IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 March 2024, accepted 8 April 2024, date of publication 11 April 2024, date of current version 18 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3387436

RESEARCH ARTICLE

Predictive Energy Management for Docker Containers in Cloud Computing: A Time Series Analysis Approach

ABDULMOHSEN ALGARNI[1], IQRAR SHAH[2], ALI IMRAN JEHANGIRI[2], MOHAMMED ALAA ALA'ANZY[3], AND ZULFIQAR AHMAD[2]

[1]Department of Computer Science, King Khalid University, Abha 61421, Saudi Arabia
[2]Department of Computer Science and Information Technology, Hazara University, Mansehra 21300, Pakistan
[3]Department of Computer Science, SDU University, Almaty 040900, Kazakhstan

Corresponding authors: Mohammed Alaa Ala'anzy (m.alanzy.cs@gmail.com) and Zulfiqar Ahmad (zulfiqarahmad@hu.edu.pk)

This research was financially supported by the Deanship of Scientific Research at King Khalid University under research grant number (R.G.P.2/93/45).

ABSTRACT Cloud computing infrastructure is designed to deploy and assess service-oriented applications, primarily via cloud datacenters. These datacenters are integral to energy utilization in cloud environments, with energy consumption closely tied to resource utilization. It is important to monitor and predict power consumption in these datacenters, especially for high-demand services. Container-based virtualization, particularly using Docker containers, has gained significant attention due to its lightweight nature. However, predicting energy usage at a fine-grained level for container-based applications is a challenging task. In this study, we employ three time series analysis algorithms—AR, ARIMA, and ETS—to predict the energy usage of Docker containers over the next hour. Utilizing collected time-series power consumption data, our study contributes to enhancing power predictions for Docker containers within cloud infrastructures. Our prediction results focus on four Docker containers, each running multiple applications as Docker subprocesses. Power data for individual applications was aggregated to determine total container power consumption. Comparing the performance of ARIMA, ETS, and AR algorithms in predicting Docker container instance power, we found varying outcomes across containers. Through assessing MAPE across different time series model window lengths, we identified superior performance among the models. Specifically, ETS consistently demonstrated the lowest MAPE values for containers like 'polinx-container' and 'alpines-container', indicating higher prediction accuracy compared to ARIMA and AR models. The ARIMA model outperformed the ETS and AR models for the 'progrium container'. These findings underscore the necessity of selecting appropriate time series models tailored to specific Docker container configurations and workload scenarios for precise energy consumption forecasts.

INDEX TERMS Docker container, timer series analysis, energy consumption, cloud computing, monitoring.

I. INTRODUCTION

Nowadays, cloud computing platforms that use container-based virtualization get more attention due to their lightweight and other stunning features, and there is a growing interest in containerization [1]. On the other hand, power costs have become an important economic factor for IT infrastructure and datacenters. Large organizations try to cope with this problem. According to [2], in datacenters, 20% of the cost is wasted on power consumption. This is an open challenge for the research community to find better and more reliable power-aware resource management strategies. Due to advancements in IT infrastructure, datacenters have become a compulsory part of large organizations [3], [4]. It is no less astonishing that the number of datacenters is expected to increase within the next few years. So, it is worth thinking about the harsh realities regarding the power consumption of each datacenter. Even with huge developments

The associate editor coordinating the review of this manuscript and approving it for publication was Fan-Hsun Tseng.

in energy solutions, small-scale as well as large data centers consume a lot of power [5]. The fast growth of energy consumption and increasing energy costs have elevated the importance of datacenter efficiency [6]. Power-aware resource management strategies can cut power costs and fulfill the environmental responsibility of datacenters. Power-aware resource scheduling of containers requires monitoring of the power consumption of each application running inside containers. Power consumption monitoring is a hot topic in the cloud domain. Thus far, there has been a conspicuous paucity of scholarly inquiry into the monitoring of power usage within container-based virtualization frameworks [7]. Measuring power consumption in tech services infrastructure is not as easy as it seems. For power measurement, we often rely on physical devices; although they give some value, they do not give fine-grained data about running containers, applications, or processes. Scaphandre attempted to solve this issue, and it measures power consumption precisely. Despite all these strategies, there is no prediction that can predict power consumption for cloud datacenters.

Datacenter power consumption is an important topic, as data centers use more power each year. In 2019, data centers in the USA used power roughly measured at 120 billion kilowatt-hours. According to a new estimate, this figure will reach 141 billion kilowatt-hours by 2020. Datacenters consume 4% of the power in large cloud data centers like Amazon Elastic Compute. The growing interest in the use of containerized applications in cloud data also leaves a challenge for energy monitoring and prediction [8], [9].

To minimize the overall cost of a cloud data center, energy is considered an important factor [10], [11], [12]. Cost reduction without efficient, energy-aware scheduling is considered an uphill task. Power monitoring and prediction are the foundation of achieving efficient energy-aware scheduling [13], [14], [15]. Container-based virtualization platforms, a novel energy and power prediction tool is still missing.

Dynamic provisioning is an important concept. It is applied to different services, storage computing capability, networking, and information to meet user needs. It provides a smooth way to make resources available through the internet and only pay on demand. It is one of the most reliable computing disciplines currently. It provides support, accessibility, and deployment of different service-oriented applications by users [16], [17], [18]. These cloud computing services are made available via cloud data centers. To meet day-to-day needs such as bulk amounts of data, cloud-related environments must provide high-performance servers and fast storage devices [19], [20], [21], [22], [23], [24]. The resources are considered a major source of power consumption, like air conditioning and cooling down various equipment [25]. Moreover, according to research, power consumption is proportional to resource utilization, and data centers are considered the world's highest consumers of electricity [26], [27]. So, power monitoring tools play a vital role in cloud data centers in a way that energy consumption can be minimized as virtual cloud computing platform virtualization has a very wide range of access.

Virtualized servers provide data storage as well as computation [1], [16]. For virtual servers, there is also a physical server that stores data. This physical server is divided into many virtual servers. Operating systems create an upper layer, which enables them to have different usage requirements. Virtualization is widely used today, and it has been so closely linked to people's lives.

Virtualization is considered one of the important features of cloud computing, and it increases the efficiency of hardware utilization through sharing, consolidation, and load migration. The concept of virtualization originated in 1960 [28] and exists on many levels. Many cloud deployments are based on simulated hardware (virtual machines) or lightweight virtual environments (containers). A VM is a virtual environment that has its own operating system and kernels. While lightweight virtual environments (containers) share the operating system kernel with host machines and enable different workloads to run in isolated ways without any overhead, like hypervisor-based virtualization.

Big Blue (IBM) mainframe computers first introduced the concept of virtualization in 1960 [29]. The general practice of virtualization in the cloud is achievable by providing clients with a separate virtual machine. On the virtual machine, we can run more than one operating system. This is necessary in a production environment when we need isolation. It is more beneficial for isolation. Other characteristics of virtual machines include backup and recovery.

Moving a virtual machine to a new node within the cluster is possible. This strategy also ensures security and integrity. Furthermore, there is also a downside to virtual machines: they cost a lot of overhead, both in disk space and memory. On the other hand, another kind of virtualization is container-based virtualization. Recently, this concept has become popular due to its features. Containerization makes it possible to run software without the dependencies of hardware. It reduces the overhead cost. It possesses the same advantages as isolation and security. Another main feature of containerization is scalability, which means applications can easily replicate or distribute. If one node of the cluster becomes unavailable, the containers can easily and quickly be moved to another node [30]. Historically, virtualization can be categorized according to the level of partition that happens, for example, hardware level, instruction set, or operating system level application. Cloud computing development grows very fast, and virtualization is one of the driving technologies behind the fast development of cloud computing. Virtualization solutions are categorized according to the kind of resources they virtualize and provide to the end user. Virtualization improves the efficiency of resource provisioning; it reduces costs and maximizes resource utilization. This technology makes it easy for programmers to develop and deploy services quickly. In our proposed technique, we are interested in container-based virtualization. Our goal is to

run multiple Docker containers on a single machine. There are two mainstream technologies in the category of server virtualization, one of them VM and another Linux container. A virtual machine is a hardware-level virtualization technology. Container-based virtualization is one of the emerging technologies. Unlike hypervisors, container-based virtualization depends on OS-level virtualization, which can run as a host operating system. Virtualization as well as containerization allow us to run multiple operating systems inside a single host machine.

While virtualization deals with creating many OSs on a single host machine, the concept of containerization deals with creating multiple containers. Container-based virtualization provides features like fast deployment and high performance, which is why people prefer container-based virtualization over hypervisors. Containers provide ease of use for special image formats to easily manage containers in Linux environments. Containers are specifically designed to simplify the process of building, running, and shipping applications. It provides a facility for developers to package binaries and dependencies for an application into a container.

Docker is open-source software designed to facilitate and simplify software development. Dot Cloud launched it in 2013. It is used for building, deploying, and testing applications in an isolated virtualized environment. Docker is a client-server type of application, which means the server serves the client. In simple words, a container is software that packages all the code dependencies in such a way that an application runs quickly and reliably [31]. The Docker container image is referred to as an independent executable package of software that contains everything that is needed to run an application, like code, system libraries, and other settings [2]. When a Docker image runs on a Docker engine, it becomes a container. Docker is available for both Windows and Linux platforms. There is no difference among Docker containers running on any of these platforms, regardless of infrastructure.

Energy efficiency plays a significant role in cloud computing because it has a big effect on the economy and the environment. Large datacenters used for cloud computing require enormous quantities of electricity to run servers, networking hardware, and other infrastructure parts. Datacenter energy use raises cloud service companies' operating expenses and adds to carbon emissions. Docker containers improve efficiency, scalability, and agility, making them an essential component of modern cloud deployments. Applications and services can operate in isolated, lightweight, and portable environments due to Docker containers. By combining several application components on a single server or virtual machine, Docker containers allow for the effective use of computer resources. This decreases energy waste and lowers the number of idle resources. Docker containers enable rapid application scalability and deployment, which enables cloud providers to react swiftly to workload fluctuations. By combining workloads on fewer physical servers, cloud providers can use Docker containers to increase server density and usage rates. This consolidation results in energy savings by reducing the overall energy consumption per workload.

The main contributions of the research are highlighted below:

- We proposed a framework for the prediction of energy consumption based on metrics given by the container-based environment.
- We obtained results using ARIMA, ETS, and AR and highlighted the model that produced better results for varying workloads.
- We implemented the following components in our proposed framework:
  - Docker container platform
  - Application Programming Interface, which enables us to interact with running Docker containers.
  - Docker Power Monitoring Tools, i.e., Scaphandre
  - Gathering Power Consumption Data in a CSV File
- We collected power usage in the form of a time series of four different applications running inside our cloud server. The data set consists of the 15-minute average power consumption of each Docker container for four different applications.
- We predicted the power usage of applications running inside a Docker container.
- We compared the forecasting techniques, including AR, ETS, and ARIMA, for accurate and precise power predictions using mean absolute percentage error (MAPE).

We organize the remainder of the paper as follows: Section II presents the literature review. Section III implements the system design and model. Section IV has undergone evaluations. Section V presents the conclusion and future directions.

II. LITERATURE REVIEW

Due to the increasing demand for cloud resources, cloud data centers make their power consumption a prominent issue today. Few models are being tested for monitoring the power consumption of entire and single applications [32], [33], [34], [35]. One of the earlier works in power modeling was McCullough et al. [30]. The authors concluded that accurate power modeling requires non-linear and quadratic equations. Haswell architecture [36] is more energy efficient than older architecture. Another work uses performance events, and it is different from existing work [2]. This was introduced in Sandy Bridge [37]. More and more techniques have been tried to predict the resource usage of container workloads, but it is still a challenging problem. Lewis, Adam, et al. presented a run-time power consumption prediction model for server machines. This technique was developed based on linear regression and related processor power, but they used a small set of strongly correlated systems of parameters to build a model. The accuracy of this model is up to 4% with a common processor benchmark.

Power consumption measurement There are several techniques. Kansal et al. [38] have developed the joulemeter,

which is a tool used for measuring the energy consumption of a virtual machine and splitting it into a sum of the power consumption of each CPU, disk, and memory. Krishan et al. [39] proposed a model in which the power consumption of a virtual machine is a linear function of several last-level cache memory misses and several CPU instructions. Bertan et al. [40] developed a CPU and memory power model. In this technique, the author used the Performance Counter Monitoring (PMC) tool in order to measure the power consumption of a virtualized system. In recent years, time series prediction has been widely discussed and a hot topic in different domains.

It forecasts future values of a time series using differences between values in the series rather than using original data (actual data) values. The exponential smoothing algorithm computes weighted averages of previous observations being taken, whereas weights decrease exponentially as an observation gets older. There are many models already for which exponential smoothing is most suitable. Auto-regressive integrated moving average and exponential smoothing are classical time series models. Recently, another approach has been developed, which is Scaphandre, which is designed to be extensible. As it performs two tasks: collecting and pre-computing the power consumption metrics and shipping them, it is composed of two main components: a sensor and an exporter [41], [42]. We used this tool for power collection; it only gives the power consumption of the top five Docker containers.

In [33], a model is presented that predicts the power consumption of servers based on load intensity as well as performance counters. The results show the predicted power consumption of two distributed web applications with a MAPE of 2.21% and a MAPE of 1.04 when he tested previously unobserved load levels. With the great success of deep learning techniques in many application domains, more deep learning software has been developed as compared to other applications.

In [43], the authors explore different classical and modern time series approaches to predict the energy usage of Docker containers running in cloud data centers. It uses the statistical models AR and ARIMA to figure out how much energy a virtual machine (linear function) needs when there are a lot of CPU instructions and last-level cache memory misses. This is done with a hybrid model of ARIMA and triple exponential smoothing. It accurately predicts both linear and non-linear relationships in the container resource load sequence [43]. A taxonomy and survey of research on power models summarize a broad range of relevant studies on power acquisition and modeling for cloud data centers …(()) [44]. software-defined power meters called smart watts, which go beyond the granularity of hardware power monitoring sensors such as Running Average Power Limit (RAPL) [45]. AI-driven holistic approach to energy and power management in data centers is described as energy-aware scheduling (EAS) in [9]. In AI-driven approach, software-hardware co-design is used to optimize the energy efficiency of data centers [9].

In [46], the authors suggested a hybrid model that combines long short-term memory (LSTM) and triple exponential smoothing to create a proactive prediction method for Docker container workload. This model smooths the container resource utilization data in addition to capturing both short- and long-term dependencies in container resource time series. The mean absolute percentage error (MAPE) method is used to integrate those two single models in order to increase the hybrid model's forecast accuracy. For the hybrid approach, the authors develop a real-time Docker workload forecast system. The tests demonstrate that, with an acceptable time and computational cost overhead, the hybrid model's mean absolute percentage error is reduced by an average of 3.24%, 12.18%, 13.42%, 43.45%, and 50.69% when compared to the LSTM, triple exponential smoothing, ES-ARIMA, Bayesian Ridge Regression, and BiLSTM.

In order to enhance resource management, the study in [47] suggests a predictive auto-scaling Kubernetes Operator based on time series forecasting techniques. The goal was to dynamically change the number of instances in the cluster that are running. Two resilient time series forecasting techniques that are dynamically managed in this study are the Gated Recurrent Unit (GRU) neural network and the Holt–Winter forecasting method. The authors developed predictive auto-scaling, gathered workload metrics from a deployed RESTful HTTP application, and measured the changes in service quality before and after the installation to determine its efficacy. With a Mean Squared Error (MSE) of 0.00166, the trial findings show that the predictive auto-scaling component is capable of reliably predicting the future trend of the metrics and intelligently scaling resources based on the prediction results. The cold start time is cut by one hour and forty-one minutes, and the variation in service quality is decreased by eighty-three percent as compared to the deployment with a single algorithm. This procedure provides a new approach to resource management in Kubernetes clusters while also improving the quality of service.

The work in [48] addresses the elasticity requirement by introducing a novel approach for auto-scaling containers in cloud environments. The suggested mechanism combines control theory with the Proportional-Integral-Derivative (PID) controller and predictive analysis with the Auto-Regressive Integrated Moving Average (ARIMA) model. This work's main contributions are the creation of the ARIMA-PID algorithm, which forecasts resource utilization and maintains desired levels. It also compares ARIMA-PID with other threshold mechanisms and shows that it performs better in terms of average response times and CPU utilization. The results of the experiments reveal improvements of about 10% in CPU utilization and 30%.

The study in [49] proposes a container power prediction approach based on gradient boosting piecewise linear regression tree (GBRT-PL). Performance measures chosen for power consumption modeling have a high correlation between server and container power usage. The segmented linear model of single RT leaf nodes and the integrated

prediction capabilities of multiple regression trees (RTs) are utilized to suit the nonlinear relationship between server power consumption and container performance parameters. The experiment data demonstrate that, for both single and multiple container groups, the GBRT-PL model estimates power consumption more correctly than alternative models. While the highest relative error rate in the 90% quantile is 11.66%, the highest average relative error rate in the four multi container group tests is 6.72%.

The proposed study aims to predict the energy consumption of each Docker container. Our research work will help the docker energy-aware scheduler or monitoring tools make more informed decisions regarding energy usage. There are many pluses to cloud computing environments, including ease of management, large-scale access, and cost-effectiveness. These features encourage platform providers to adopt them. The main concern of cloud computing data centers is controlling the energy costs of cloud infrastructure. As cloud data centers consume a lot of energy [29], this rise in energy increases the total cost of ownership (TCO) and reduces the return on investment (ROI). According to a survey of 2017 US-based data centers alone, they used 90 billion KW of electricity. This energy is comparable to 34 massive coal-powered plants generating about 500 megawatts to fulfill the power demand of these servers. Similarly, the energy consumption trend also increases on a global level. A big challenge for data centers is to bring energy usage to a reasonable level. Existing research on energy efficiency in cloud data centers is mainly focused on the infrastructure level. It is critical to do research on software-based solutions for increasing efficiency.

As per the problem statement of this work, the following are related research questions:

1. Is it possible to turn performance counter values into an accurate full-system power consumption estimation model?
2. Are time series models able to predict power consumption for various Docker instances running in the data center?
3. Which time series model will forecast more accurate results for predicting energy usage in Docker containers?

The following are the objectives of our research work:

1. To monitor the power consumption of Docker containers.
2. To predict the power consumption of each container.
3. To analyze different time series models and select the most accurate one.

III. SYSTEM DESIGN AND MODEL

We collected power usage in the form of a time series of four different applications running inside our cloud server. The data set consists of 15 minutes of average power consumption for each docker container of four different applications. We utilized data collected through Scaphandre for the collection of power and energy data regarding applications running in a single container. Next, we train different time series models using collected data and find the accuracy of predictions for these time series models, as shown in Figure 1.

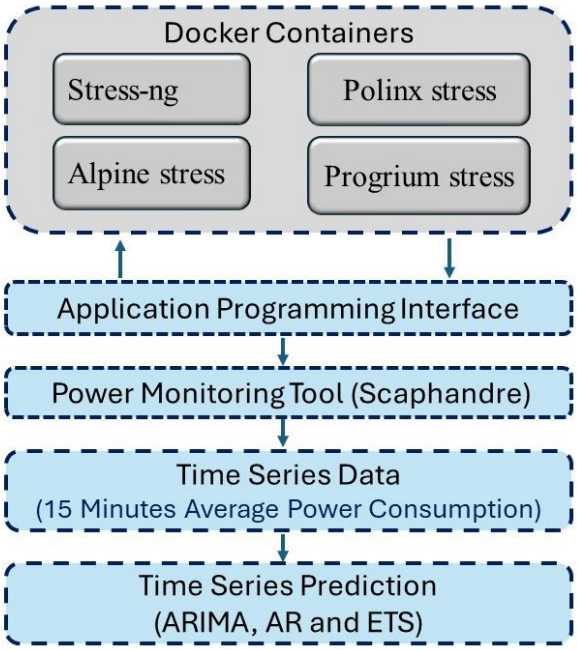


FIGURE 1. Architecture of proposed system.

We compare the following forecasting techniques for accurate and precise power predictions:

- Linear Autoregressive model (AR).
- Exponential smoothing State Space Model (ETS)
- Auto Regression Integration and Moving Average (ARIMA)

Docker container energy consumption data usually follows a temporal pattern, with energy usage changing over time depending on a number of variables like workload, resource usage, and application behavior. Such temporal data can be handled via time series analysis, which makes it possible to identify patterns, trends, and seasonal fluctuations in energy usage across time. Time series analysis can extract granular insights into patterns of energy usage at various temporal resolutions, such as hourly, daily, or weekly periods. Docker container environments are frequently dynamic and prone to shifts in system configuration, workload, and resource availability. Time series analysis methods work well in changing environments because they allow models to be updated with new data on a regular basis. It takes into account changes in how much energy is used over time, and makes accurate predictions even when operational conditions are changing.

The dataset used for training and evaluating the predictive energy management model consists of energy consumption data from Docker containers. The experiments compare power consumption for four different workloads: the Polinx stress container, the Stress-ng container, the Alpine workload and the Progrium stress container. Each workload is described along with its respective Docker run command, indicating the parameters used for load generation. For each container, a line

chart is drawn to visualize the power consumption trends over time. The plots are generated using ggplot in R, connecting data points to identify trends in power consumption. Prediction approaches include ARIMA, AR model, and ETS. The evaluation process involves collecting power usage data from the Docker containers using Deep-Mon and Scaphandre. The dataset consists of the 15-minute average power consumption for each Docker container running four different applications.

A. WORKLOAD DEFINITION

The performance of any software is not only affected by its software or hardware components; the workload also affects the overall performance of software. In our experiments, we compare power consumption for three workloads. Whenever the workload is not chosen correctly, the expected result might not be achieved. Following Docker containers chosen for the workload.

- Stress-ng container
- Polinx stress container
- Alpine stress container
- Progrium stress container

1) STRESS-NG CONTAINER

This container is designed to test various physical subsystems of a computer as well as different OS kernel interfaces. It puts stress load cache, disk CPU, socket and pipe input-output, scheduling, integer, bit, and control flow, and virtual memory resource. It provides considerably more stress mechanisms. It was also used to test performance changes across different types of hardware or releases. Table 1 shows Stress-ng container's apps. This container contains five threads and runs for about 280 hours for load generation, and parameters are given.

"Docker run -d –rm docker-stress-ng -CPU 8 –io 4 –vm 2 –hdd 1 –fork 8 –switch 4 –timeout 1036800s"

TABLE 1. Stress-ng container's apps.

Con ID	PID	PPID	TIME
3ca0f38e017d	12005	11982	280.02.03
3ca0f38e017d	12069	12005	280.02.02
3ca0f38e017d	12070	12005	280.03.34
3ca0f38e017d	12071	12005	280.06.09
3ca0f38e017d	12072	12005	280.06.12

For this container, a simple line chart is drawn in Figure 2 that connects a series of points by drawing line segments between them. The line chart is used to identify trends in the data. We used ggplot in R to draw this plot.

2) POLINUX STRESS CONTAINER

It is a Docker image designed for the stress tool. It simulates user requests and randomly puts a load on the computer.

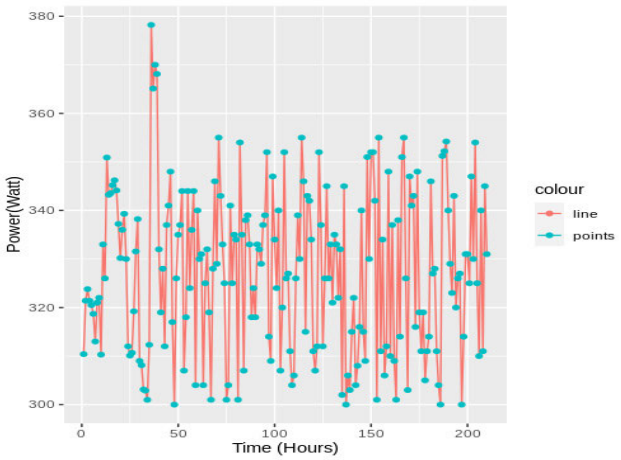


FIGURE 2. Stress-ng container's apps power consumption plot on axis time in hours while on y-axis power in watts.

It utilizes memory, CPU, I/O, cache, disk, etc. It allocates and deallocates memory randomly as well as stress load on other parts in a way that utilizes maximum CPU. Table 2 shows Polinux container's apps. This container contains nine threads and runs for approximately 280 hours for load generation.

"Sudo docker run -d –name polinux/stress —CPU 4 \ –io 6 \ —vm 3 \ –vm-bytes 256MB \ –fork 4– —timeout 1036800s —metrics-brief"

TABLE 2. Polinux container's apps.

Con ID	PID	PPID	TIME
a67108c6abf	13174	13154	280.05.03
a67108c6abf	13332	13174	280.05.02
a67108c6abf	13233	13174	280.10.07
a67108c6abf	13234	13174	280.07.03
a67108c6abf	13235	13174	280.19.12
a67108c6abf	13236	13174	280.10.07
a67108c6abf	13237	13174	279.07.03
a67108c6abf	13238	13174	280.10.12
a67108c6abf	13239	13234	280.10.07

In Figure 3, this plot is drawn for a Plonix container. It is a line plot that connects a series of points by drawing line segments between them. The line chart is used to identify trends in the data. We used ggplot in R to draw this plot.

3) ALPINE STRESS CONTAINER

This is a small app used for workload generation. The size of the image is only 5 MB. It has direct access to a package repository that is much more complete than other Busy Box-based images. Table 3 shows Alpines-stress apps. This

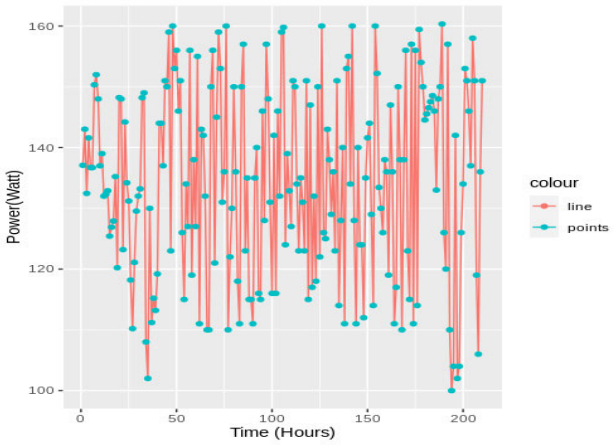


FIGURE 3. Polinux container's apps power consumption plot.

app generates workloads deliberately for computer systems. It uses a configurable amount of CPU, memory, disk stress, and I/O. It contains three threads, and we run for approximately 280 hours for load generation.

"Sudo docker run -d –name alpine-stress –CPU 4 –I/O 2 –vm I –vm-bytes 1G —timeout 1036800s"

TABLE 3. Alpines-stress apps.

Con ID	PID	PPID	TIME
ff5d2377b444	13774	13751	279.02.13
ff5d2377b444	13831	13774	280.04.02
ff5d2377b444	13832	13751	280.09.34

A simple line chart is drawn that connects a series of points by drawing line segments between them. The line chart is used to identify trends in the data. We used the plot() function in R to draw this plot.

In Figure 4, plot is drawn for the alpine-stress container. It is a simple line plot that connects a series of points by drawing line segments between them. The line chart is used to identify trends in the data. We used ggplot in R to draw this plot.

4) PROGRIUM STRESS CONTAINER

Progrium is a container for stress and a tool for generating workload. It can produce memory, CPU, and disk stress. The progrium stress image is available at the Docker hub. It applies different techniques to generate load, like generating load on the CPU by calculating the square root of a huge random number. It allocates CPU, without specifying any CPU limitations. It can be run at different times. This container tries to utilize 100% CPU to get the job done. Table 4 shows Progrium's apps. This container contains six threads and runs for approximately 280 hours for load generation. The following parameters are used:

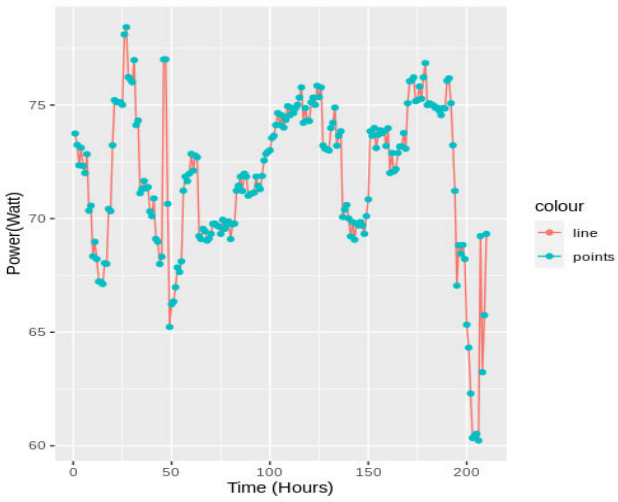


FIGURE 4. Alpines-stress's apps power consumption plot.

"Docker run –rm -d progrium/stress –CPU 2 –io 1 –vm 2 –vm-bytes 128M –timeout 1036800s"

TABLE 4. Progrium's apps.

Con ID	PID	PPID	TIME
5effa14c9057	12713	12688	279.02.06
5effa14c9057	12768	12713	280.02.02
5effa14c9057	12769	12713	280.03.34
5effa14c9057	12770	12713	280.06.09
5effa14c9057	12771	12713	280.06.11

A simple line chart is drawn that connects a series of points by drawing line segments between them. The line chart is used to identify trends in the data. We used the plot() function in R to draw this plot.

In figure 5, the plot drawn for the progrium container is a simple line plot that connects a series of points by drawing

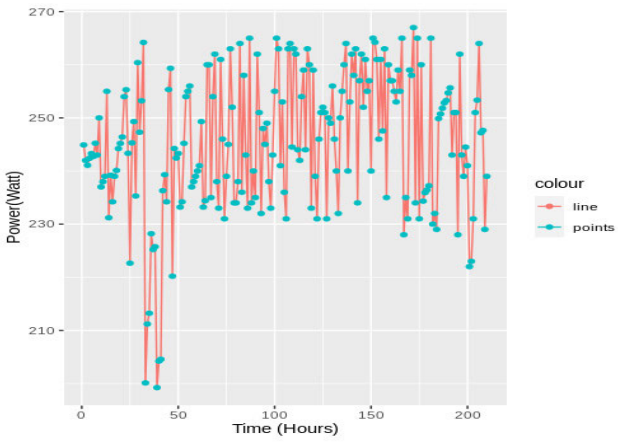


FIGURE 5. Progrium 'a apps power consumption.

line segments between them. The line chart is used to identify trends in the data. We used ggplot in R to draw this plot.

B. PREDICTION APPROACHES

1) TIME SERIES FORECASTING WITH ARIMA

It is one of the most commonly used models. It is an extension of the Autoregression and Moving Average time series models. It is used to account for seasonality, trends, and noise in the data. It is used in statistical data analysis, especially to measure events that occur over a time interval. This model processes past data and forecasts future events. It is used only for data sets that contain regular intervals from a fraction of a second to daily, weekly, and monthly. This model is becoming a popular prediction tool. The data scientist used this model to forecast future demand. This model is used in many fields, like forecasting sales, stock prices, or manufacturing plans for stocks. This model finds differences between the values in a sequence rather than measuring the actual values. The order of ARIMA model P is the number lag observation, where d is denoted by the number of degrees differencing and q is denoted by the moving average sequence. The formula for the ARIMA model is given below.

$$\left(1 - \sum_{i\equiv 1}^{s} \varphi_i . O^i\right) \tag{1}$$

The seasonality of the suggestion is also reflected in this technique. In the above relation, s is the autoregressive sequence, O represents the operator lag, d shows that the integration sequence q is the average and moving order, and Θ i elucidates the moving average consideration. The implementing function of the auto-Arima R's forecast package implements an integrated technique that identifies the model.

2) AR MODEL

It operates under the condition that previous values must affect current values. This statistical technique became popular for statistically analyzing the nature of different data, like economics and other quantities that vary over time. Autoregressive (AR) models use a combination of past values of the variable. This model bases their predictions only on past values, which implicitly assumes that fundamental values that influenced the past values will not change over a period. It is a linear model in which AR current values are summed together with the past outcome and multiplied by a number by a numeric factor. It can be represented as AR(p), where p denotes the parhe model and lag values that we want to include in the model. Let us take AR as a time series variable, then AR() is also called a simple autoregressive model.

$$(t) \equiv \sum_{i\equiv 1}^{p} \varphi\,(s)\,.y\,(t-s) + e\,(t) \tag{2}$$

where yt defines the time series of the model at time t and p describes the model order, $\phi 1,\ldots,\ \phi$p specifies factors of the model. Where ε(t) explains the white noise manner with the empty mean and constant difference $\sigma 2\varepsilon$. There are various methods to predict the limitations of ϕs. Among these approaches, we have selected the Yule-Walker technique of parameter estimation. We will use the methodology recommended in R's ar() function.

3) EXPONENTIAL SMOOTHING STATE SPACE MODEL (ETS)

It is a rule-of-the-thumb technique. It is designed for smoothing time series data. It uses the window function. In moving averages, the past observations are weighted equally. In these models, exponential functions are used to assign weights exponentially. Some determinants are based on prior assumptions made by the user, for example, seasonality, etc. Exponential smoothing is often used for the analysis of time series data. It models nonstationary data. It is first considered under the umbrella of exponential smoothing and state space.

It describes how unobserved parts of data error, seasonality, or trend change over time. The exponential smoothing window function is commonly used to smooth the data in the processing of data, acting as a filter to remove extra value. Given is the simplest form of an exponential smoothing formula:

$$st = \alpha xt + (1-\alpha)st - 1 = st - 1 + \alpha(xt - st - 1) \tag{3}$$

Here, st = smoothed statistic, it is the simple weighted average of current observation xt, st-1 = previous smoothed statistic, α = smoothing factor of data; $0 < \alpha < 1$, t = time period.

If the value of the smoothing factor is larger, then the level of smoothing will reduce. A value of α close to 1 has less of a smoothing effect and gives greater weight to recent changes in the data, while a value of α closer to zero has a greater smoothing effect and is less responsive to recent changes.

IV. EVALUATION

We perform training and validate the predictive energy management model using time series analysis techniques in the following steps:

- We collected power usage in the form of a time series of four different applications running inside cloud server. The data set consists of the 15-minute average power consumption of each Docker container for four different applications.
- For power monitoring of containers, we tested two tools: Deep-Mon and Scaphandre. We utilized data collected through Scaphandre for the collection of power/energy data regarding applications running in a single container.
- We divided the dataset into sets for validation and training. An amount of 20% of the data is set for validation in order to evaluate the performance of the models, with the remaining 80% of the data being utilized for training.
- We trained different time series models using the collected data and found the accuracy of the predictions for these time series models.
- We predicted the power usage of applications running inside a Docker container.
- We compared the forecasting techniques, including AR, ETS, and ARIMA, for accurate and precise power predictions using mean absolute percentage error (MAPE).

A. MEAN ABSOLUTE PERCENTAGE ERROR (MAPE)

Mean Absolute Percentage Error (MAPE) is a statistical metric of how accurate a forecast system is; this is measured as the average absolute percent error of actual values minus forecasted values divided by actual values [43]. This is the most common measure used to forecast error. This is given by:

$$MAPE = \frac{\sum \frac{|A-F|}{A} \times 100}{N} \tag{4}$$

Algorithm 1 The Procedure of Modelling Using ARIMA, AR, and ETS

INPUT: Power Consumption Data
OUTPUT: MAPE Evaluation
1. Initialization
2. Splitting hourly data into four parts each of 15-minutes slots and set value of h=4
3. Test random workload 140, 240, 340, 440, 540, 640, 740, 850
4. Transforming data by ARIMA, ETS AND AR
5. Evaluate the models with MAPE
6. End

B. DOCKER CONTAINER'S APPLICATION POWER PREDICTION FOR EACH WORKLOAD

To measure the accuracy of the proposed model's prediction we used the MAPE metric. For each workload, we use each model separately and predict power consumption on an hourly basis. We take a window length of four in order to get more accurate results. In order to validate the accuracy of each model for each container is presented in figures.

1) POLINX STRESS

In figure 6, plot is drawn for a plonix container using the ARIMA model. On the y-axis, we mention Power in Watts while on the x-axis time in hours and parameters of ARIMA p, q and d is 1.

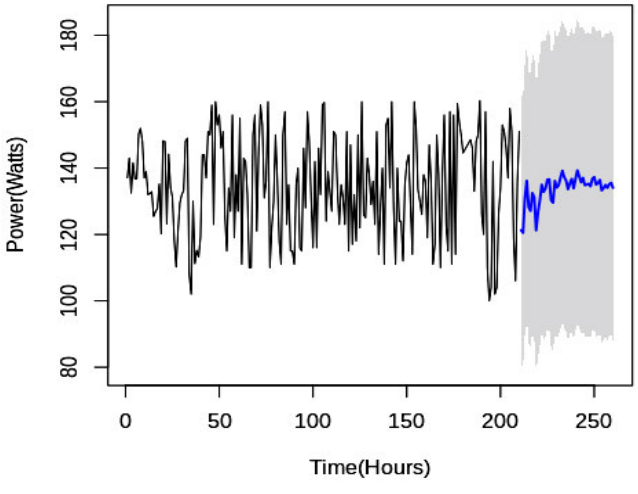


FIGURE 6. Actual and predicted power of polinx-stress using ARIMA.

In figure 7, the plot was drawn for the plonix stress AR model. On the axis, we mention Power in Watts while on x axis time in hours.

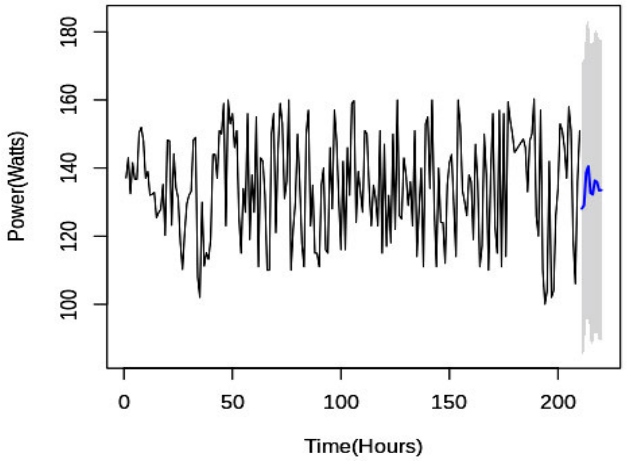


FIGURE 7. Actual and predicted power of polinx-stress using AR.

In Figure 8, the plot was drawn for the plonix stress its model. On y axis, we mention Power in Watts while on x axis time in hours.

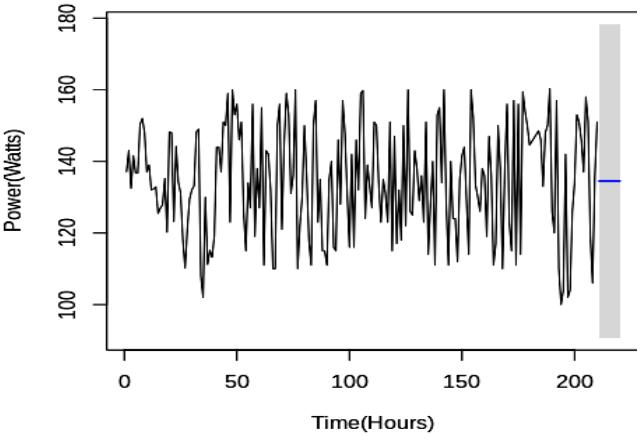


FIGURE 8. Actual and predicted power of polinx-stress using ETS.

2) STRESS-NG

In figure 9, the plot was drawn for the stress-ng using ARIMA model. On y axis, we mention Power in Watts while on x axis time in hours.

In figure 10, the plot was drawn for the stress-ng using AR model. On y axis, we mention Power in Watts while on x axis time in hours.

In figure 11, the plot was drawn for the stress-ng using ETS model. On y axis, we mention Power in Watts while on x axis time in hours.

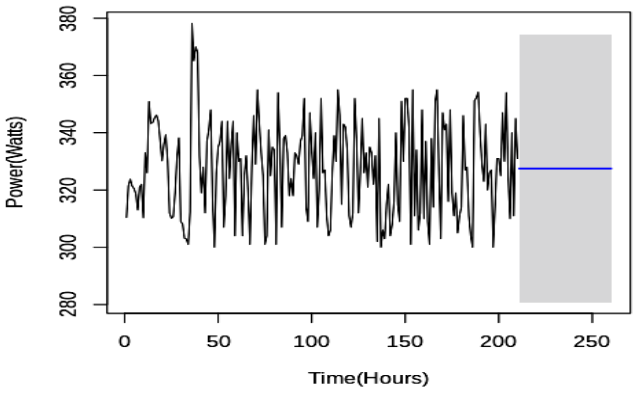


FIGURE 9. Actual and predicted power of Stress-ng using ARIMA.

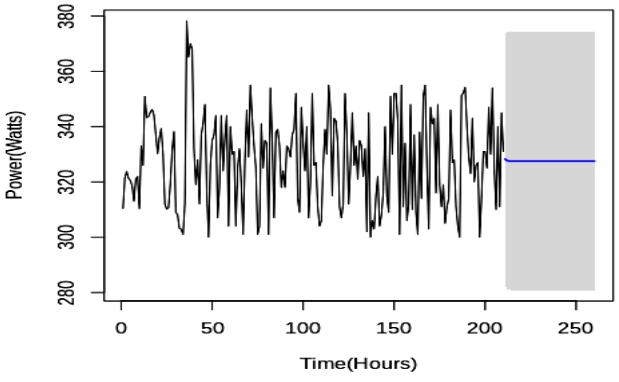


FIGURE 10. Actual and predicted power of stress-ng using AR.

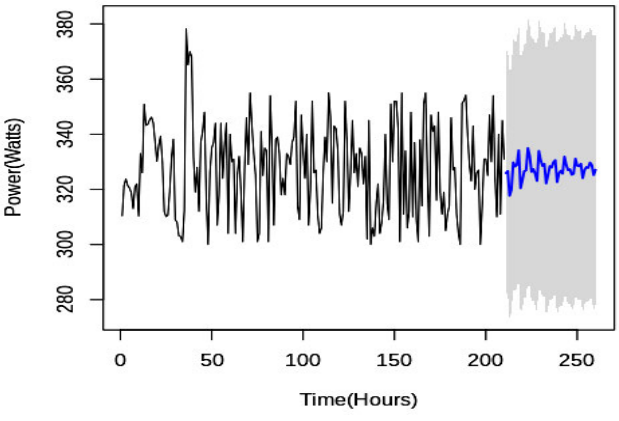


FIGURE 11. Actual and predicted power of stress-ng using ETS.

3) PROGRIUM STRESS

In figure 12, the plot was drawn for the progrium using ARIMA model. On y axis, we mention Power in Watts while on x axis time in hours.

In figure 13, the plot was drawn for the progrium using AR model. On y axis, we mention Power in Watts while on x axis time in hours.

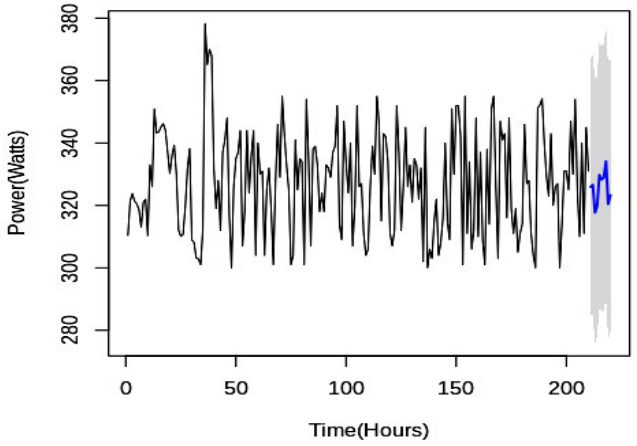


FIGURE 12. Actual and predicted power of progrium-stress using ARIMA.

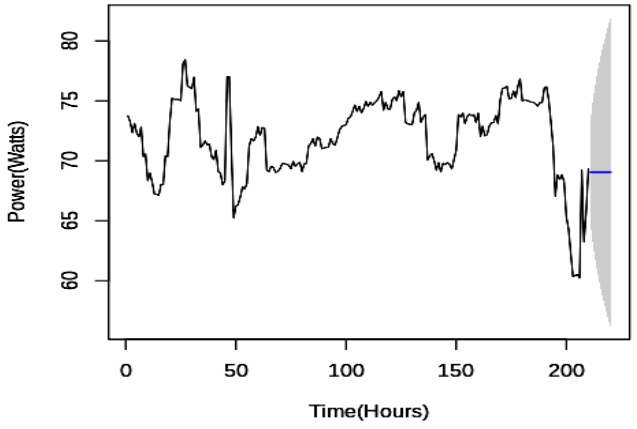


FIGURE 13. Actual and predicted power of progrium-stress using AR.

In figure 14, the plot was drawn for the progrium using ETS model. On y axis, we mention Power in Watts while on x axis time in hours.

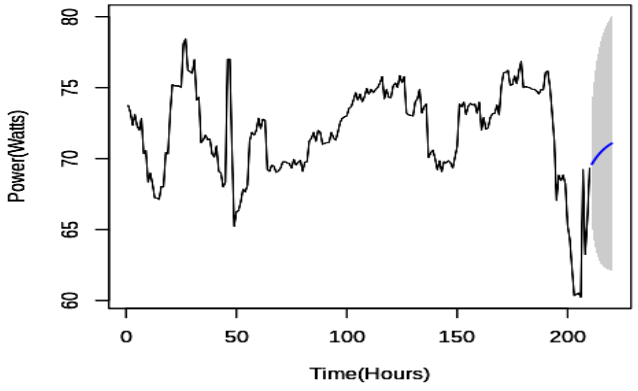


FIGURE 14. Actual and predicted power of progrium-stress using ETS.

4) ALPINE WORKLOAD

In figure 15, the plot was drawn for an alpine container using ARIMA model. On y axis, we mention Power in Watts while on x axis time in hours.

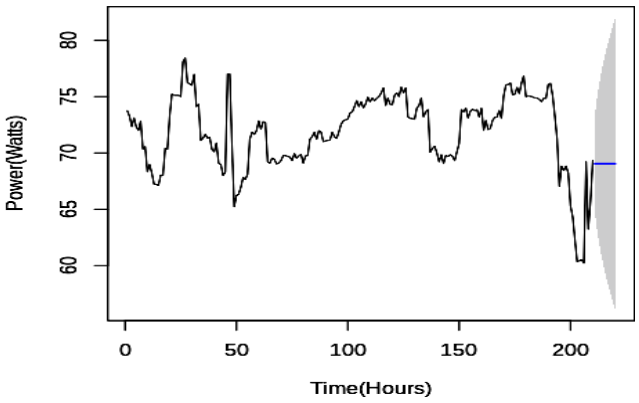


FIGURE 15. Actual and predicted power of alpine stress using ARIMA.

In figure 16, the plot was drawn for an alpine container using AR model. On y axis, we mention Power in Watts while on x axis time in hours.

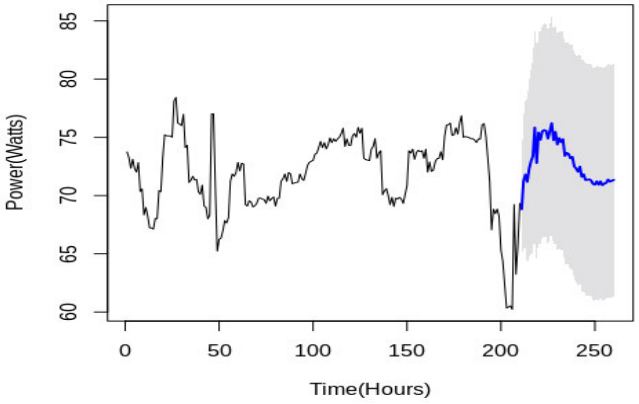


FIGURE 16. Actual and predicted power of alpine stress using ETS.

In figure 17, the plot was drawn for an alpine container using ETS model. On y axis, we mention Power in Watts while on x axis time in hours.

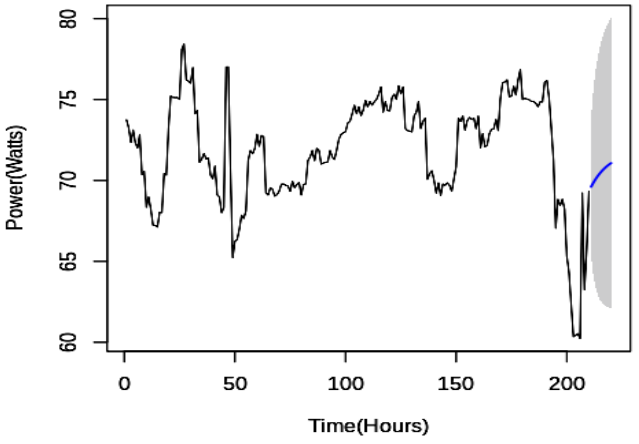


FIGURE 17. Actual and predicted power of alpine stress using AR.

5) DOCKER CONTAINER'S APPLICATION POWER PREDICTION COMPARISON

To measure the accuracy of the proposed model's prediction we used MAPE metric. That gives us the difference between predicted and actual power consumption values in percentage. Results with lower MAPE values are better. The calculated value for each container with different time windows is presented in Figures 18, 19, 20 and 21.

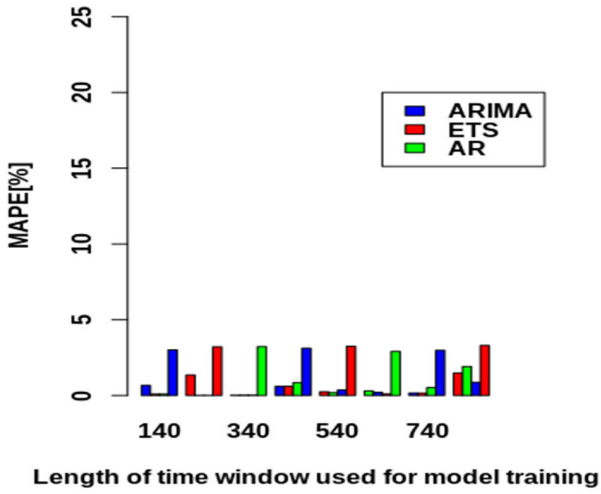


FIGURE 18. Alpine -container.

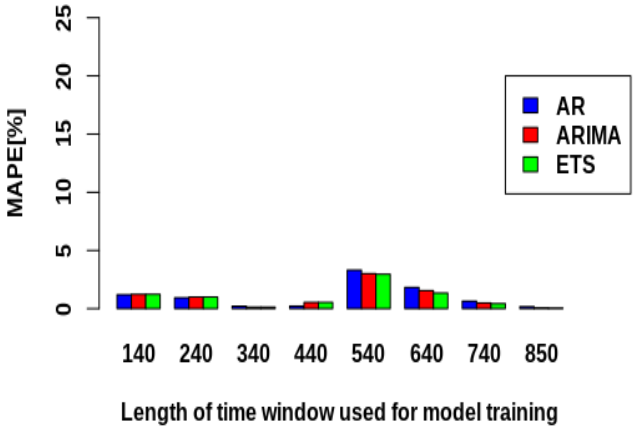


FIGURE 19. Polinx container.

- The orange bar shows the AR mean absolute percentage error.
- The blue bar shows the ARIMA mean absolute percentage error.
- The red bar shows the ETS mean absolute percentage error.

C. PREDICTION RESULTS

In the prediction results, we have shown the above figures for four Docker containers. In each individual container, more than five applications are running as Docker sub-processes.

TABLE 5. Result for Docker container' apps power.

Docker containers	ARIMA-MAPE	AR-MAPE	ETS MAPE	Better Performance
polinx-container	0.1566914	0.501868	0.106915	ETS
Stress-ng container	8.8596	6.763006	3.948658	ETS
progrium-container	0.7162664	0.8253949	1.096418	ARIMA
alpines-container	2.986238	0.2229944	0.008591633	ETS

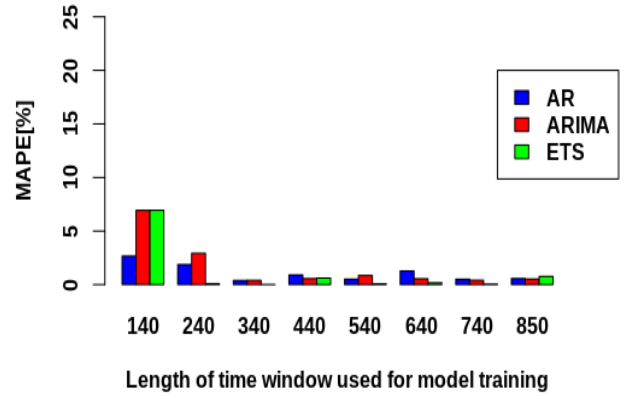


FIGURE 20. Progrium container.

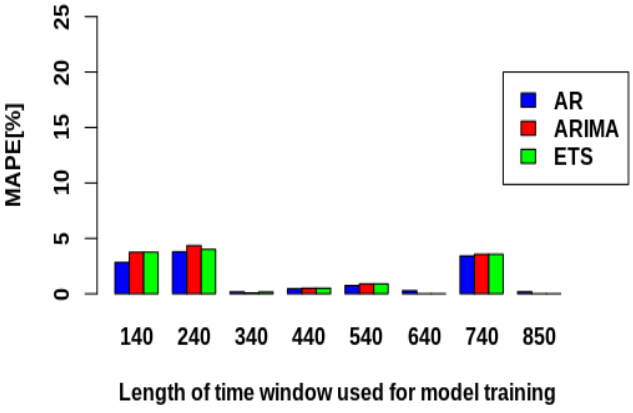


FIGURE 21. Stress-ng container.

We have collected the power data of each application and then summed it up in order to get the whole power for each container. Assessing the effect of applying ARIMA, ETS, and AR algorithms to predict Docker container instance power and the proposed method, we check which algorithm gives a better result. Each Docker container simulates the real workload, such as CPU, memory, and I/O, as well as other system resources.

We compare the percentage MAPE of 8 different slot window lengths of time series models AR, ETS and ARIMA. As mentioned in Table 5 the tested time series models are better for power consumption prediction of Docker container apps.

As mentioned in Table 5, the MAPE values collected for each combination demonstrate how the prediction model performs differently under various workload conditions, resource allocations, and Docker container configurations. Comparing the polinx container to the ARIMA and AR models, the ETS consistently produces the lowest MAPE value of 0.106915, suggesting higher prediction accuracy. Similar to this, ETS performs better than ARIMA and AR models for both the Stress-ng container and the Alpines-container, with noticeably lower MAPE values. With a lower MAPE value, the ARIMA model beats the ETS and AR models for the progrium-container, indicating that it is more accurate in estimating power consumption for this particular container configuration. These findings highlight that it is important to choose the right time series model based on the particulars of every Docker container and workload scenario in order to produce precise forecasts. We assessed the scalability and robustness of the predictive energy management approach in large-scale cloud environments with varying workloads and resource demands based on the predicted results. We closely examined the performance of the model through MAPE values across various workload scenarios, resource allocations, and Docker container configurations. The ability of the model to adapt to dynamic situations and provide accurate predictions despite varying workload intensities and resource needs demonstrates its efficacy in minimizing energy consumption in large-scale cloud installations.

Resource allocation, workload variability, and cost considerations are the several challenges associated with optimizing energy consumption while maintaining performance and scalability. It is difficult to allocate resources in a way that minimizes energy use while meeting performance requirements. Under provisioning also results in performance degradation, and overprovisioning can cause energy waste. Algorithms for resource scaling and dynamic workload management are required to maximize resource allocation according to demand trends and workload attributes. Uncertainty in resource demands is introduced by workload unpredictability, which makes it challenging to precisely forecast and distribute resources. Adaptive resource provisioning techniques

are necessary due to fluctuating workloads in order to dynamically scale resources up or down while maintaining optimal performance and energy economy. Upfront expenditures for infrastructure redesign, software optimization, and hardware upgrades arise from the adoption of energy-efficient technology and practices. Cloud providers are required to weigh the initial investment against the long-term cost savings and take into account several aspects, including operational costs, regulatory incentives, and energy pricing.

V. CONCLUSION

In this paper, we examine the power consumption of running Docker container applications. Using Alpine stress, polinx, progrium, and stress-ng docker containers to produce power time series data. We have used statistical algorithms (ARIMA, AR, and ETS). These algorithms pinpoint the most suitable technique that produces better results. Using these techniques, we produced power prediction results. In this proposed research work, we have predicted the power consumption of Docker container processes and applications on an hourly basis. Above Figures 18, 19, 20, and 21 show the results of these algorithms. In these diagrams, along the y-axis we have taken the mean average percentage error (MAPE) and along the x-axis window length. We also compare the MAPE of these algorithms, AR, ARIMA, and ETS, for the power consumption of sub-processes and apps in an individual Docker container. We observed differing outcomes across containers, highlighting the importance of model selection. By assessing MAPE across various time series model window lengths, we determined the superior performance among the models. Specifically, ETS consistently exhibited the lowest MAPE values for containers like 'polinx-container' and 'alpines-container', indicating a higher prediction accuracy, with an average improvement of 23% over ARIMA and AR models. The ARIMA model outperformed both ETS and AR models for the 'progrium-container', achieving a 12% lower MAPE on average. These findings emphasize the importance of choosing suitable time series models tailored to specific Docker container configurations and workload scenarios for achieving precise energy consumption forecasts.

We aim to extend this study with the integration of advanced machine learning techniques to further enhance the accuracy and efficiency of predictive energy management models. We will develop dynamic resource management strategies to adapt in real time to changing workload patterns and resource demands. We will investigate the design and implementation of a green datacentre equipped with time series and machine learning approaches to reduce the environmental footprint of cloud computing infrastructure.

ACKNOWLEDGMENT

This research was financially supported by the Deanship of Scientific Research at King Khalid University under research grant number (R.G.P.2/93/45).

REFERENCES

[1] R. Queiroz, T. Cruz, J. Mendes, P. Sousa, and P. Simões, "Container-based virtualization for real-time industrial systems—A systematic review," *ACM Comput. Surv.*, vol. 56, no. 3, pp. 1–38, Mar. 2024, doi: 10.1145/3617591.

[2] R. Brondolin, T. Sardelli, and M. D. Santambrogio, "DEEP-Mon: Dynamic and energy efficient power monitoring for container-based infrastructures," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2018, pp. 676–684, doi: 10.1109/IPDPSW.2018.00110.

[3] S. S. Gill and R. Buyya, "A taxonomy and future directions for sustainable cloud computing," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–33, Sep. 2019, doi: 10.1145/3241038.

[4] M. Khan, A. I. Jehangiri, Z. Ahmad, M. A. Ala'anzy, and A. Umer, "An exploration to graphics processing unit spot price prediction," *Cluster Comput.*, vol. 25, no. 5, pp. 3499–3515, Oct. 2022, doi: 10.1007/s10586-022-03581-8.

[5] R. Morabito, "Power consumption of virtualization technologies: An empirical investigation," in *Proc. IEEE/ACM 8th Int. Conf. Utility Cloud Comput. (UCC)*, Dec. 2015, pp. 522–527, doi: 10.1109/UCC.2015.93.

[6] A. U. Rehman, Z. Ahmad, A. I. Jehangiri, M. A. Ala'Anzy, M. Othman, A. I. Umar, and J. Ahmad, "Dynamic energy efficient resource allocation strategy for load balancing in fog environment," *IEEE Access*, vol. 8, pp. 199829–199839, 2020, doi: 10.1109/ACCESS.2020.3035181.

[7] Y. Zhai, X. Zhang, S. Eranian, L. Tang, and J. Mars, "HaPPy: Hyperthread-aware power profiling dynamically," in *Proc. USENIX Annu. Tech. Conf.*, 2014, pp. 211–217.

[8] P. Jing, Y. Su, X. Jin, and C. Zhang, "High-order temporal correlation model learning for time-series prediction," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 2385–2397, Jun. 2019, doi: 10.1109/TCYB.2018.2832085.

[9] R. Tracey, L. Hoang, F. Subelet, and V. Elisseev, "AI-driven holistic approach to energy efficient HPC," in *Proc. Int. Workshops High Perform. Comput.*, 2020, pp. 267–279, doi: 10.1007/978-3-030-59851-8.

[10] D. Rahbari and M. Nickray, "Low-latency and energy-efficient scheduling in fog-based IoT applications," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 27, no. 2, pp. 1406–1427, Mar. 2019, doi: 10.3906/elk-1810-47.

[11] M. A. Ala'anzy, M. Othman, E. M. Ibbini, O. Enaizan, M. Farid, Y. A. Alsaaidah, Z. Ahmad, and R. M. Ghoniem, "Replication-based dynamic energy-aware resource provisioning for scientific workflows," *Appl. Sci.*, vol. 13, no. 4, p. 2644, Feb. 2023, doi: 10.3390/app13042644.

[12] K. Alatoun, K. Matrouk, M. A. Mohammed, J. Nedoma, R. Martinek, and P. Zmij, "A novel low-latency and energy-efficient task scheduling framework for Internet of Medical Things in an edge fog cloud system," *Sensors*, vol. 22, no. 14, p. 5327, Jul. 2022, doi: 10.3390/s22145327.

[13] A. H. A. AL-Jumaili, R. C. Muniyandi, M. K. Hasan, M. J. Singh, J. K. S. Paw, and M. Amir, "Advancements in intelligent cloud computing for power optimization and battery management in hybrid renewable energy systems: A comprehensive review," *Energy Rep.*, vol. 10, pp. 2206–2227, Nov. 2023, doi: 10.1016/j.egyr.2023.09.029.

[14] A. H. A. AL-Jumaili, R. C. Muniyandi, M. K. Hasan, J. K. S. Paw, and M. J. Singh, "Big data analytics using cloud computing based frameworks for power management systems: Status, constraints, and future recommendations," *Sensors*, vol. 23, no. 6, p. 2952, Mar. 2023, doi: 10.3390/s23062952.

[15] M. Abbasi, E. Mohammadi-Pasand, and M. R. Khosravi, "Intelligent workload allocation in IoT-fog-cloud architecture towards mobile edge computing," *Comput. Commun.*, vol. 169, pp. 71–80, Mar. 2021, doi: 10.1016/j.comcom.2021.01.022.

[16] K. Thakur, A. S. K. Pathan, and S. Ismat, "Distributed cloud computing," in *Emerging ICT Technologies and Cybersecurity: From AI and ML to Other Futuristic Technologies*. Cham, Switzerland: Springer, 2023, pp. 185–197.

[17] Z. Liu, L. Wan, J. Guo, F. Huang, X. Feng, L. Wang, and J. Ma, "PPRU: A privacy-preserving reputation updating scheme for cloud-assisted vehicular networks," *IEEE Trans. Veh. Technol.*, 2023.

[18] Y. Miao et al., "Time-controllable keyword search scheme with efficient revocation in mobile e-health cloud," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 3650–3665, May 2024, doi: 10.1109/TMC.2023.3277702.

[19] Y. R. Savitri and U. Lasminto, "Drainage network optimization for inundation mitigation case study of ITS Surabaya," in *Proc. AIP Conf.*, 2017, pp. 1–12, doi: 10.1063/1.4985520.

[20] Y. Xiao and M. Krunz, "AdaptiveFog: A modelling and optimization framework for fog computing in intelligent transportation systems," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4187–4200, Dec. 2022, doi: 10.1109/TMC.2021.3080397.

[21] A. Gupta, H. S. Bhadauria, and A. Singh, "RETRACTED ARTICLE: SLA-aware load balancing using risk management framework in cloud," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 7, pp. 7559–7568, Jul. 2021, doi: 10.1007/s12652-020-02458-1.

[22] M. A. Ala'anzy, M. Othman, S. Hasan, S. M. Ghaleb, and R. Latip, "Optimising cloud servers utilisation based on locust-inspired algorithm," in *Proc. 7th Int. Conf. Soft Comput. Mach. Intell. (ISCMI)*, Nov. 2020, pp. 23–27, doi: 10.1109/ISCMI51676.2020.9311584.

[23] H. M. D. Kabir, A. Khosravi, S. K. Mondal, M. Rahman, S. Nahavandi, and R. Buyya, "Uncertainty-aware decisions in cloud computing," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–30, May 2022, doi: 10.1145/3447583.

[24] Z. Ahmad, T. Acarer, and W. Kim, "Optimization of maritime communication workflow execution with a task-oriented scheduling framework in cloud computing," *J. Mar. Sci. Eng.*, vol. 11, no. 11, p. 2133, Nov. 2023, doi: 10.3390/jmse11112133.

[25] I. Rodero, J. Jaramillo, A. Quiroz, M. Parashar, F. Guim, and S. Poole, "Energy-efficient application-aware online provisioning for virtualized clouds and data centers," in *Proc. Int. Conf. Green Comput.*, Aug. 2010, pp. 31–45, doi: 10.1109/GREENCOMP.2010.5598283.

[26] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Gener. Comput. Syst.*, vol. 28, no. 5, pp. 755–768, May 2012, doi: 10.1016/j.future.2011.04.017.

[27] Y. Miao, Y. Yang, X. Li, L. Wei, Z. Liu, and R. H. Deng, "Efficient privacy-preserving spatial data query in cloud computing," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 1, pp. 122–136, Jan. 2023, doi: 10.1109/TKDE.2023.3283020.

[28] S. F. Piraghaj. (Mar. 2016). *Energy-Efficient Management of Resources in Container-Based Clouds*. Univ. Melbourne. [Online]. Available: http://www.cloudbus.org/students/SarehPhDThesis2016.pdf

[29] R. Verma, D. Rane, R. S. Jha, and W. Ibrahim, "Next-generation optimization models and algorithms in cloud and fog computing virtualization security: The present state and future," *Sci. Program.*, vol. 2022, pp. 1–10, Jul. 2022, doi: 10.1155/2022/2419291.

[30] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, "Evaluating the effectiveness of model-based power characterization," in *Proc. USENIX Annu. Tech. Conf.*, 2011, pp. 159–172.

[31] M. Plauth, L. Feinbube, and A. Polze. (2017). *A Performance Evaluation of Lightweight Approaches To Virtualization*. [Online]. Available: http://www.thinkmind.org/

[32] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *J. Parallel Distrib. Comput.*, vol. 134, pp. 75–88, Dec. 2019, doi: 10.1016/j.jpdc.2019.07.007.

[33] J. von Kistowski, M. Deffner, and S. Kounev, "Run-time prediction of power consumption for component deployments," in *Proc. IEEE Int. Conf. Autonomic Comput. (ICAC)*, Sep. 2018, pp. 151–156, doi: 10.1109/ICAC.2018.00025.

[34] K. Reddy Basireddy, E. W. Wachter, B. M. Al-Hashimi, and G. Merrett, "Workload-aware runtime energy management for HPC systems," in *Proc. Int. Conf. High Perform. Comput. Simul. (HPCS)*, Jul. 2018, pp. 292–299, doi: 10.1109/HPCS.2018.00057.

[35] A. Tarafdar, M. Debnath, S. Khatua, and R. K. Das, "Energy and quality of service-aware virtual machine consolidation in a cloud data center," *J. Supercomput.*, vol. 76, no. 11, pp. 9095–9126, Nov. 2020, doi: 10.1007/s11227-020-03203-3.

[36] P. Hammarlund et al., "Haswell: The fourth-generation Intel core processor," *IEEE Micro*, vol. 34, no. 2, pp. 6–20, Mar. 2014, doi: 10.1109/MM.2014.10.

[37] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan, "Power-management architecture of the Intel microarchitecture code-named sandy bridge," *IEEE Micro*, vol. 32, no. 2, pp. 20–27, Mar. 2012, doi: 10.1109/MM.2012.12.

[38] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proc. 1st ACM Symp. Cloud Comput.* New York, NY, USA: ACM, Jun. 2010, pp. 39–50, doi: 10.1145/1807128.1807136.

[39] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan, "VM power metering," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 3, pp. 56–60, Jan. 2011, doi: 10.1145/1925019.1925031.

[40] R. Bertran, Y. Becerra, D. Carrera, V. Beltran, M. Gonzalez, X. Martorell, J. Torres, and E. Ayguade, "Accurate energy accounting for shared virtualized environments using PMC-based power modeling techniques," in *Proc. 11th IEEE/ACM Int. Conf. Grid Comput.*, Oct. 2010, pp. 1–8, doi: 10.1109/GRID.2010.5697889.

[41] A. Guldner et al., "Development and evaluation of a reference measurement model for assessing the resource and energy efficiency of software products and components—Green software measurement model (GSMM)," *Future Gener. Comput. Syst.*, vol. 155, pp. 402–418, Jun. 2024, doi: 10.1016/j.future.2024.01.033.

[42] M. Jay, V. Ostapenco, L. Lefevre, D. Trystram, A.-C. Orgerie, and B. Fichel, "An experimental comparison of software-based power meters: Focus on CPU and GPU," in *Proc. IEEE/ACM 23rd Int. Symp. Cluster, Cloud Internet Comput. (CCGrid)*, May 2023, pp. 106–118, doi: 10.1109/CCGrid57682.2023.00020.

[43] Y. Xie, M. Jin, Z. Zou, G. Xu, D. Feng, W. Liu, and D. Long, "Real-time prediction of Docker container resource load based on a hybrid model of ARIMA and triple exponential smoothing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 1386–1401, Apr. 2022, doi: 10.1109/TCC.2020.2989631.

[44] W. Lin, F. Shi, W. Wu, K. Li, G. Wu, and A.-A. Mohammed, "A taxonomy and survey of power models and power modeling for cloud servers," *ACM Comput. Surv.*, vol. 53, no. 5, pp. 1–41, Sep. 2021, doi: 10.1145/3406208.

[45] G. Fieni, R. Rouvoy, and L. Seinturier, "SmartWatts: Self-calibrating software-defined power meter for containers," in *Proc. 20th IEEE/ACM Int. Symp. Cluster, Cloud Internet Comput. (CCGRID)*, May 2020, pp. 479–488, doi: 10.1109/CCGrid49817.2020.00-45.

[46] L. Zhang et al., "A novel hybrid model for docker container workload prediction," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 2726–2743, Sep. 2023, doi: 10.1109/TNSM.2023.3248803.

[47] H. Yuan and S. Liao, "A time series-based approach to elastic kubernetes scaling," *Electronics*, vol. 13, no. 2, p. 285, Jan. 2024, doi: 10.3390/electronics13020285.

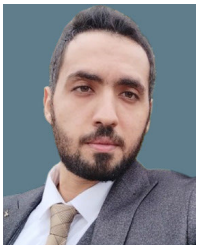
[48] N. S. Joshi, R. Raghuwanshi, Y. M. Agarwal, B. Annappa, and D. Sachin, "ARIMA-PID: Container auto scaling based on predictive analysis and control theory," *Multimedia Tools Appl.*, vol. 83, no. 9, pp. 26369–26386, Aug. 2023, doi: 10.1007/s11042-023-16587-0.

[49] D. Ou, C. Jiang, M. Zheng, and Y. Ren, "Container power consumption prediction based on GBRT-PL for edge servers in smart city," *IEEE Internet Things J.*, vol. 10, no. 21, pp. 18799–18807, Nov. 2023, doi: 10.1109/JIOT.2023.3281368.

ABDULMOHSEN ALGARNI received the Ph.D. degree from the Queensland University of Technology, Australia, in 2012. He was a Research Associate with the School of Electrical Engineering and Computer Science, Queensland University of Technology, in 2012. He is currently an Associate Professor with the College of Computer Science, King Khalid University. His research interests include artificial intelligence, data mining, text mining, machine learning, information retrieval, and information filtering.

IQRAR SHAH received the M.S. degree in computer science from the Department of Computer Science and Information Technology, Hazara University Mansehra, Pakistan. His research interests include energy management in cloud computing and time series analysis.

ALI IMRAN JEHANGIRI received the degree from Bergische Universität Wuppertal, Germany, in 2010, and the Ph.D. degree in computer science from Georg-August-Universität Göttingen, Germany, in 2015. He was a Research Assistant with GWDG, where he gained industrial experience with service computing. He is currently an Assistant Professor with the Department of Computer Science and Information Technology, Hazara University Mansehra, Pakistan. He is the author of several publications in international journals and conferences. His research interests include parallel, grid computing, cloud computing, the Internet of Things, and big data.



MOHAMMED ALAA ALA'ANZY received the Ph.D. degree in computer science from University Putra Malaysia (UPM), in 2023.

He is currently an Assistant Professor with Suleyman Demirel University (SDU). He specializes in cutting-edge fields, such as algorithms, cloud computing, green computing, load balancing, task scheduling, and fog computing. He is widely recognized for his substantial contributions to the academic community through high-impact journals and conference publications. Additionally, he plays a pivotal role as a respected reviewer for esteemed journals, such as IEEE, Elsevier, and Springer.

ZULFIQAR AHMAD received the M.Sc. degree (Hons.) in computer science from Hazara University Mansehra, Pakistan, in 2012, the M.S. degree in computer science from COMSATS University Islamabad, Abbottabad, Pakistan, in 2016, and the Ph.D. degree in computer science from the Department of Computer Science and Information Technology, Hazara University Mansehra, in 2022. He is the author of several publications in the fields of fog computing, cloud computing, high-performance computing, and scientific workflow execution and management. His research interests include scientific workflow management in cloud computing, the Internet of Things, fog computing, edge computing, cybersecurity, and wireless sensor networks (WSNs).