solution to enable proactive auto-scaling within on-premises environments, potentially reducing costs in the context of cloud repatriation.

KEYWORDS

Auto-scaling, Time Series Forecasting, On-premises Software Systems, Cloud Repatriation

1 Introduction

The adoption of cloud computing has settled down in recent years as the primary option in business contexts [25, 29]. Driven by its potential to provide scalable infrastructure solutions [25], this operational model presents numerous advantages, including the virtualization of hardware, software, storage, and networking resources [7, 41]. Accordingly, cloud computing represents a fundamental shift from traditional on-premises infrastructure to an environment of remotely managed and scalable resources [1, 53]. At the core of cloud computing are the virtualization technologies [35]. They provide elasticity and scalability by allowing computational resources to dynamically adjust to varying user demands and workload fluctuations without administrative intervention [20].

Despite the overall adoption of cloud computing, organizations are now transitioning back from a cloud-first model to what is known as cloud repatriation [18, 24] (migrating software systems and resources back to on-premises infrastructures). Several factors motivate this movement [52], including concerns over data governance due to stringent regulatory requirements and the unpredictable costs associated with cloud services. A 2024 IDC study [51] revealed that approximately 80% of surveyed organizations expect to repatriate some computing and storage resources within the coming years. Indeed, organizations have reported financial benefits; for example, 37Signals [1] announced it saved over $10 million in five years after repatriating its computational resources.

An immediate challenge associated with cloud repatriation is scalability [3, 17, 45], which becomes a critical operational concern when organizations shift software systems from elastic cloud environments to fixed-capacity on-premises infrastructures [3, 34]. Unlike cloud platforms, which inherently offer virtually unlimited scalability through automated provisioning and geographically distributed resources, on-premises environments are constrained by the complexities involved in capacity planning and hardware acquisition. Consequently, organizations implementing repatriation often encounter technical obstacles in achieving the desired elasticity, especially with software systems characterized by highly fluctuating or spiky workloads. Therefore, when not well managed, these challenges undermine the quality-of-service and impact the cost benefits initially motivating the repatriation effort [34].

Implementing auto-scaling mechanisms in on-premises infrastructures presents a set of technical challenges [15, 42]. In contrast to cloud platforms that provide resource pools and hypervisor-level orchestration technologies, on-premises auto-scaling must operate subject to restricted automation capabilities [15]. In this context,

[1] https://world.hey.com/dhh/our-cloud-exit-savings-will-now-top-ten-million-over-five-years-c7d9b5bd

a critical requirement for enabling auto-scaling in on-premises infrastructures is the availability of accurate and timely workload telemetry [31], which serve as indicators of system load. The effectiveness of auto-scaling strategies also depends on the decision-making model employed [15]: whether reactive (triggered by surpassing predefined thresholds) or proactive (utilizing predictive analytics to anticipate demand and scale in advance). When accurate forecasting data is available, proactive strategies can be used to anticipate workload fluctuations and reallocate resources in advance, potentially reducing bottlenecks and performance degradation risk [23, 30, 33, 38, 50]. In contrast, reactive mechanisms respond, in general, after thresholds are breached, which can lead, in most cases, to delays during sudden demand spikes [15]. Consequently, although they are more complex to implement, proactive auto-scaling strategies, in certain conditions, provide more effective support for maintaining system stability and ensuring adherence to quality-of-service expectations in the face of fluctuating workloads [31].

Thus, effectively predicting workload is crucial for enabling a system to make proactive rather than reactive decisions [15, 31]. Workload prediction is a form of time series forecasting that assesses historical patterns to anticipate future behaviors [8, 23, 38, 49]. Various approaches can be utilized [23, 30, 33, 38, 50], from statistical methods (e.g., ARIMA – AutoRegressive Integrated Moving Average [27, 44] and ETS – Exponential Smoothing State Space [5, 33]) to more sophisticated machine learning techniques. In this work, we conducted a comparative analysis of these methods, focusing on applying LSTM [23, 38, 50] in real-world scenarios. Our empirical study utilized a non-intrusive approach that leverages system logs (e.g., request rates) as workload telemetry data to promote elasticity in repatriated software systems without requiring structural modifications. As a result, it offers the following key insights:

- Based on our analysis of sample entropy and operational metrics, the 10-minute temporal resolution represents the optimal balance between accuracy and computational cost. This finding emphasizes the importance of temporal resolutions in revealing historical patterns within time series data, providing a basis for decision-making in auto-scaling scenarios.
- While more complex than traditional ARIMA and ETS models, the LSTM model offered a remarkable 36.18% reduction in training time compared to ARIMA and 19.54% less than ETS. Notably, this enhanced efficiency does not compromise accuracy; rather, the LSTM model demonstrates superior predictive performance with an 18.02% reduction in Mean Absolute Percentage Error (MAPE). The LSTM's ability to manage multi-horizon forecasting was also shown to be essential for proactive workload prediction and timely resource adjustments, as observed in the results.
- The LSTM model achieved reductions of up to 8.77% in RMSE and 18.02% in MAPE on the benchmark dataset, outperforming the statistical methods. When applied to the industrial-grade dataset, the model yielded superior results in both MAE and MAPE, demonstrating the best performance in long-term forecasting scenarios. These findings highlight the LSTM's effectiveness in precise workload prediction, which might make it suitable for immediate decision-making in auto-scaling scenarios.

The rest of the paper is organized as follows: Section 2 presents the context about auto-scaling and time series analysis and modeling using LSTM neural networks. Section 3 details our study design. Section 4 explores and discusses the results. Section 5 describes the threats to the validity of our study. Finally, Section 6 presents the final considerations.

2 Background and Related Work

In order to lay the foundation for the rest of our paper, we provide some background and discuss related work on auto-scaling, time series analysis, and modeling in this section.

2.1 Auto-scaling

Auto-scaling refers to the autonomous allocation of computational resources to software systems in response to fluctuating workload [39]. This process involves increasing or decreasing infrastructure resources (e.g., processing units, memory, or container instances) based on pre-established policies or forecasting-based control mechanisms. The primary goal of auto-scaling is to align system capacity with immediate or predicted workload fluctuations, thereby ensuring quality-of-service without requiring manual intervention [42, 54]. In cloud computing environments, auto-scaling is a core capability [47]. And, indeed, primary cloud services (e.g., AWS [2], Azure [3], Google Cloud [4]) provide integrated orchestration layers that support the automated provisioning and de-provisioning of resources.

Auto-scaling strategies can be divided into two primary categories [15]: reactive and proactive. Reactive auto-scaling responds to real-time workload telemetry data that exceed predefined thresholds [4]. For example, CPU utilization exceeds 80%, or request latency surpasses acceptable limits. While reactive strategies are of relative simplicity and low overhead, they exhibit a lagging nature (resources are only provisioned in response to past events). This might result in long periods of under-provisioning or over-provisioning [40]. Proactive auto-scaling diverges from reactive strategies by anticipating future demand spikes using predictive analytics [50]. It leverages historical telemetry data and forecasting models to provision resources before demand arises [15].

Proactive strategies often use techniques from time series analysis [33] and machine learning [23, 30, 38, 50] to estimate future resource demands and perform scaling decisions. As stated by Dogani et al. [15], even though these strategies are more computationally intensive and complex to calibrate, proactive auto-scaling has the potential to minimize performance degradation and optimize infrastructure usage, especially in environments lacking inherent elasticity, such as on-premises infrastructures. In [50], the author proposes a proactive auto-scaling approach utilizing a BiLSTM model enhanced by attention and RL for dynamic scaling. Indeed, their solution improved quality-of-service in container-based environments by up to 20%. However, by integrating RL with the

[2] https://aws.amazon.com/autoscaling
[3] https://learn.microsoft.com/en-us/azure/azure-monitor
[4] https://cloud.google.com/compute/docs/autoscaler

BiLSTM model, their approach added intricate barriers to auto-scaling mechanism operation and maintenance for some small or medium-sized organizations. Although traditional linear models are more straightforward and easy to deploy and maintain, as stated by Ang et al. [49] and Tsay et al. [46], they are also limited in capturing complex temporal dependencies typical in real-world datasets.

2.2 Time Series Analysis

A time series refers to a sequence of observations organized in chronological order, where each data point is associated with a specific timestamp and occurs at either regular or irregular intervals. Time series analysis is an area of statistics with applications across various fields [8]. The objective of this analysis is to derive insights from chronological data to understand past behaviors and make future predictions [37]. The time series analysis often considers factors such as seasonality, autocorrelation, stationarity, and complexity. Comprehending these dynamics is essential for determining necessary transformations and identifying the most appropriate parameters for developing predictive models.

2.2.1 Stationarity. A time series is stationary when its statistical properties remain unchanged over time (e.g., mean, variance, and autocovariance). There are two types of stationarity: first-order, where only the mean and variance are continuous, and second-order, where autocovariance is also constant [22]. Stationarity can be visually confirmed using a correlogram that depicts the Autocorrelation Function (ACF). In the correlogram, stationary processes rapidly decrease to zero, while non-stationary processes show a slower decay [13]. Stationarity can also be evaluated through unit root statistical tests, such as the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test [28], which tests the null hypothesis that a time series is stationary around a deterministic trend, against the alternative hypothesis of non-stationarity due to a unit root process. Additionally, the Augmented Dickey-Fuller (ADF) test [14] assesses whether a series is stationary around a linear trend, assuming an autoregressive structure of order p [13].

2.2.2 Seasonality. Seasonality refers to the recurring and periodic patterns observed in a time series at regular intervals, which can occur daily, weekly, monthly, or at any specified period. These patterns are often described as cyclical [19]. For instance, the number of requests received by an e-commerce platform may surge during promotional campaigns or on weekends, while the load on an enterprise resource planning (ERP) system is likely to increase at the end of each month due to financial closing operations.

2.2.3 Decomposition. Time series decomposition aims to separate a non-stationary time series into its deterministic non-stationary components (e.g., trend and seasonality) and a residual stochastic component. This decomposition enables more detailed analysis and improved predictions. The deterministic components are predictable and contribute to the prediction process through their estimates or extrapolation. To enhance prediction accuracy, it is also essential to analyze the remaining stochastic component [13].

2.2.4 Complexity. Sample Entropy is a statistical measure employed to assess the complexity and predictability of time series data. The basic idea behind Sample Entropy is to examine the frequency and irregularity of patterns within the time series to quantify the uncertainty or inherent unpredictability associated with a specific dataset [48]. Generally, a higher Sample Entropy value indicates increased complexity and a more significant degree of unpredictability within the time series.

2.3 Time Series Modeling

Time series modeling encloses a range of forecasting strategies and makes selecting the most appropriate model for each scenario a significant challenge [33]. In recent decades, the area has diversified significantly regarding methods and techniques. Nevertheless, classical statistical models—such as Autoregressive Integrated Moving Average (ARIMA), Error, Trend, Seasonality (ETS), and Vector Autoregression (VAR)—continue to play a crucial role in time series analysis and forecasting. Complementarily, machine learning techniques have been employed to predict temporal relationships in time series data. For example, linear regression models are highly regarded for their ability to be easily interpreted and their efficiency in identifying and capturing simple trends in data [16]. Moreover, ensemble-based methods (e.g., Random Forests [6] and XGBoost [9]) also demonstrated strong performance, especially in situations involving high-dimensional or noisy data. Kernel-based methods like Support Vector Regression (SVR) [12] are also frequently applied to predict temporal relationships in time series data due to their effectiveness in managing non-linear relationships through well-defined regularization principles.

Recently, deep learning techniques, especially Recurrent Neural Networks (RNNs) along with their variants such as Long Short-Term Memory (LSTM) [21] and Gated Recurrent Unit (GRU) [10], have shown remarkable success in capturing long-range temporal dependencies. This success is especially evident in non-stationary or highly dynamic environments [21], where systems and processes undergo rapid workload fluctuations that traditional methods struggle to model effectively. Therefore, these models are well-suited for forecasting scenarios that involve complex temporal patterns, where manual feature engineering may be impractical or less effective, such as for auto-scaling software systems deployed in on-premises environments. In these situations, workloads can fluctuate significantly and unpredictably, making it challenging to anticipate and respond to changes using predefined features. Indeed, deep learning techniques can learn complex temporal patterns directly from raw operational data (e.g., request rates extracted from logs) and achieve more accurate resource allocation without extensive domain-specific modeling or system instrumentation.

2.3.1 Long Short Term Memory - LSTM. [21] LSTM is a typical Recurrent Neural Network (RNN) based on the artificial neural networks paradigm. Its structure mimics biological neural networks through interconnected computational units known as neurons. RNNs are specifically designed to analyze chronological data where there is a dependency between current and previous values, and LSTM stands out in this context for its exceptional ability to learn and preserve long-term dependencies in historical data [23]. Therefore, our choice of LSTM is fundamentally a design trade-off prioritizing real-time feasibility. In general, other deep learning techniques (e.g., transformer-based models and specialized Large

Language Models) often suffer from prohibitive training costs and massive data dependencies that conflict with the reality of available on-premise logs.

Indeed, by leveraging a memory cell architecture, LSTM avoids the vanishing and exploding gradient problems often faced by traditional RNNs. Within the LSTM architecture, a linear structure traverses the entire network, modulating information as it moves forward, which is complemented by three primary gates (i) Forget gate: Implemented through a sigmoid layer, decides which information should be discarded from the cell state. Each state C_{t-1} generates a value between 0 and 1, representing the amount of information to be removed; (ii) Input gate: Consisting of two layers - a sigmoid and a *tanh* - decides which new information will be stored in the cell state. The output of these two layers is multiplied and added to the cell state calculated by the forget gate; (iii) Output gate: Uses a combination of sigmoid and *tanh* layers to produce a filtered version of the cell state, determining the network's final output based on the current state.

3 Study Design

We conducted an empirical study to evaluate a workload forecasting model centered on historical access logs as the primary telemetry data and designed for seamless integration into software systems deployed in on-premises environments. Our study focused on two main aspects: (i) the influence of temporal granularity on forecasting accuracy and (ii) the effectiveness of LSTMs for workload forecasting. Specifically, we conducted a comparative analysis of the LSTM model against traditional statistical methods (ARIMA and ETS) and machine-learned techniques. We utilized two distinct datasets: ClarkNet [5], a benchmark dataset, and an in-field dataset derived from a production-grade software system operating in an on-premises environment.

3.1 Goal and Research Questions

This study aims to evaluate a forecasting model for request-based workloads in software systems deployed in on-premises environments, particularly in cloud repatriation scenarios. We used the organization proposed by the Goal/Question/Metric to define the goals of our study. According to the proposed goal definition template, the scope of our study can be summarized below. In addition, based on our goal, we came up with two research questions (RQs), which are also presented below.

> **Analyze** request-based workload time series
> **with the purpose of** evaluating LSTM-based forecasting models
> **with respect to** prediction accuracy, training time, resolution sensitivity, and multi-horizon forecasting performance
> **from the point of view of the** software reliability engineering
> **in the context of** proactive auto-scaling strategies for on-premises software systems undergoing cloud repatriation, using historical access logs as the primary telemetry source.

- **RQ1: What is the impact of temporal granularity on the effectiveness of forecasting models?** Choosing the proper temporal resolution involves balancing between model accuracy and computational overhead. While coarser time intervals make detecting workload fluctuations more challenging, excessively fine-grained temporal resolution can increase training time, cause overfitting, and result in high training costs. Various studies have adopted different temporal resolutions for workload forecasting. In Kumar et al. [26], the authors employed 1-minute intervals to capture high-frequency temporal fluctuations, although this could introduce undesired noise. In contrast, Singh et al. [43] and another study by Kumar et al.[27] opted for 10-minute intervals, which can smooth out short-term fluctuations at the price of losing finer-grained dynamics. Messias et al. aggregated 1-second intervals into hourly maximums to align with cloud billing periods. Although temporal resolution significantly affects time series forecasting, as we can see, current research provides limited guidance on choosing granularity for optimal performance. In this research question, we examine how different sampling resolutions influence forecasting models' predictive accuracy and computational efficiency in real-world workload scenarios.
- **RQ2: What is the impact of LSTM on workload forecasting based on request rate?** In this research question, we investigate the effectiveness of LSTM in handling complex and non-linear time series characterized by sudden spikes, conditional variation over time, and temporal irreversibility [36] - typical patterns in workloads of software systems. Indeed, a study by Janardhanan et al. [23] shows that LSTM, owing to its ability to model non-linear temporal dynamics, produces more stable forecasting performance than ARIMA in the context of CPU workload prediction. These findings highlight the suitability of LSTM models for forecasting in environments with sudden workload fluctuations and non-stationary behavior, where linear models often fail to adapt due to their limited expressiveness in capturing temporal non-linearities.

3.2 Datasets

As mentioned, our study was based on log data from two distinct sources: the ClarkNet Traces, our benchmark, and log records from a software system deployed and operating in an on-premise environment. The log format includes several elements (see Figure 1), including the source IP address, a timestamp, details of the HTTP request initiation, the returned HTTP status code, the number of bytes transmitted, reference information, user agent data, and the time taken to process the request. This log-based historical data

```
27.16.0.40 - [03/May/2024:22:41:51 -0300]
"GET /home/ HTTP/1.1" 200 16323
"https://www.google.com/"
"Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/124.0.0.0 Mobile Safari/
537.36"
116486
```

Figure 1: Example of a log entry containing request metadata

[5] https://ita.ee.lbl.gov/html/contrib/ClarkNet-HTTP.html

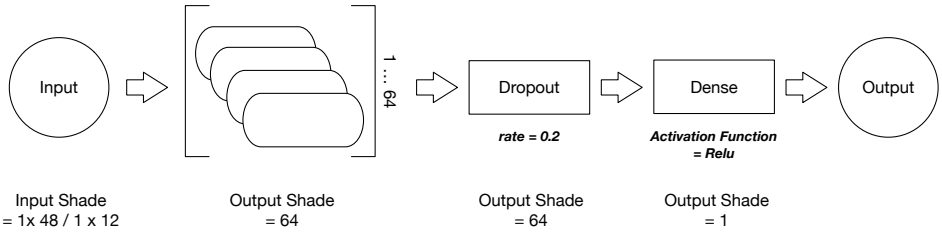

Figure 2: WFLS-LSTM Model

enabled our analysis of access peaks, allowing us to identify performance degradation and bottlenecks over time.

3.2.1 Benchmark dataset - ClarkNet Traces. It consists of two weeks of records of all HTTP requests to the ClarkNet WWW server, an Internet Service Provider (ISP) for the Baltimore-Washington DC metropolitan area, publicly available [11]. The resulting time series comprises 1,209,511 observations of the number of HTTP requests per second, with values ranging from 0 to 45 and an average of 3 requests per second.

3.2.2 In-Field System Dataset. The second dataset was collected from a web-based enterprise software system to support evaluating forecasting models under real operational conditions. It comprises approximately three months of logs extracted from the system's execution environment, resulting in a time series with 8,875,129 observations. This series is nearly seven times longer than the ClarkNet Traces benchmark and exhibits substantially greater variability in request rates, with values ranging from 0 to 900 requests per second and a mean of approximately three requests per second. Such dispersion reflects the presence of short-lived workload fluctuations, which may have affected system performance during the observed period. This dataset provides a realistic industrial-grade foundation for validating the applicability and robustness of the proposed forecasting model in a production-like context.

3.3 Data Extraction and Time Series Analysis and Modeling

We started the data extraction, time series analysis, and modeling by collecting the historical access logs, which served as our primary telemetry source. After extraction, the raw logs were cleaned and anonymized to comply with privacy constraints and avoid exposing user-specific data. The data anonymization involved removing sensitive information: IP addresses, URL query parameters, and referrer headers that could link requests to individual users or sessions. An illustrative example of a typical log entry is shown in Figure 1. After anonymization, we transformed these logs into time series that model the request frequencies per second. Subsequently, we conducted the exploratory analysis across various temporal resolutions (1-minute, 10-minute, 1-hour). In this step, we examined how these three sampling resolutions impact forecasting models' predictive accuracy and computational efficiency in realistic workload scenarios.

The next step involved developing the predictive model. Given the inherent non-linearity and complexity of the time series created in the previous step, we used a Long Short-Term Memory (LSTM) network. This choice is supported by existing literature. As Ang et al. [49] and Tsay et al. [46] pointed out in their work, the traditional linear models are limited in capturing complex temporal dependencies typical in real-world datasets. Indeed, LSTMs were designed explicitly for processing chronological data and excel at modeling intricate temporal patterns. Figure 2 presents the proposed model architecture (WFLS-LSTM), which includes an LSTM layer with 64 units, followed by a dropout layer (20% random deactivation during training) and a dense layer with one unit. The dense layer employs the ReLU activation function, defined as $f(x) = \max(0, x)$, ensuring all predictions are non-negative. The input window size is set to 48 for 1-minute and 10-minute time series, and 12 for hourly series, corresponding to the number of previous observations used to forecast the next value. The total number of trainable parameters is 39,745.

Next, we performed a systematic hyperparameter optimization process to enhance the predictive performance of the model built in the previous step. Given the sensitivity of LSTM architectures to hyperparameters such as the number of hidden units, learning rate, batch size, and dropout rate, we employed a randomized search strategy combined with cross-validation to identify configurations that balanced accuracy and training efficiency. Unlike model parameters learned from the data, hyperparameters must be carefully selected before training, as they directly influence the model's capacity to generalize across different workload scenarios. By utilizing hyperparameter optimization, we can improve model performance beyond the default configurations offered by standard prediction libraries [32]. In this study, we employed the Grid Search technique, which systematically explores all possible combinations of hyperparameters within a predefined search space to recognize the configuration that maximizes model performance.

Table 1 summarizes the hyperparameters and their corresponding value ranges explored during optimization. We considered as hyperparameters the learning rate, the number of units across up to four sequential LSTM layers, and the dropout rate. Three distinct loss function options were evaluated: mean squared error (MSE), mean squared logarithmic error (MSLE), and Huber loss. The final model was configured to utilize the Mean Squared Error (MSE) loss function, primarily due to its widespread use and effectiveness despite its sensitivity to outliers. Given its computational efficiency and adaptive learning rate during training, the chosen optimizer was Adam. It was configured with a specific learning rate denoted as 0.00015. The Early Stopping technique was implemented. This technique interrupts the training process if the validation error

Table 1: Hyperparameters and Tested Values

Hyperparameter	Values
Learning rate	0.00015, 0.001, 0.01
Units 1st layer	64, 240, 256, 512
Units 2nd layer	0, 32, 64, 128
Units 3rd layer	0, 16, 32
Units 4th layer	0, 8, 16
Dropout rate	0.2, 0.3
Loss function	mean_squared_error, mean_squared_logarithmic_error, huber

does not improve over 40 consecutive epochs, ensuring that the model remains generalizable and performs well on unseen data.

Finally, the model evaluation step included a performance assessment on training and independent test datasets. To conduct the comparative analysis, we utilized four commonly used metrics: Root Mean Squared Error (RMSE), Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and Mean Absolute Error (MAE). These metrics were chosen because they provide our study with a clear baseline for comparison. The model's predictive capabilities were also assessed across various forecast horizons (6, 12, 48, and 144 steps ahead), representing increasingly long-term predictions and reflecting standard operational planning intervals. We also examined computational efficiency in training duration and prediction generation speed across the entire test dataset, confirming its suitability for real-time applications such as automated scaling of software systems in on-premise environments.

3.4 Experiment Setup

The experiments were conducted in an empirical environment with an Intel Core i7-1165G7 processor, 20 GB of RAM, and an Intel Iris Xe GPU. We employed the TensorFlow 2.16.1 framework in a Python 3.10.11 environment on Windows 11 to train the deep learning models. The training, executed on a single computer without GPU acceleration, had its total time recorded using TensorFlow's time logging function.

4 Results and Discussion

In this section, we analyze and discuss the results. First, we present the empirical evaluation of the forecasting model's performance using the Clarknet Traces benchmark dataset, which addresses research question RQ1. Following that, we provide an in-depth examination of the model's effectiveness when applied to the in-field dataset, thereby answering research question RQ2.

4.1 RQ1: What is the impact of temporal granularity on the effectiveness of forecasting models?

In order to answer RQ1, we used the ClarkNet dataset to explore how varying sampling resolutions influence the predictive accuracy and computational efficiency of the proposed forecasting model

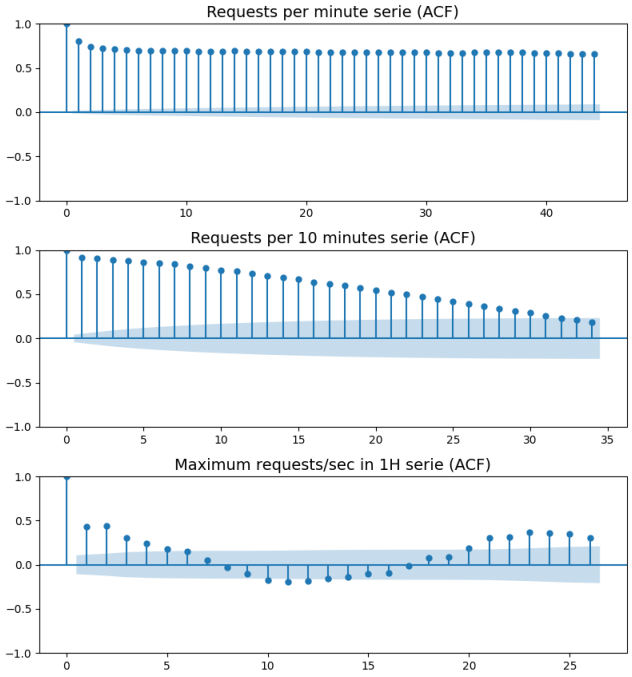

Figure 3: ACF - Clarknet Traces at different resolutions

(WFLS-LSTM). To contextualize the results, we compared the performance of our LSTM-based model against the findings reported in prior studies [26] [27] [43], which evaluated a broad range of statistical and machine-learn models, including LR, SVR, AR, MA, ARMA, ARIMA, TASM, KNN, SVM, and Naïve.

Temporal Dynamics Analysis. As we can see in Figure 3, the slow and gradual decline observed in the Autocorrelation Graphs (ACF) suggests the presence of long-term dependencies typically associated with autoregressive or near-non-stationary processes. However, despite this visual analysis, the results from the Augmented Dickey-Fuller (ADF) and Kwiatkowski-Phillips-Schmidt-Shin (KPSS) tests confirmed the stationarity of the time series across all sampling resolutions. Specifically, the ADF tests (1-minute, 10-minute, and 1-hour) yielded significantly negative statistics with p-values below 0.001 (0.00026, 7.45×10^{-7}, 1.09×10^{-8}, respectively), while KPSS statistics remained below the critical value thresholds at the 5% level, consistently indicating stationarity for all series (0.3989, 0.1423, 0.1362). This apparent discrepancy highlights that persistent temporal dependencies may still exist within the ClarkNet dataset. This time series characteristic makes LSTM a fitting choice, primarily because it was designed to learn from and utilize long-range patterns through internal memory structures.

The time series analysis also revealed the presence of seasonality, as demonstrated by the Autocorrelation Graph depicted in Figure 4. This finding indicates that the series is not stationary, as evidenced by pronounced peaks in the autocorrelation at lags corresponding to multiples of 1440, 144, and 24 (1-minute, 10-minute, and 1-hour intervals, respectively). Such peaks indicate the existence of daily,

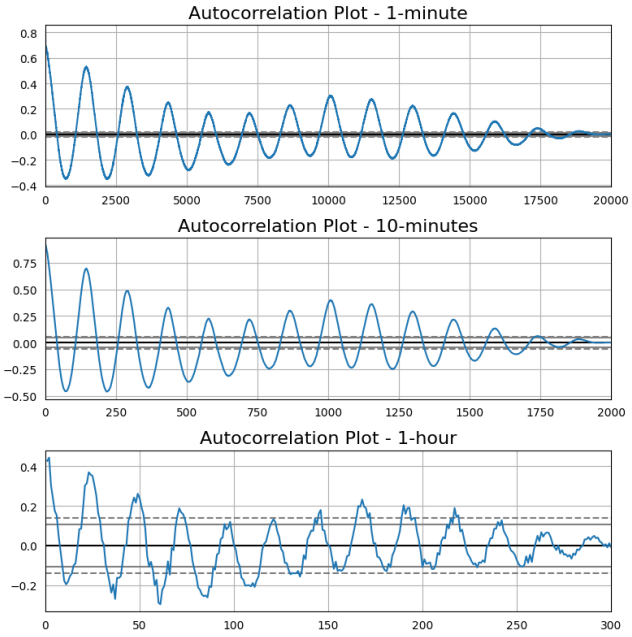

Figure 4: Seasonality through autocorrelation plot

10-minute, and hourly periodic behaviors in the dataset. These results emphasize the significant seasonal dynamics embedded within the observed data and show that models capable of harnessing and optimizing long-range dependencies, such as those that follow the LSTM architecture, are beneficial and crucial in such cases.

The Sample Entropy analysis conducted across various temporal resolutions of the ClarkNet time series reveals a heterogeneous complexity distribution. The 1-minute and 1-hour series show higher entropy values of 1.2733 and 1.5431, respectively, while the 10-minute series presents a significantly lower entropy of 0.9202. These entropy patterns thus support our observation that the workload exhibits non-linear and scale-sensitive characteristics. Accordingly, opting for models that can capture the temporal irregularities and latent dependencies – such as LSTM networks – is not only a suggested choice but also aligns with the structural properties that arise from the workload telemetry data.

Predictive Performance Across Time Resolutions. To assess the effectiveness of the proposed model, we conducted a comparative evaluation against existing approaches that employed the same benchmark dataset. The analysis considered standard forecasting error metrics—MAE, MSE, RMSE, and MAPE—across different sampling resolutions (1-minute, 10-minute, and 1-hour). The following results demonstrate the model's performance in prior studies and highlight its predictive advantages in various temporal granularities.

Table 2 presents the comparative results of forecasting models evaluated over the 1-minute resolution time series based on standard error metrics. The proposed model consistently yielded the lowest error rates across all metrics (MAE, MSE, RMSE, and MAPE) among the alternatives examined. The reduction observed in RMSE (12.95%) and MAPE (18.28%) is particularly noteworthy. It

Table 2: Comparison - resolution series / min

Model	MAE	MSE	RMSE	MAPE
LR[1]	-	-	64.37	34.52
SVR[1]	–	-	64.67	32.95
AR[1]	-	-	54.60	30.42
MA[1]	–	-	59.24	33.14
ARMA[1]	-	-	59.25	32.98
ARIMA[1]	-	-	51.24	26.64
TASM[1]	-	-	42.24	20.63
WFLS-LSTM*	**27.78**	**1352.25**	**36.77**	**16.86**

[1] Results from [26]; * Results from this work.

reflects improved absolute predictive accuracy and a more stable relative performance. These results support the robustness of the model when confronted with the fine-grained temporal fluctuations characteristic of high-frequency web traffic data.

Table 3 provides a comparison of the models applied to time series data sampled at a 10-minute resolution. The results reveal that the proposed model achieved consistently lower error values in all evaluation metrics in contrast to its counterparts. In particular, improvements of 4.65% in both RMSE and MSE metrics were observed in the TASM model [43]. This model has previously been recognized as the most accurate in earlier studies. Furthermore, a 15.24% reduction in MAPE and a modest but consistent enhancement of 0.94% in MAE further support our model's ability to generalize effectively under intermediate temporal aggregation.

Table 4 presents a comparative analysis of the models trained on time series data with a one-hour resolution. The evaluation of the proposed model's performance revealed an improvement of 7.94% in the RMSE metric compared to the reference model. Additionally, a 2.30% enhancement was noted in terms of the MAE metric. However,

Table 3: Comparison - resolution series / 10min

Model	MAE	MSE	RMSE	MAPE
KNN[1]	210.45	70265.12	250.81	12.45
LR[1]	265.23	79638.23	295.14	15.24
SVM[1]	250.63	95325.48	320.15	18.26
ARMA[1]	220.30	86257.26	250.45	15.69
ARIMA[1]	168.26	58234.18	235.72	11.52
Naïve[2]	192.75	65561.45	256.04	12.87
LR[2]	207.63	79569.25	282.08	14.52
AR[2]	178.81	57862.46	240.54	12.41
MA[2]	197.15	69282.43	263.21	13.78
ARMA[2]	219.24	81878.24	286.14	15.19
ARIMA[2]	181.19	58415.94	241.69	12.47
SVR[2]	230.65	93,023.06	304.99	16.12
TASM[2]	152.40	45911.11	214.26	10.72
WFLS-LSTM*	**150.97**	**41747.34**	**204.32**	**9.09**

[1] Results from [27]; [2] Results from [43]; * Results from this work.

Table 4: Comparison - resolution series /hour

Model	MAE	MSE	RMSE	MAPE
Naïve[1]	5.37	-	7.41	28.01
AR[1]	4.46	-	6.44	22.90
ARIMA (1, 1)[1]	4.29	-	6.16	**22.07**
ARIMA[1]	4.51	-	6.33	23.96
ETS[1]	4.45	-	6.24	23.24
GA[1]	4.35	-	6.17	22.40
WFLS-LSTM*	**4.25**	**32.21**	**5.68**	22.82

[1] Results from [33]; * Results from this work.

it is important to highlight that, regarding the MAPE metric, the proposed model demonstrated a decline in performance of 3.40% relative to the model it was benchmarked against.

Prediction Behavior and Model Efficiency. Figure 5 illustrates the forecasting behavior of the proposed LSTM model applied to the ClarkNet workload traces at three distinct temporal resolutions: 1-minute, 10-minute, and 1-hour. The visualizations contrast the original series (blue) with the predictions generated during training (orange) and testing (green). These results evidence the model's capacity to follow both the short-term fluctuations of the data. As we can see in the results, the alignment between predicted and actual values is consistent in the 10-minute resolution. These observations correspond to the lowest entropy level observed in the previous analyses.

Moreover, the computational cost associated with each configuration reveals a scaling pattern aligned with temporal granularity: training the model required 444.19 seconds for the 1-minute series, 208.95 seconds for the 10-minute series, and 35.77 seconds for the 1-hour series. Prediction times were 3.46, 0.65, and 0.44 seconds, respectively. These results emphasize not only the scalability of the LSTM model across resolutions but also its sensitivity to the structural complexity embedded in each model of the workload.

Discussion. The results revealed significant differences in predictive performance and computational efficiency trade-offs based on the resolution of the time series data. The 1-minute series yielded the most accurate results, but this gain came at the cost of significantly longer training and inference times. This implication may limit its applicability in latency-sensitive auto-scaling scenarios. On the other hand, the 10-minute resolution series, despite not being the most accurate in absolute terms, showed the lowest entropy and a more regular structure. It also presented the noteworthy training (208.95 seconds) and prediction (0.65 seconds) times. Finally, the 1-hour resolution failed to maintain predictive superiority compared to baseline statistical and machine-learning approaches. Its coarse granularity may obscure short-term workload shifts, which is critical in auto-scaling contexts. From our perspective, particularly in on-premises infrastructures where adaptation must be timely and resource-aware, the 10-minute resolution emerges as the most operationally viable alternative for proactive workload forecasting.

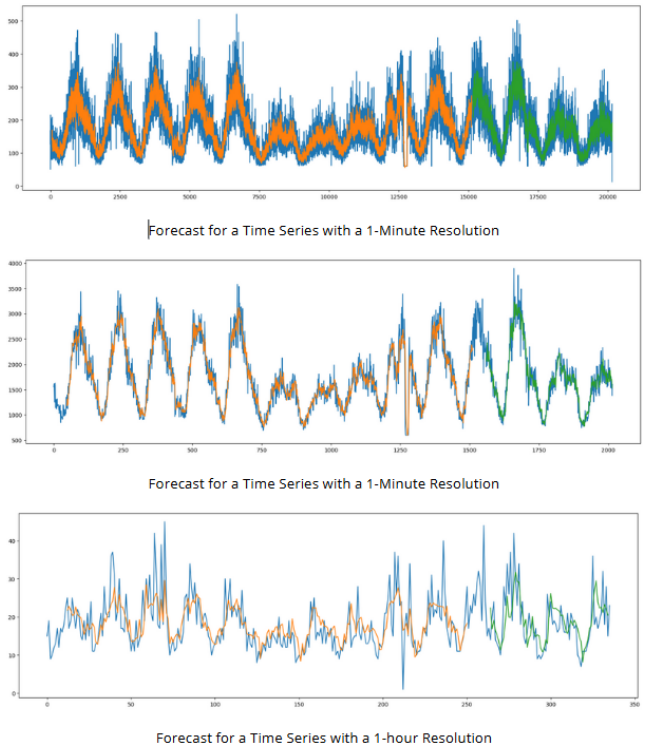

Figure 5: Predictions in series at different resolutions

4.2 What is the impact of LSTM on workload forecasting based on request rate?

The time series analysis of the in-field dataset indicated striking similarities with the Clarknet dataset. The stationarity statistical tests demonstrate that the datasets are stationary, except for the 1-minute resolution series. The autocorrelation analysis also exhibited the characteristics of slow decay and revealed peaks at markers consistent with those in the Clarknet series, highlighting the presence of seasonality. Additionally, the analysis reveals greater volatility in this dataset, as illustrated by the trend obtained through time series decomposition (see Figure 6). This evidence reinforces the need to utilize models capable of effectively capturing these non-linear patterns.

LSTM vs. Statistical Models on In-Field Workload. To compare the proposed LSTM model with traditional statistical methods, we applied both approaches to an industrial-grade workload time series obtained from an operational web system. Based on the previous research question (RQ1) results, which indicated that a 10-minute resolution offers the best balance between predictive

Figure 6: Trend - production-grade software system

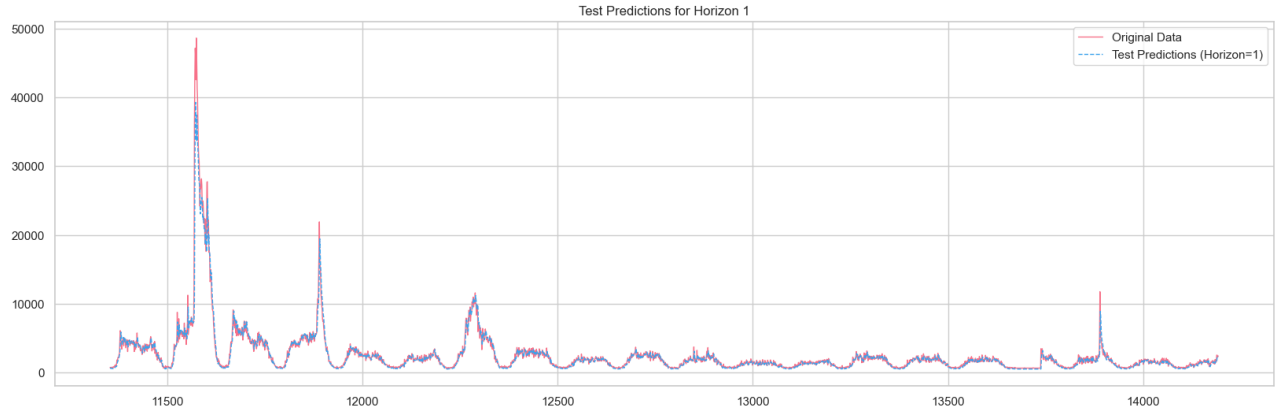

Figure 7: Prediction on test set

performance and series complexity, we resampled the original request log to 10-minute intervals, resulting in a dataset with 14,793 observations. This dataset preserves the original workload pattern while reducing high-frequency noise and computational cost. As shown in Table 5, the error metrics (RMSE and MSE) are higher than those of Clarknet. This discrepancy arises because the actual series is approximately seven times larger than the benchmark series, has 20 times greater scale values, and contains sudden peaks in the data. RMSE, as an error measure, considers the squared differences between forecasted and actual values. Consequently, with a larger series scale, the absolute differences between forecasted and actual values also increase, resulting in a higher RMSE. The results in Table 5 show that the LSTM model achieved superior results in MAPE and MAE, reducing relative error by 11.06% and 4.75%, respectively, compared to the ETS model. However, we observed that the ETS model outperformed our model when considering the RMSE and MSE, achieving improvements of 8.90% and 17.02%.

Accuracy and Operational Suitability. Figure 7 shows the alignment between the observed request rate and the LSTM predictions over the test dataset, using a forecast horizon of one step ahead. As we can see, our proposed model demonstrated strong adherence to daily seasonal patterns and is notably responsive to abrupt fluctuations and peak loads, characteristics commonly observed in real-world web traffic. These results also show the LSTM's model capacity to capture complex non-linear dependencies in high-variance time series. In addition, Table 6 presents the results of a quantitative assessment of the WFLS-LSTM model's performance across various forecast horizons (6, 12, 48, and 144 steps ahead). In this case, we calculated each prediction independently for a future time step rather than depending on sequential multi-step outputs. These results revealed that, throughout all horizons and across all error metrics – including RMSE, MSE, MAE, and MAPE – our proposed LSTM model consistently demonstrated superior performance compared to the classical ETS and ARIMA models.

We also assessed the time required for training and adjusting the ARIMA, ETS, and LSTM models; and the forecasting effort considering the entire in-field dataset. Notably, the proposed model exhibited competitive computational efficiency given the characteristics of the industrial-grade time series. It is important to recognize that once the LSTM model is trained, it can generate forecasts efficiently without retraining or adjusting for each new prediction. In contrast, the ETS and ARIMA models demonstrated competitive performance only when using a rolling window forecasting strategy. Indeed, this statistical method requires continuous adjustments for each new forecast, and as we may notice, ongoing adjustments are computationally intensive and impractical for proactive autoscaling in on-premise environments. Overall, the time needed for these adjustments and forecasts may exceed the intervals necessary for effectively scaling the system to accommodate sudden spikes.

Table 5: Model Evaluation

Model	RMSE	MSE	MAE	MAPE
ETS	**850.98**	**724167.49**	386.67	16.46
ARIMA	966.28	933702.02	386.90	19.53
WFLS-LSTM	934.17	872672.35	**368.30**	**14.64**

Table 6: Model evaluation for different prediction horizons

Model	H	RMSE	MSE	MAE	MAPE
ETS	6	1906.05	3633032.77	1840.26	273.19
	12	1659.03	2752394.03	1597.76	230.43
	48	3681.50	13553513.67	3113.67	128.31
	144	3237.67	10482571.64	2770.48	98.22
ARIMA	6	337.00	113572.17	319.61	47.41
	12	473.55	224250.77	447.27	63.33
	48	1879.95	3534238.48	1472.32	52.29
	144	1751.76	3068676.17	1500.60	53.92
WFLS-LSTM	6	**213.81**	**45713.88**	**185.04**	**27.98**
	12	**145.19**	**21080.05**	**125.65**	**18.05**
	48	**780.10**	**608562.09**	**529.40**	**27.22**
	144	**758.12**	**574743.36**	**552.90**	**19.70**

Discussion. The results of the in-field dataset analysis showed that while the LSTM model achieved better performance in absolute metrics (MAE and MAPE), ETS models excelled in quadratic metrics (RMSE and MSE), which are more sensitive to significant errors. These results show that LSTM networks can minimize average errors, increasing forecast accuracy. However, its inferior performance in quadratic metrics (RMSE and MSE) suggests a reduced ability to capture abrupt peaks and outliers with the same precision as ETS models, particularly in datasets with high variability. Based on this study, we might emphasize key advantages of our LSTM model in contexts requiring periodic and real-time forecasting: (i) high efficiency as the inference phase remains computationally efficient despite the considerable training complexity of recurrent neural networks; (ii) reduced need for retraining or recalibration, which in general results into lower operational costs; (iii) ability to perform accurate multi-horizon forecasts, enabling the development of sophisticated and proactive auto-scaling methods.

5 Threats to Validity

To ensure the validity of our study results, we considered several potential threats and addressed them as follows:

Internal Validity. Internal validity may be compromised by errors in data collection and analysis. To mitigate this risk, we relied on reliable data sources: the benchmark dataset was obtained through official repositories, while the in-field dataset was collected directly from server access logs. Additionally, pre-processing steps were implemented to minimize potential errors.

External Validity. A key threat to external validity is the generalizability of our results to systems beyond the scope of this study. While the study focuses on on-premise software systems, the statistical analysis of time series from both datasets, despite their differing request scales (0-45 and 0-900 requests/second), revealed shared properties such as stationarity and seasonality. These commonalities suggest the model's applicability to other web systems with similar patterns. However, it may exhibit limitations for systems with markedly different characteristics, such as corporate intranets or restricted-access environments, where user behavior and resource utilization patterns diverge significantly.

Construct Validity. The use of request rate as the primary metric introduces a potential threat, as it does not explicitly account for resource consumption. However, this metric was chosen for its computational efficiency and its direct correlation with user behavior, which indirectly influences resource usage. Other metrics, such as response time and resource utilization, could complement the auto-scaling system, particularly in reactive approaches, to provide a more comprehensive view of system performance.

6 Conclusion

In this work, we empirically evaluated the usefulness of LSTM deep learning algorithms for enabling elasticity within on-premise environments, particularly concerning repatriated software systems. Our research centered on two primary dimensions: (i) the impact of temporal granularity on forecasting accuracy; and (ii) the performance of LSTM neural networks in workload forecasting. We compared a proposed LSTM model (WFLS-LSTM) against conventional statistical methodologies and a broad range of machine-learn models. We employed two distinct datasets: ClarkNet and an additional dataset derived from logs from a software system operating in an on-premises environment.

Our analysis of the impact of temporal granularity on forecasting accuracy indicated that the proposed LSTM model outperformed traditional statistical methods and other machine learning algorithms. Specifically, at a 1-minute resolution, the WFLS-LSTM model achieved an 8.77% reduction in RMSE and an 18.02% decrease in MAPE, while demonstrating robust performance across multiple time scales. The temporal resolution analysis indicated that the 10-minute resolution balanced prediction accuracy and computational efficiency. As we could observe, the 1-minute resolution, while potentially offering finer granularity and presenting the best results, might incur substantially increased computational overhead. On the other hand, the 1-hour resolution lacks the sensitivity required to detect important short-term workload variations.

The WFLS-LSTM model, when applied to industrial-grade time series, demonstrated competitive performance. Our proposed model consistently outperformed ARIMA and ETS models in absolute error metrics such as MAE and MAPE. In particular, the ETS model achieved superior performance in quadratic error metrics (RMSE and MSE). These metrics are more sensitive to more considerable variations in actual values. Therefore, these findings emphasize the inference capabilities of LSTM algorithms. Unlike ARIMA and ETS, which demand regular retraining and continuous parameter tuning, LSTMs demand less frequent retraining. These results shed some light on the potential of LSTM models to support proactive auto-scaling in an on-premise environment, which requires real-time workload forecasting to maintain quality of service policies.

In forthcoming research, we plan to extend the application of the WFLS-LSTM model to a broader range of software systems that produce operational logs, such as IoT platforms and distributed software systems. It is expected that the 10-minute temporal resolution and model configuration evaluated in the study will generalize effectively, maintaining the promising trade-off between forecasting accuracy and computational efficiency, even if retraining is required for domain-specific dynamics. From an architectural perspective, the proposed methodology aligns with the MAPE-K autonomic computing paradigm. It enables components to learn from historical behavior and formulate context-aware action plans. Finally, there is an intention to utilize the WFLS-LSTM model in Kubernetes-based infrastructures as a predictive mechanism to inform custom metrics for auto-scaling controllers and to evaluate its effectiveness in supporting proactive scaling decisions in container-based environments.

ARTIFACT AVAILABILITY

The artifacts used for the study, including the raw data and the experimental results, among others, are available on Zenodo [2].

ACKNOWLEDGEMENTS

The authors gratefully acknowledge financial support from FAEPEX (grants No. 3404/23 and No. 2382/24) and from CAPES for its support of the Graduate Program in Computer Science at UFSJ.

REFERENCES

[1] CP Amajuoyi, LK Nwobodo, and MD Adegbola. 2024. Transforming business scalability and operational flexibility with advanced cloud computing technologies. *Computer Science & IT Research Journal* 5, 6 (2024), 1469–1487.

[2] anon. 2025. LSTM-based Forecasting for Non-linear Workloads in On-premises Software Systems. doi:10.5281/zenodo.15285346 Computational notebook.

[3] Kuciuk Artiom. 2025. Cloud Migration Framework: Transitioning from On-Premises to Azure Cloud for Improved System Reliability and Scalability. *The American Journal of Applied sciences* 7, 02 (2025), 5–11.

[4] Dariusz Rafal Augustyn. 2017. Improvements of the Reactive Auto Scaling Method for Cloud Platform. In *Computer Networks*, Piotr Gaj, Andrzej Kwiecień, and Michał Sawicki (Eds.). Springer International Publishing, Cham, 422–431.

[5] V Bandari. 2020. Cloud Workload Forecasting with Holt-Winters, State Space Model, and GRU. *Journal of Artificial Intelligence and Machine Learning in Management* 4, 1 (2020), 27–41.

[6] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32.

[7] Hui Chang. 2022. The Differences and Advantages between Cloud Services and Traditional Services. In *2022 8th Annual International Conference on Network and Information Systems for Computers (ICNISC)*. 497–499.

[8] C. Chatfield. 2016. *The Analysis of Time Series: An Introduction* (6th ed.). Vol. 11. Chapman and Hall/CRC. 2834–2839 pages.

[9] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794.

[10] Kyunghyun Cho, Bart van Merrienboer, Yoshua Bengio, and Holger Schwenk. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of EMNLP 2014*. https://arxiv.org/abs/1406.1078

[11] ClarkNet. 1995. ClarkNet HTTP Dataset. https://ita.ee.lbl.gov/.

[12] C. Cortes and V. Vapnik. 1995. Support-Vector Networks. *Machine Learning* 20, 3 (1995), 273–297.

[13] Fatoumata Dama and Christine Sinoquet. 2021. Analysis and modeling to forecast in time series: a systematic review. *CoRR* abs/2104.00164 (2021). arXiv:2104.00164 https://arxiv.org/abs/2104.00164

[14] D. Dickey and Wayne Fuller. 1979. Distribution of the Estimators for Autoregressive Time Series With a Unit Root. *JASA. Journal of the American Statistical Association* 74 (06 1979). doi:10.2307/2286348

[15] Javad Dogani, Reza Namvar, and Farshad Khunjush. 2023. Auto-scaling techniques in container-based cloud and edge/fog computing: Taxonomy and survey. *Computer Communications* 209 (2023), 120–150.

[16] N.R. Draper and H. Smith. 1998. *Applied Regression Analysis*. Wiley.

[17] Cameron Fisher. 2018. Cloud versus on-premise computing. *American Journal of Industrial and Business Management* 8, 9 (2018), 1991–2006.

[18] A Shaji George. 2024. The Cloud Comedown: Understanding the Emerging Trend of Cloud Exit Strategies. *Partners Universal International Innovation Journal* 2, 5 (2024), 1–32.

[19] Stephen Haben, Martin Voss, and William Holderbaum. 2023. Previsão de Séries Temporais: Conceitos e Definições Essenciais. In *Conceitos Básicos e Métodos em Previsão de Carga*. Springer. doi:10.1007/978-3-031-27852-5_5

[20] Nikolas Roman Herbst, Samuel Kounev, and Ralf Reussner. 2013. Elasticity in cloud computing: What it is, and what it is not. In *10th international conference on autonomic computing (ICAC 13)*. 23–27.

[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[22] Rob J. Hyndman and George Athanasopoulos. 2021. *Forecasting: Principles and Practice* (3rd ed.). OTexts, Melbourne, Australia. http://otexts.com/fpp3/

[23] Deepak Janardhanan and Enda Barrett. 2017. CPU workload forecasting of machines in data centers using LSTM recurrent neural networks and ARIMA models. In *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*. 55–60.

[24] Kiran Jewargi. 2023. Public Cloud to Cloud Repatriation Trend. *Sch J Eng Tech* 1 (2023), 1–3.

[25] Abul Khayer, Md. Shamim Talukder, Yukun Bao, and Md. Nahin Hossain. 2020. Cloud computing adoption and its impact on SMEs' performance for cloud supported operations: A dual-stage analytical approach. *Technology in Society* 60 (2020), 101225.

[26] Jitendra Kumar and Ashutosh Kumar Singh. 2018. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems* 81 (2018), 41–52.

[27] Krishan Kumar, K. Gangadhara Rao, Suneetha Bulla, and D. Venkateswarulu. 2021. Forecasting of Cloud Computing Services Workload using Machine Learning. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)* 12, 11 (2021), 4841–4846.

[28] Denis Kwiatkowski, Peter C.B. Phillips, Peter Schmidt, and Yongcheol Shin. 1992. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics* 54, 1 (1992), 159–178. doi:10.1016/0304-4076(92)90104-Y

[29] HS Lim and Wahidah Husain. 2013. A study on cloud computing adoption in e-business. *Jurnal Sistem Informasi* 9, 1 (2013), 13–17.

[30] A. I. Maiyza, N. O. Korany, K. Banawan, et al. 2023. VTGAN: hybrid generative adversarial networks for cloud workload prediction. *Journal of Cloud Computing* 12 (2023), 97.

[31] Mohammad Masdari and Afsane Khoshnevis. 2020. A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing* 23, 4 (2020), 2399–2424.

[32] S. Meisenbacher, M. Turowski, K. Phipps, M. Rätz, D. Müller, V. Hagenmeyer, and R. Mikut. 2022. Review of automated time series forecasting pipelines. *WIREs Data Mining and Knowledge Discovery* 12 (2022). Issue 6.

[33] Valter Rogério Messias, Julio Cezar Estrella, Ricardo Ehlers, Marcos José Santana, Regina Carlucci Santana, and Stephan Reiff-Marganiec. 2016. Combining time series prediction models using genetic algorithm to autoscaling Web applications hosted in the cloud infrastructure. *Neural Computing and Applications* 27, 8 (1 Nov 2016), 2383–2406.

[34] Abdulghafour Mohammad and Yasir Abbas. 2024. Key Challenges of Cloud Computing Resource Allocation in Small and Medium Enterprises. *Digital* 4, 2 (2024), 372–388.

[35] Rafael Moreno-Vozmediano, Rubén S Montero, and Ignacio M Llorente. 2012. Iaas cloud architecture: From virtualized datacenters to federated cloud infrastructures. *Computer* 45, 12 (2012), 65–72.

[36] P.A. Morettin and C.M.C. Toloi. 2018. *Análise de séries temporais: modelos lineares univariados*. BLUCHER.

[37] A. Nielsen. 2019. *Practical Time Series Analysis: Prediction with Statistics and Machine Learning*. O'Reilly Media.

[38] E. Patel and D. S. Kushwaha. 2022. A hybrid CNN-LSTM model for predicting server load in cloud computing. *The Journal of Supercomputing* 78 (2022), 1–30.

[39] Damiano Perri, Marco Simonetti, Sergio Tasso, Federico Ragni, and Osvaldo Gervasi. 2021. Implementing a Scalable and Elastic Computing Environment Based on Cloud Containers. In *Computational Science and Its Applications – ICCSA 2021*, Osvaldo Gervasi, Beniamino Murgante, Sanjay Misra, Chiara Garau, Ivan Blečić, David Taniar, Bernady O. Apduhan, Ana Maria A. C. Rocha, Eufemia Tarantino, and Carmelo Maria Torre (Eds.). Springer International Publishing, Cham, 676–689.

[40] Vladimir Podolskiy, Anshul Jindal, and Michael Gerndt. 2018. IaaS Reactive Autoscaling Performance Challenges. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. 954–957.

[41] E.G. Radhika and G. Sudha Sadasivam. 2021. A review on prediction based autoscaling techniques for heterogeneous applications in cloud environment. *Materials Today: Proceedings* 45 (2021), 2793–2800.

[42] Krzysztof Rzadca, Pawel Findeisen, Jacek Swiderski, Przemyslaw Zych, Przemyslaw Broniek, Jarek Kusmierek, Pawel Nowak, Beata Strack, Piotr Witusowski, Steven Hand, et al. 2020. Autopilot: workload autoscaling at google. In *Proceedings of the Fifteenth European Conference on Computer Systems*. 1–16.

[43] P. Singh, P. Gupta, and K. Jyoti. 2018. Tasm: Technocrat ARIMA and SVR Model for Workload Prediction of Web Applications in Cloud. *Cluster Computing* 22, 2 (2018), 619–633.

[44] Parminder Singh, Pooja Gupta, and Kiran Jyoti. 2019. TASM: technocrat ARIMA and SVR model for workload prediction of web applications in cloud. *Cluster Computing* 22, 2 (1 Jun 2019), 619–633.

[45] Emmanuel Tachu. 2022. A quantitative study of the relationship between cloud flexibility and on-premise flexibility. *Issues in Information Systems* 23, 1 (2022).

[46] R.S. Tsay and R. Chen. 2018. *Nonlinear Time Series Analysis*. Wiley.

[47] A. Ullah, J. Li, Y. Shen, et al. 2018. A control theoretical view of cloud elasticity: taxonomy, survey and challenges. *Cluster Computing* 21 (2018), 1735–1764.

[48] Fei Wang and LiGang Zhao. 2019. Complexity Analysis of Air Traffic Flow Based on Sample Entropy. In *Chinese Control And Decision Conference*. 5368–5371.

[49] Ang Xuan, Mengmeng Yin, Yupei Li, Xiyu Chen, and Zhenliang Ma. 2022. A comprehensive evaluation of statistical, machine learning and deep learning models for time series prediction. In *2022 7th International Conference on Data Science and Machine Learning Applications (CDMA)*. 55–60.

[50] Ming Yan, XiaoMeng Liang, ZhiHui Lu, Jie Wu, and Wei Zhang. 2021. HANSEL: Adaptive horizontal scaling of microservices using Bi-LSTM. *Applied Soft Computing* 105 (2021), 107216.

[51] Natalya Yezhkova. 2024. *Assessing the Scale of Workload Repatriation: Insights from IDC's Server and Storage Workloads Surveys, 1H23 and 2H23*. Technical Report US50903124. International Data Corporation (IDC). https://www.idc.com/getdoc.jsp?containerId=US50903124 IDC Survey Report.

[52] Nathalie Zeghmouli. 2025. The cloud computing dilemma for financial services institutions. *Journal of Securities Operations & Custody* 17, 2 (2025), 166–177.

[53] Minqi Zhou, Rong Zhang, Dadan Zeng, and Weining Qian. 2010. Services in the cloud computing era: A survey. In *2010 4th international universal communication symposium*. IEEE, 40–46.

[54] Ding Zou, Wei Lu, Zhibo Zhu, Xingyu Lu, Jun Zhou, Xiaojin Wang, Kangyu Liu, Kefan Wang, Renen Sun, and Haiqing Wang. 2024. OptScaler: A Collaborative Framework for Robust Autoscaling in the Cloud. *Proc. VLDB Endow.* 17, 12 (2024), 4090–4103.