2022 IEEE Smartworld, Ubiquitous Intelligence & Computing, Scalable Computing & Communications, Digital Twin, Privacy Computing, Metaverse, Autonomous & Trusted Vehicles (SmartWorld/UIC/ScalCom/DigitalTwin/PriComp/Metaverse)

TESCO: Multiple Simulations based AI-augmented Fog computing for QoS Optimization

Sundas Iftikhar
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, United Kingdom
s.iftikhar@qmul.ac.uk

Uttkarsh Raj
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, United Kingdom
u.raj@qmul.ac.uk

Shreshth Tuli
Department of Computing
Imperial College London
London, United Kingdom
s.tuli20@imperial.ac.uk

Muhammed Golec
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, United Kingdom
m.golec@qmul.ac.uk

Deepraj Chowdhury
Department of Electronics and communication engineering
International Institute of Information Technology
Naya Raipur, India
-deepraj19101@iiitnr.edu.in

Sukhpal Singh Gill
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, United Kingdom
s.s.gill@qmul.ac.uk

Steve Uhlig
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, United Kingdom
steve.uhlig@qmul.ac.uk

***Abstract*—Fog computing is one of the most widely used paradigms for analyzing and computing the data locally, instead of sending data to remote cloud servers. The main objective of fog computing is to lower the latency of services compared to cloud systems while also reducing the requirement of network bandwidth across computing devices. But as in a fog environment, more data is processed locally, fog nodes usually receive more requests daily, leading to system contention and processing overload. To tackle this, the state-of-the-art relies on machine learning coupled with simulations to forecast processing loads and limit the chances of node contention. However, prior work performs a single simulation to test whether a resource management decision, such as task placement, is optimal or not. This usually ignores the stochastic aspects in the environment or the noise in simulations. To resolve this, we proposed a novel approach called TESCO to perform simulations to test multiple candidate scheduling decisions and decide the optimal one. We demonstrate that TESCO outperforms state-of-the-art scheduler (COSCO) that uses single simulations with low overheads.**

***Index Terms*—AI-augmented fog computing, QoS optimization, Task scheduling**

I. Introduction

Fog computing is a recent trend in the area of distributed systems. The literature has reported that data processing has moved away from cloud computing towards more local processing environments, also referred to as fog computing [1][2]. To reduce the latency of the system, cloud computing has been extended closer to the "edge" of the network following the data gravity principle. Herein, the data processing occurs in the local devices near to the edge device rather than in the cloud [3]. Figure. 1 shows layers in the architecture of fog computing. Fog computing can reduce latency of network and storage requirement of a system, by bringing them closer to end users, which results in faster response to the users [4]. The main reason behind utilizing fog computing presently is to give better Quality of Service (QoS) for latency-sensitive IoT applications. It also calls for effectively assessment and monitoring the offered services. In fog computing, resources can be hosted at terminals, such as Wi-Fi access points or set-top boxes. It runs close to the ground, generates automatic reactions that provide value [5]. Adopting fog with cloud is distinguished due to its closeness to end users, ability to enable mobility, and dense geographical dispersion. However, the fundamental differentiating factor for fog architectures is that the nodes are resource constrained and prone to contention. With all the benefits of fog computing, inherent issues with fog architecture make implementing latency-sensitive applications in fog architecture challenging [6]. As in fog environments more data is processed locally, fog devices get more service request daily, leading to system contention and processing

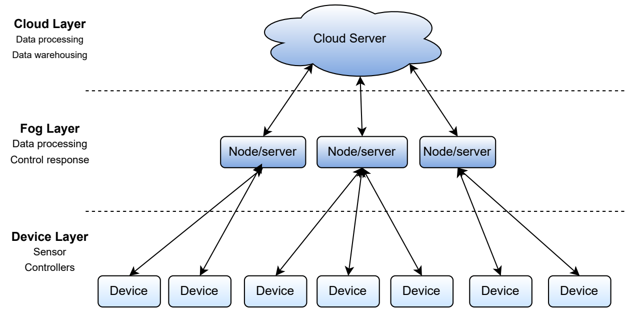

Fig. 1. Fog Architecture

overload. Efficient management of system resources in fog layer is very important to avoid system contention and processing overload. According to the prior work [7], a set of the resource management methods employed in cloud computing are irrelevant in fog computing because of the latter's unique characteristics, such as its reliance on a wide range of devices, geographical dispersion, and mobility.

New studies [8][9] confirm the importance of optimising aspects such as execution time, response time, dependability, energy usage, and waiting time for contemporary applications and load balancing and system contention avoidance in fog environment [10]. Latest advancement in Artificial Intelligence (AI) has attained the attention of researcher for resource management issues. Different resource management techniques based on AI solutions and non AI solutions are presented to handle resource contention, load balancing, latency, energy use, resource utilisation, and associated costs challenges in literature [5][6]. AI based techniques are emerging as promising techniques to handle these issues and optimize above mentioned aspects. In addition, we can use AI techniques such as Machine learning (ML) on millions of terabytes of gathered data to analyse fog system operations; this is why fog computing is expected to eventually overtake cloud computing as the industry standard, bringing together Internet of Things (IoT), distributed intelligence, and Big Data [11]. In prior works, ML methods are used for workload predictions, resource consumption prediction, anomaly prediction, decision making (Reinforcement learning). Some state-of-the-art solutions typically use a combination of data-driven ML methods with simulated runs on a cyber-physical digital twin of the computational infrastructure. This enables such methods to look-ahead into the resource consumption and load requirements for a future time step. Such predictions facilitate in making a more informed decisions. However, prior work tends to use ML methods with simulation for single decision to generate feedback signals and informed resource management solutions. In this work, we conjecture that simulations are more powerful in compared to the way they are used in prior work. We leverage simulations to not only generate feedback signals for single scheduling decision, but generate feedback signals for multiple scheduling decisions and compare across decisions and execute optimal placement. We do this by proposing a novel approach, which we refer to as mulTiplE Simulations based sCheduler to offer AI-augmented fog computing framework called **TESCO**. The main contributions of this work are:

- Proposing a co-simulation based AI technique TESCO for QoS Optimization in fog environment.
- Comparing variations of TESCO against state-of-the-art methods that leverage single-step simulations.
- Sensitivity analysis of the hyperparameters in TESCO to present and leverage performance and compute overhead tradeoffs.
- Evaluations that demonstrate that TESCO outperforms state of the art approaches by improving execution times, wait time, response times, Service Level Agreement (or SLA) violation and energy consumption by up to 3.15%, 13.64%, 3.16%, 34% and 11%, respectively.

II. Related Work

In this section, we discuss and present the literature. With the rapid development of AI and chip technologies, the diversity of IoT has also increased. We can now see these IoT devices in fields as diverse as e-health, military applications, and education [5]. The variety and number of devices with different energy consumption and processing power connected to the edge have increased with the number of IoT devices. This variety creates undesirable fluctuations in response time and energy consumption in edge/fog computing. Therefore, there is a need for solutions that can handle workload demands with low latency in an energy-efficient manner. When the literature is examined, it is seen that there are studies to overcome this problem with schedule-compute tasks. According to Tuli et al. [6] optimized the QoS parameters by formulating a Gradient-based algorithm with the COSCO framework they proposed. It also provides a simulation environment for edge with this work and allows container orchestration. They claimed that the gradient-based back-propagation approach they proposed, performs better than other state-of-the-art scheduler approaches and reduces system contention. The proposed approach uses a neural network model (Gradient Based Optimization Strategy) as a surrogate that predicts QoS parameters of the system for a future timestep. The surrogate model is trained using simulation based feedback generated by forecasting the resource utilization characteristics of the running workloads. This enables training of data-driven models such as those that are based on deep neural networks using simulation based feedback signals as supervision labels. In another study, the authors proposed a new task scheduling framework called HUNTER to optimize energy efficiency in rapidly growing Cloud Data Centres (CDC) [12]. The amount of energy consumed for cooling infrastructure in data centers can be as large as the amount of energy spent for computing [13]. This results in an increase in energy consumption and thus carbon footprint in CDC. Using Gated Graph Convolution Network (GGCN) model, energy, thermal and cooling factors are modelled. Thus, it is aimed to optimize energy efficiency in CDCs. The authors evaluated

TABLE I
COMPARISON OF TESCO WITH EXISTING WORKS.

Study	**Task Scheduling**	**Simulation Mode**
COSCO [6]	Fog/Edge	Single Simulation
HUNTER [12]	Cloud	Single Simulation
Ifogsim [18]	Fog/Edge	Single Simulation
CloudSim [9]	Cloud	Single Simulation
TESCO	Fog/Edge	Multiple Simulation

the performance of HUNTER in a simulated environment using the CloudSim and COSCO frameworks. In Calheiros et al. study [9], Cloudsim, a comprehensive simulation for simulating and modelling cloud computing environments, is proposed. In addition, CloudSim provides migration flexibility for space and time-based shares for virtualized services. Thus, with CloudSim, the authors aim to design new application provisioning algorithms for cloud computing. A drawback of such approaches is the requirement of large amounts of data to train deep neural networks as surrogate models [14, 15]. This may not be feasible in many settings as generating data has significant time and cost overheads [16]. Another challenge with such approaches is that they are trained for specific system configurations while generating data at training time. If this is not the case in at test time or there is concept drifts within the distributions of the resource requirements of the running workloads, then these approaches need to be re-trained. This imposes further overheads for fine-tuning neural models [17]. Thus, it is crucial to have approaches that are data agnostic and use the power of simulations to take decisions. This work takes a step in the direction in making resource management agnostic to pre-collected data. We do this using an approach that is purely based on heuristics and simulations. We use a convex combination of response time and energy consumption to choose across multiple candidate solutions (more details given in Section IV). Table I summarizes the similarities and differences among TESCO and the related methods.

III. METHODOLOGY

A. Analysis

Realizing the claimed influence and benefits of fog computing on people's lives, we analyze the existing literature on fog computing, discussed in section II, and found that it is an emerging paradigm of distributed systems that includes all intermediate devices between cloud and IoT layer. The execution, storage, and network latency can be significantly reduced by pulling the execution and storage devices in proximity to the user in form of fog computing. It results in several benefits in form of reduced cost, and latency, improved reliability, increased stability, and lower network load. As a case study of fog computing, we can consider example of Netflix. Netflix caches and processes popular videos locally using fog computing and give a better viewing experience. While unpopular videos are stored and processed in the cloud layer which results in higher latency and also in some cases lower video quality. But the question is can we cache and process all the videos on edge? Assuming that there may be billions of edge nodes near the users and limited Cloud nodes reachable to all which results in running everything at the edge is pretty expensive. To make the deployments workable, edge devices are commonly close to the users and have a limited capacity for processing, whereas Cloud devices on other hand are far but can handle large amounts of workload. Therefore, there is a need to balance task placement to meet the user latency demands which also minimizes deployment and the execution costs [6].

B. Challenges and Motivation

During the literature review, major challenges we found are: delivering low latency for time critical applications and reduce energy consumption. Many industries such as health-care, robotics, smart vehicles, smart cities require low response time for tasks that are sensitive to service level agreements. The major challenge for achieving low response-time and energy consumption is provoked by the modern applications with high dynamic workloads and the host machines with non-stationary resource capacities. A few applications like Scavenging method and renewable resources can help drive the transition to more sustainable model also some works used reinforcement learning and other AI methods to solve these problems to some extent but some times not being ample to adapt to the volatile settings or simply being not capable of keeping with the extreme user demands [6]. This work focuses to solve existing problems to provide better QoS in fog environments.

C. Design and Implementation

For optimization of the QoS parameters, we considered the convex matrix of Response time and Energy consumption. Equal weights are given to both parameters. The objective function for an interval can be calculated using equation 1:

$$\Theta(P_t) = \alpha.AEC_t + \beta.ART_t. \tag{1}$$

where as Average energy consumption and average response time is calculated using equation 2, 3 taken from work [6]. In the equation, Average Energy Consumption is represented by AEC, ART represents Average Response Time, host is represented by h, Power function of host with $Powerh_i$, specific instant t, and Leaving tasks by L_t.

$$\mathbf{AEC_t} = \frac{\sum h_i \epsilon H \int_{t=s(I_t)}^{s(I_{t+1})} Powerh_i(t)dt}{|A_t| \sum_{h_i \epsilon H} Power_{h_i}^{max} \times (t_{i+1} - t_i)} \tag{2}$$

$$\mathbf{ART_t} = \frac{\sum_{l_j^t \epsilon L_t} ResponseTime(l_j^t)}{|L_t|\, max_{s \leq t} max_{l_j^s \epsilon L_s} ResponseTime(l_j^s)} \tag{3}$$

We choose a simulation-based approach to implement our technique. After careful analysis of available simulators, an event-based simulator named COSCO is selected for implementation. Bitbrain workload traces [19] are used to test the performance of the proposed approach. For the design and implementation of work, it is assumed that Instructions per second, disk capacity, and network bandwidth of each host are known ahead of time. Resource consumption for a task can be calculated in the environment in a given time.

D. *Evaluation*

The proposed technique is evaluated by comparing its results with the single simulation technique. Single Simulation is executed with different variations of parameters, e.g., changing number of hosts in the environment and interval time. By doing this, we observed that single simulation technique only considered the top heuristic score which does not guarantee the best score is achieved. This issue is addressed using TESCO where we have top 10, 20, 30 or 40 implementation decisions. Evaluation of TESCO showed comparatively better results for QoS parameters in fog.

IV. TESCO Scheduler

A. *System Architecture*

This section describes the architecture of our system. We divided our system in three layers: IoT layer, fog management layer and fog resource layer.

- IoT Layer: This layer consists of two types of devices. IoT devices and Gateway devices. Tasks are sent to gateway devices and gateway devices are responsible for sending tasks to fog management layer.
- Fog Management Layer: Fog management consists of two components: Scheduler and Resource Monitoring service. Resource monitoring service is responsible for monitoring upcoming tasks and resources available. When new tasks arrive it alerts the scheduler. The scheduler performs its duty in three steps: Host selection, Container selection for migration, and target host selection.
- Fog Resource Layer: fog resource layer has Edge sub-layer and Cloud sub-layer where we have processing and storage resources. Edge devices are close to the users and have a limited capacity for processing, whereas Cloud devices on other hand are far but can handle large amounts of workload.

We partly modified the single simulation-based scheduler (i.e., *Single-Simulation.py*) which test whether a resource management decision, such as task placement, is optimal or not and extended it to Multiple simulation-based scheduler (i.e., *Multiple-Simulation.py*) which tests multiple candidate solutions to decide the optimal one to help in improving QoS parameters for fog computing.

B. *Proposed Technique*

Our proposed technique works in three steps: Host selection, Container selection and Target host selection. First, it selects host from available host from which container needs to be migrated, in second step it selects the the containers which have high host Instructions per second (IPS) utilization and thirdly it selects appropriate host on which new container or migrating container is to be placed.

- *Host Selection*: For the selection of hosts which need container migration, a threshold policy has been used.

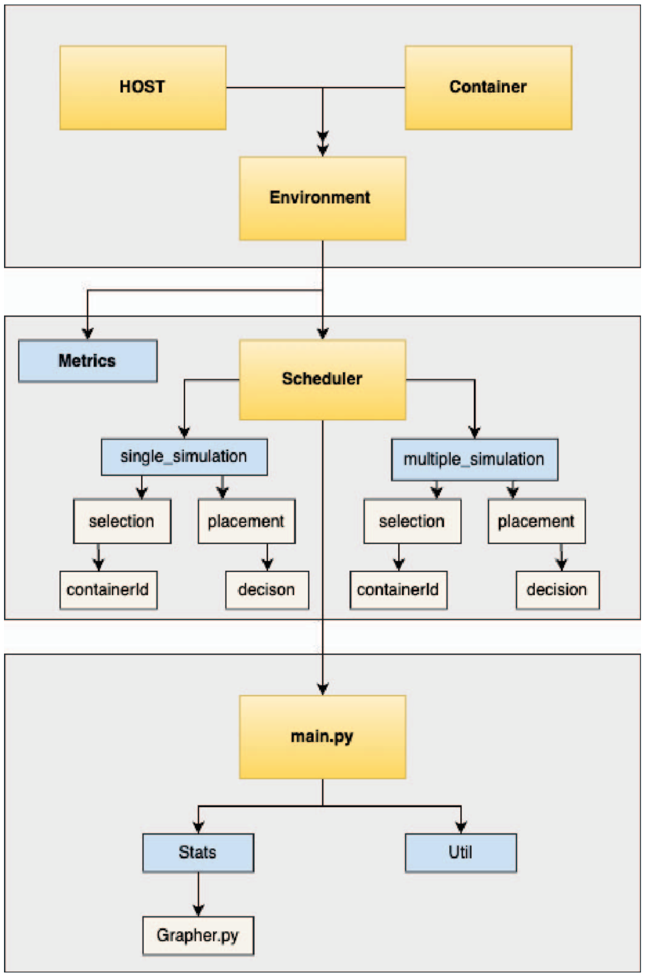

Fig. 2. System Architecture

The threshold is set to 70 percent. If the utilization of host CPU is greater than 70, then those hosts are selected. We adopted this threshold value from work [20].

- *Container Selection*: From the selected hosts, the containers which have the highest CPU utilization ratio are selected for migration.
- *Targeted Host Selection*: For placement of new containers and previously selected containers in interval It, gradient based scheduler gives preferred decisions for placement. Using Long Short-Term Memory (LSTM), utilization metrics for next interval are predicted. Now, with predicted task and host utilization metrics from LSTM and decision from gradient based scheduler, simulator predicts QoS at end of interval It. These inputs are feed to another neural approximator for convex combination of objective function. Then again using trained model and scheduler data set is generated to train neural network. It returns the decisions with hyper- parameters set to 10,20,30,40. At interval It output of previous interval is

given to the gradient based scheduler. Hyper-parameters (decision-variable) are set to 10, 20, 30, and 40 to observe the performance of model. It uses hyper-parameter value and utilization metrics to output preferred decision. Simulator is used to estimate objective function for given decisions. Detailed description of our simulation is given below:

- *main.py*: All the global constants for our simulation are initialized in main.py. Since we used event based simulator, we set simulation steps first. The Interval-time is set to 300 sec (seconds) or 5 minutes and every scheduling interval will get executed in every 5 minute. Number of containers are set equal to number of hosts. Total-power is set to 1000 kilowatts and bandwidth of router is 10000 megabytes per second.
- *stats.py*: This component maintains record of workload characteristics, waiting tasks in queue, and utilization metrics of tasks and hosts. On completion of each interval, this component calls savestats() function which saves idle hosts, active hosts, active tasks, waiting tasks and the workload information.
- *scheduler.py*: The scheduler component enables the scheduling of the decisions with help of task selection and task placement. Task *selection()* function picks the container to be migrated to different host based on threshold policy whereas task *placement()* function returns the targeted host for every task selected by *selection()* function or the new tasks created during that interval.
- *Multiple-Simulation.py*: In Multiple Simulation, we execute the *multiple Simulation* method for top 10, 20, 30 or 40 decisions. It work in three steps: First the scheduler gives top scheduling decision based on the hyper-parameter set. Then utilization metric predictor is used to predict utilization metrics for next interval and in third step top decisions and utilization metrics are used and simulation is run to predict QoS parameters.

The pseudo-code of the steps applied for task orchestration is shown in Algorithm 1. Here T the tasks to be scheduled on host h. For time interval It if time interval is equal to zero, task is allocated randomly otherwise, it gets output of previous interval $At - 1$. The Utilization metrics U_metric are feed to trained Long short-term memory (LSTM) model which gives prediction for utilization metrics of next interval. Scheduling decision is simulated and objective function $O(Pt)$ is calculated based on simulation.

V. EXPERIMENTAL RESULTS

This section presents the simulation setup, baseline approach and comparative results of Multiple Simulation with *Single-Simulation* from the paper [6] for a given set of 10 hosts.

A. Simulation setup

To test the efficacy of the proposed approach and compare its results with the baseline approach, we have done experiments in simulated environment. We simulated 10 Azure fog host machines with six hosts in fog layer and 4 in the cloud layer. Hosts correspond to simulation with MIPS, RAM, Bandwidth capacities with power consumption with CPU utilization. Since in simulation, placement of host on geographical places is not feasible, we have depicted this with network and latency attributes of hosts. The detailed specifications of host machines are given in Table II.

Algorithm 1 TESCO Algorithm

```
0: Input: Task, Host, $\mathbf{LSTM_{trainedmodel}}$, $\mathbf{O(p_t)}$,
   function − approximator f
   Output: Decision
   Begin
      For scheduling interval $I_t$
      if  t==0
        D $\leftarrow D_{random}$
      $else$
        $Get \ominus(A_{t-1}), \ominus(H_{t-1})$
      $U_{metric}(A_t) \leftarrow LSTM(U_{metric}(A_{t'}))$
        set  hyper-parameter (decision-variable)
        dec = scheduler (I_t)
        $U(H_i)$, $AEC$,$ART$ $\leftarrow$ simulated (Dec (A_t), $U(A_t)$)
        calculate O (P_t)
      $D = minimize(D, f, O(P_t), \ominus(A_{t-1}), \ominus(H_{t-1}))$
      $fine - tune\ f$
      $return\ D$
   End =0
```

B. Baseline approach

We have evaluated the proposed approach TESCO against the Single simulation scheduling approach. In the Single simulation approach, 1) coupled simulations in which the simulator runs in the background with a scheduling algorithm and generates more data to facilitate the decision-making of an AI model and 2) container orchestration are combined to estimate QoS parameters in the near future. It allows single-step simulation of container migration decisions.

C. Metrics

For evaluation, we used average energy consumption, average response time, SLA violation, scheduling time, and average wait time metrics, and their values are calculated using equation 2, 3, 4, taken from [6] work. Here L_t, represents leaving tasks.

$$\mathbf{SLA} = \frac{\sum_t \sum_{l_j^t \epsilon L_t} \mathbb{I}(ResponseTime(l_j^t) \leq \psi(l_j^t)}{\sum_t |L_t|} \tag{4}$$

Scheduling Time: Average time to reach the scheduling decision over all intervals in the run.

Average wait time: Average time for a container in the wait queue before it starts execution.

TABLE II
CHARACTERISTICS OF HOST MACHINES IN SIMULATED ENVIRONMENT

Quantity	Core	MIPS	RAM (MB)	RAM BW (MB/s)	Ping time (ms)	Disk BW ((MB/s)	Network BW ((MB/s)	Layer
4	2	4029	4295	372	3	13.4	1000	Fog
2	4	8102	17180	360	3	10.3	1000	Fog
2	4	8102	17180	360	76	10.3	1000	Cloud
2	2	2000	34360	376	76	11.64	2500	Cloud

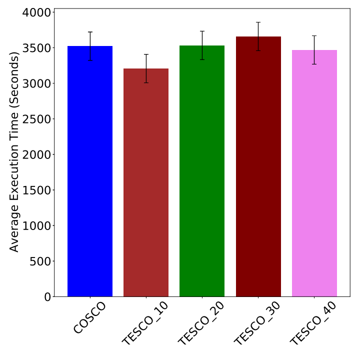

Fig. 3. Average Execution Time

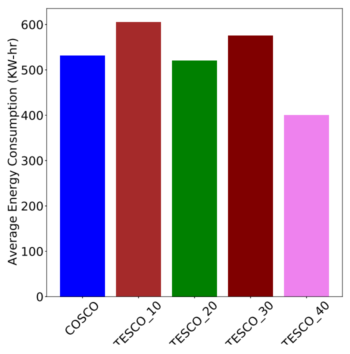

Fig. 5. Average Energy Consumption

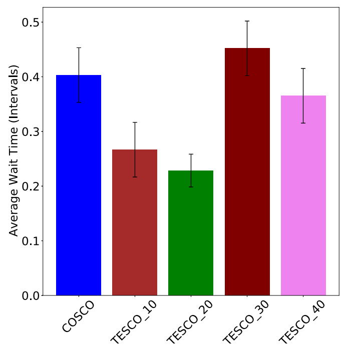

Fig. 4. Average Wait Time

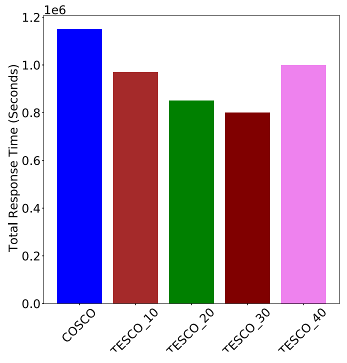

Fig. 6. Total Response Time

D. Results

For comparative analysis of our technique for QoS parameters in fog computing, we run both COSCO (single simulation scheduler) and TESCO (multiple simulation scheduler with hyper-parameters (decisions) set equal to 10, 20, 30, and 40 respectively). The comparative results of TESCO with baseline approach are presented in Table I. We considered average energy consumption, average execution time, average response time, average wait time, and SLA violations as comparison metrics.

- *Execution time*: We evaluated the performance of TESCO in terms of execution time and measured its value in seconds (s). We observed a significant difference in execution time with TESCO scaling lower execution time than single simulation. Figure.3 shows execution time for single simulation and TESCO with various different values of hyper-parameters(decisions). It is noted that TESCO with 10 decisions has shown better results than all others in terms of execution time. Execution time for single simulation and TESCO for 10 decisions is observed to decrease relatively by 3.15% percent which puts a relatively better impact.
- *Average Wait time*: It is the average time between the interval a container is created and the interval its execution started. Wait time is measured in intervals. Figure.4 shows wait time of TESCO with 10, 20 decisions is comparatively lower than single-simulation. It is also noted that a decrease in wait time also decreased response time accordingly. It is also observed that the decrease in wait time is 13.64% with 10 decisions and 17.46% with

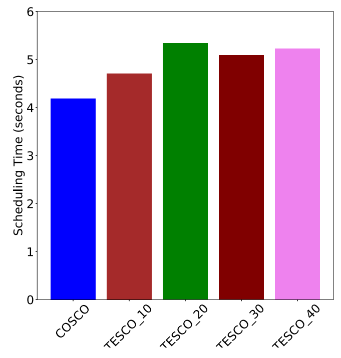

Fig. 7. Average Scheduling Time

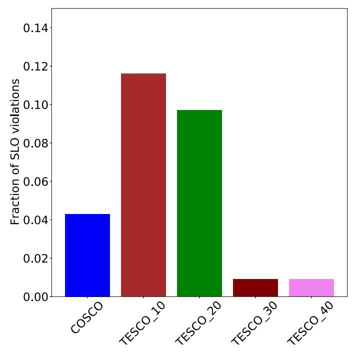

Fig. 8. Fraction of SLO Violation

20 decisions when compared with Single- Simulation.

- *Average Energy Consumption*: Average Energy plays a vital role when it comes to fog computing. Though energy-efficient equipment is used and there are fewer carbon emissions, the cloud infrastructure reduces energy waste. After our experiment is performed, it can be depicted from figure.5 that energy consumption for decisions 20 and 40 showed better result when compared with single-simulation. Multiple simulation, when compared with Single simulation show 11% improvement in energy consumption.
- *Total Response time*: It is defined as sum of time taken to schedule and execute received tasks. Response time shown in fig 6 is measured in seconds. With the comparison of Single-simulation with TESCO with varying number of candidate decisions (i.e., 10,20,30 and 40), it is observed that response time of TESCO scales lower than Single Simulation with decision value 10. With difference in the values when compared with Single Simulation it has been noted that Response time decreased by 3.16%.
- *Average Scheduling time*: Scheduling time is average time to reach the scheduling decision over all intervals and is measured in seconds (s). Figure. 7 shows comparison of scheduling time for TESCO and single-simulation method. The scheduling time for Single Simulation is 4.19 seconds and for the rest of Multiple Simulation it is 4.71, 5.34, 5.09, 5.23 seconds respectively. Due to the reason of using multiple simulation it was expected for the scheduling time to go higher as compared to Single-simulation.
- *SLA Violation*: SLA is an agreement that the client and the cloud service provider sign to guarantee a higher level of service. Because of rising user needs, cloud service providers are presently unable to guarantee QoS, that results in SLA violations. From figure.8, it can be noted that SLA violation rate has drastically decreased for decisions 30 and 40 and which puts a better impact overall. After comparing it with single simulation it is calculated that the SLA violation has raised down to 34%.

TABLE III
COMPARISON OF TESCO AND BASELINE APPROACH USING 10 HOSTS

Metrics	COSCO	TESCO 10	TESCO 20	TESCO 30	TESCO 40
AEC (kW.h)	531.87	605.06	520.80	575.42	400.06
AET (sec)	3521.29	3206.79	3532.36	3658.26	3468.17
ART (sec)	3521.93	3205.91	3531.98	3648.63	3461.29
AWT (interval)	40315.55	26667.11	22847.08	45193.27	36524.15
F (index)	0.00165	0.0018	0.00182	0.00188	0.00173
SV	0.043	0.116	0.097	0.009	0.009
AST (sec)	4.19	4.71	5.34	5.09	5.23

Abbreviations used in Table III are - AEC: Average Energy, AET: Average Execution Time, ART: Average Response Time, AWT: Average Wait Time, SV: SLA Violations, AST: Average Scheduling Time.

Finally with experimenting TESCO and executing simulation with different number of hosts, simulation-steps and interval-time, we observed that the TESCO with 20 decisions is performing better than other simulation techniques. Thus the optimal approach for our current model is TESCO with 20 hyper-parameters with lower scheduling time.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we proposed a multiple simulation-based scheduling technique (TESCO) for fog systems where we use AI coupled with simulations to forecast processing loads and limit the chances of node contention. Contrary to other state-of-art-works, where single simulation is used to test whether a decision is optimal or not, we propose TESCO where simulations are performed to test multiple candidate scheduling decisions and decide the optimal one. In this way, this work also addresses stochastic aspects in the fog computing environment and the noise in simulations. We have compared the performance of TESCO with COSCO baseline, which is a single simulation based technique. Experimental results has shown TESCO outperforms state-of-the-art scheduler that uses single simulations with low overheads.

A. Future Work

TESCO demonstrates better performance in scheduling tasks to hosts in fog but still there ways to improve its performance in the future. This work can be extended to work on the optimization of other QoS parameters such as energy consumption, reliability and availability [5]. TESCO scheduler is currently making decisions on the basis of the container's existing Instructions per second. We can also use other ML models to predict instructions per second for future intervals.

Software Availability

All code, datasets and results are publicly available as part of a GitHub repository under CC-BY License. The repository can be accessed using the URL: https://github.com/sunndas/TESCO

References

[1] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*, 2015, pp. 37–42.

[2] S. Iftikhar, M. Golec, D. Chowdhury, S. S. Gill, and S. Uhlig, "Fogdlearner: A deep learning-based cardiac health diagnosis framework using fog computing," in *Australasian Computer Science Week 2022*, 2022, pp. 136–144.

[3] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. S. Goren, C. Mahmoudi *et al.*, "Fog computing conceptual model," 2018.

[4] S. Iftikhar, M. Golec, D. Chowdhury, S. S. Gill, and S. Uhlig, "Fog computing based router-distributor application for sustainable smart home," in *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*. IEEE, 2022, pp. 1–5.

[5] S. S. Gill, M. Xu, C. Ottaviani, P. Patros, R. Bahsoon, A. Shaghaghi, M. Golec, V. Stankovski, H. Wu, A. Abraham *et al.*, "Ai for next generation computing: Emerging trends and future directions," *Internet of Things*, vol. 19, p. 100514, 2022.

[6] S. Tuli, S. R. Poojara, S. N. Srirama, G. Casale, and N. R. Jennings, "Cosco: Container orchestration using co-simulation and gradient based optimization for fog computing environments," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 1, pp. 101–116, 2021.

[7] B. G. Batista, C. H. G. Ferreira, D. C. M. Segura, D. M. Leite Filho, and M. L. M. Peixoto, "A qos-driven approach for cloud computing addressing attributes of performance and security," *Future Generation Computer Systems*, vol. 68, pp. 260–274, 2017.

[8] Z. M. Nayeri, T. Ghafarian, and B. Javadi, "Application placement in fog computing with ai approach: Taxonomy and a state of the art survey," *Journal of Network and Computer Applications*, vol. 185, p. 103078, 2021.

[9] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[10] S. Iftikhar, A. Tariq, and S. A. Khan, "Optimal task allocation algorithm for cost minimization and load balancing of gsd teams," *Lecture Notes on Software Engineering*, vol. 4, no. 1, pp. 16–19, 2016.

[11] Q. D. La, M. V. Ngo, T. Q. Dinh, T. Q. Quek, and H. Shin, "Enabling intelligence in fog computing to achieve energy and latency reduction," *Digital Communications and Networks*, vol. 5, no. 1, pp. 3–9, 2019.

[12] S. Tuli, S. S. Gill, M. Xu, P. Garraghan, R. Bahsoon, S. Dustdar, R. Sakellariou, O. Rana, R. Buyya, G. Casale *et al.*, "Hunter: Ai based holistic resource management for sustainable cloud computing," *Journal of Systems and Software*, vol. 184, p. 111124, 2022.

[13] E. Pakbaznia and M. Pedram, "Minimizing data center cooling and server power costs," in *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*, 2009, pp. 145–150.

[14] A. Lakhan, Q.-U.-A. Mastoi, M. Elhoseny, M. S. Memon, and M. A. Mohammed, "Deep neural network-based application partitioning and scheduling for hospitals and medical enterprises using iot assisted mobile fog cloud," *Enterprise Information Systems*, vol. 16, no. 7, p. 1883122, 2022.

[15] T. Goethals, F. De Turck, and B. Volckaert, "Self-organizing fog support services for responsive edge computing," *Journal of Network and Systems Management*, vol. 29, no. 2, pp. 1–33, 2021.

[16] P. Kang and P. Lama, "Robust resource scaling of containerized microservices with probabilistic machine learning," in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2020, pp. 122–131.

[17] B. Gu, J. Kong, A. Munir, and Y. G. Kim, "A framework for distributed deep neural network training with heterogeneous computing platforms," in *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2019, pp. 430–437.

[18] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[19] S. Shen, V. Van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2015, pp. 465–474.

[20] R. Ranjan, R. Buyya, and M. Parashar, "Special section on autonomic cloud computing: technologies, services, and applications," 2011.