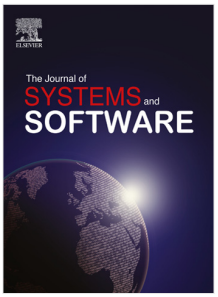
Journal Pre-proof



HUNTER: AI based holistic resource management for sustainable cloud computing

Shreshth Tuli, Sukhpal Singh Gill, Minxian Xu, Peter Garraghan, Rami Bahsoon, Scharam Dustdar, Rizos Sakellariou, Omer Rana, Rajkumar Buyya, Giuliano Casale, Nicholas R. Jennings

PII: S0164-1212(21)00221-1
DOI: https://doi.org/10.1016/j.jss.2021.111124
Reference: JSS 111124

To appear in: *The Journal of Systems & Software*

Received date : 3 September 2021
Accepted date : 11 October 2021

Please cite this article as: S. Tuli, S.S. Gill, M. Xu et al., HUNTER: AI based holistic resource management for sustainable cloud computing. *The Journal of Systems & Software* (2021), doi: https://doi.org/10.1016/j.jss.2021.111124.

HUNTER: AI based Holistic Resource Management for Sustainable Cloud Computing

Shreshth Tuli[£], Sukhpal Singh Gill*, Minxian Xu†, Peter Garraghan‡, Rami Bahsoon§, Schahram Dustdar¶, Rizos Sakellariou‖, Omer Rana**, Rajkumar Buyya††, Giuliano Casale[£] and Nicholas R. Jennings[£]

[£]Department of Computing, Imperial College London, UK
s.tuli20@imperial.ac.uk, g.casale@imperial.ac.uk, n.jennings@imperial.ac.uk

*School of Electronic Engineering and Computer Science, Queen Mary University of London, UK
s.s.gill@qmul.ac.uk

†Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China

mx.xu@siat.ac.cn

‡School of Computing and Communications, Lancaster University, UK

p.garraghan@lancaster.ac.uk

§School of Computer Science, University of Birmingham, Birmingham, UK

r.bahsoon@cs.bham.ac.uk

¶Distributed Systems Group, Vienna University of Technology, Vienna, Austria

dustdar@dsg.tuwien.ac.at

‖Department of Computer Science, University of Manchester, Oxford Road, Manchester, UK

rizos@manchester.ac.uk

**School of Computer Science and Informatics, Cardiff University, Cardiff, UK

ranaof@cardiff.ac.uk

††Cloud Computing and Distributed Systems (CLOUDS) Laboratory,School of Computing and Information Systems, The University of Melbourne, Australia

rbuyya@unimelb.edu.au

Corresponding author: Sukhpal Singh Gill, School of Electronic Engineering and Computer Science, Queen Mary University of London, London, E1 4NS, UK. Email: s.s.gill@qmul.ac.uk

HUNTER: AI based Holistic Resource Management for Sustainable Cloud Computing

Shreshth Tuli[1], Sukhpal Singh Gill[2], Minxian Xu[3], Peter Garraghan[4], Rami Bahsoon[5], Scharam Dustdar[6], Rizos Sakellariou[7], Omer Rana[8], Rajkumar Buyya[9], Giuliano Casale[1], Nicholas R. Jennings[1]

Abstract

The worldwide adoption of cloud data centers (CDCs) has given rise to the ubiquitous demand for hosting application services on the cloud. Further, contemporary data-intensive industries have seen a sharp upsurge in the resource requirements of modern applications. This has led to the provisioning of an increased number of cloud servers, giving rise to higher energy consumption and, consequently, sustainability concerns. Traditional heuristics and reinforcement learning based algorithms for energy-efficient cloud resource management address the scalability and adaptability related challenges to a limited extent. Existing work often fails to capture dependencies across thermal characteristics of hosts, resource consumption of tasks and the corresponding scheduling decisions. This leads to poor scalability and an increase in the compute resource requirements, particularly in environments with non-stationary resource demands. To address these limitations, we propose an artificial intelligence (AI) based holistic resource management technique for sustainable cloud computing called HUNTER. The proposed model formulates the goal of optimizing energy efficiency in data centers as a multi-objective scheduling problem, considering three important models: energy, thermal and cooling. HUNTER utilizes a Gated Graph Convolution Network as a surrogate model for approximating the Quality of Service (QoS) for a system state and generating optimal scheduling decisions. Experiments on simulated and physical cloud environments using the CloudSim toolkit and the COSCO framework show that HUNTER outperforms state-of-the-art baselines in terms of energy consumption, SLA violation, scheduling time, cost and temperature by up to 12, 35, 43, 54 and 3 percent respectively.

Keywords: Holistic Resource Management, Energy-Efficiency, Cloud Computing, Artificial Intelligence, Thermal Management.

1. Introduction

Cloud computing has proven to be a reliable, cost-effective and scalable computing service choice to host and deliver software solutions for diverse industrial applications [1]. Many businesses have migrated to cloud data centers (CDCs) to take advantage of on-demand, elastic and scalable resource provisioning, saving companies on capital investments and maintenance of in-house infrastructure [2]. The plethora of deployment choices offered by most cloud providers allows users to customize resources according to their objectives. However, the rise of AI and Internet of Things (IoT) applications in Industry 4.0 [3] has led to an increase in the overall requirements of cloud resources. In particular, cloud providers, such as Amazon, Microsoft and Google, have witnessed heavy reliance and increase in the number of cloud data centers to fulfill the increasing demands of users [4].

A large amount of energy is required to run these cloud data centers efficiently. Specifically, there is a need to manage the cloud resources effectively to lower the energy consumption and help reduce the cost and carbon footprints. This demand often comes with high energy consumption, a major part of which is attributed to the cooling costs [2]. The cooling infrastructure of a CDC can consume almost the same level of energy as the computing nodes themselves [5]. Major public cloud providers need to invest in large scale cooling infrastructures, making it an expensive exercise [6]. Producing holistic energy-aware models for resource management, that consider both cooling and computational costs, has been acknowledged as an important open problem [6]. Specifically, the research gap presents a need for task scheduling in CDCs that considers the energy, thermal and cooling costs as optimization objectives [2].

Challenges. The problem of providing holistic resource management for sustainable cloud computing is fundamentally challenging because the relationship between energy consumption, the computational infrastructure and the cooling system is complex. Another challenge is the coordination of the scheduling decisions for different tasks that considers both computing power and cooling power in tandem. Another factor to con-

[1]Department of Computing, Imperial College London, UK
[2]School of Electronic Engineering and Computer Science, Queen Mary University of London, UK
[3]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
[4]School of Computing and Communications, Lancaster University, UK
[5]School of Computer Science, University of Birmingham, UK
[6]Distributed Systems Group, Vienna University of Technology, Austria
[7]Department of Computer Science, University of Manchester, UK
[8]School of Computer Science and Informatics, Cardiff University, UK
[9]Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne, Australia
E-mail addresses: s.tuli20@imperial.ac.uk (S. Tuli), s.s.gill@qmul.ac.uk (S.S. Gill), mx.xu@siat.ac.cn (M. Xu), p.garraghan@lancaster.ac.uk (P. Garraghan), r.bahsoon@cs.bham.ac.uk (R. Bahsoon), dustdar@dsg.tuwien.ac.at (S. Dustdar), rizos@manchester.ac.uk (R. Sakellariou), ranaof@cardiff.ac.uk (O. Rana), rbuyya@unimelb.edu.au (R. Buyya), g.casale@imperial.ac.uk (G. Casale) and n.jennings@imperial.ac.uk (N.R. Jennings).

sider in minimizing energy consumption is the reduction in resource intensive or thermal hotspots that can degrade the performance of the system [6]. Consequently, as non-stationary workloads are required to be serviced, the cooling systems and hence the power and temperature metrics of hosts change dynamically [7]. Furthermore, tasks running in a datacenter may be dependent on one another. This is a common case when jobs are allocated to a cloud environment with each job consisting of multiple independent tasks and service level agreements (SLAs) being defined for each job. The overall response time and SLA violations would then be defined at the job level instead of being measured for each task. Moreover, in hybrid public-private clouds, the host machines have different resource capacities in terms of their CPU, RAM, disk and network capabilities. These issues further complicate scheduling as now a representation of the system also needs to capture the inter task dependencies and host heterogeneity.

Existing solutions. Over the past few years many resource management techniques have been proposed that target SLA compliance and the improvement of Quality of Service (QoS) [2]. Specific solutions that target sustainable computing, aim at leveraging monitored metrics like energy consumption and temperature of host machines [2]. Only a few solutions also consider the energy and cost implications of cooling solutions [6]. Most prior work presents meta-heuristic algorithms [6] and deep learning techniques [7]. Most state-of-the-art models use meta-heuristic approaches like genetic algorithms or integer linear programming [6, 8, 9, 10, 11]. Other recent methods use reinforcement learning (RL), specifically the traditional tabular models like Q-Learning [12, 13]. However, such meta-heuristic and RL techniques require several exploratory decisions before updating their models, making it harder to adapt quickly in highly volatile settings, considering inter-task dependencies, thermal characteristics or converging quickly to a scheduling decision [14]. All such features are crucial for a holistic solution for sustainable scheduling [2].

Background and new insights. As is common in most prior work, modelling the optimization variable, *i.e.* the scheduling decision, as a placement matrix does not capture the inter-task dependencies well [15]. A better approach is to use geometric modelling of the scheduling decisions, particularly as a graph, as it enables structure specific extraction of the system state information. Recently proposed Artificial Intelligence (AI) techniques, such as graph neural networks or graph encoders, capture such geometric data very well [16, 17]. One such network, called Gated Graph Convolution Network (GGCN) enables aggregation of the graph node information using convolution operations and message passing [16], making it suitable to model distributed computing network as a graph. This enables a more versatile optimization approach that also takes into account the task hierarchy and edge-cloud hierarchy, not considered in most prior work [14, 9]. We use a GGCN model as a surrogate of the QoS objective scores allowing us to swiftly run placement optimization. Such a surrogate model enables us to quickly get the QoS score for an input (scheduling decision) without actually executing it in the physical environment, saving us time and cost. Such deep surrogate models are commonly used in the literature [10, 13, 14].

Our contributions. In this work, we significantly extend our previous work [6] by proposing a ***H***olistic reso***U***rce ma***N***agemen***T*** technique for ***E***nergy-efficient cloud computing using a***R***tificial intelligence, called ***HUNTER***. The proposed method uses a GGCN network as a QoS surrogate to optimize the scheduling decision for a hybrid public-private cloud environment. It uses *performance to power ratio* as a heuristic to explore the scheduling search space that enables to significantly reduce scheduling time. In our previous work [6], we proposed a holistic management technique for cloud resources and established a relationship between replication and service consolidation to improve the energy-efficiency and cut the carbon footprint. However, our previous work did not deal with heterogeneous resources and dynamic workloads. In this work, we extend existing thermal and energy consumption models to also include the cooling overheads [7]. Further, to prevent excessive scheduling overheads, we use performance to power ratio as a heuristic to significantly reduce the time to converge to a scheduling decision [18]. To adapt in volatile scenarios, we periodically adjust the weights of the deep surrogate model using backpropagation.

The contributions of this research work are summarized as:

- We propose a novel energy-efficient resource management approach (HUNTER) that uses *GGCN as a deep surrogate model* for quick QoS estimation and three sustainability models, *viz*, thermal, energy and cooling.
- Extensive experiments on simulated (using the CloudSim [19] toolkit) and physical cloud testbeds (using the COSCO [14] framework) show that the proposed model outperforms state-of-the-art schedulers for sustainable computing.
- HUNTER gives the *best* energy consumption, SLA violation, cost and temperature by 11.90%, 35.41%, 53.86% and 3.47% respectively. HUNTER achieves this with 42.78% lower scheduling overheads compared to the best baseline.

The rest of the article is organized as follows. Section 2 overviews the related work. Section 3 presents the HUNTER scheduler. Performance evaluation is carried out in Section 4 with additional results and analysis in Section 5. Section 6 concludes the paper and proposes future work.

2. Related Work

A significant amount of research has been devoted to the area of resource management in cloud computing. Table 1 summarizes the comparison of HUNTER with existing works based on important key features. Given our scope of holistic management of resources particularly focusing on sustainability, we classify the state-of-the-art work into two main categories: 1) meta-heuristic methods (rows 1-3) and 2) reinforcement learning models (rows 4-8). The 'holistic' column represents whether the approach provides an end-to-end solution for scheduling, considering all parameters for sustainable cloud computing [6]. The 'dynamic' column represents whether the technique adapts on-the-fly for non-stationary environments.

Table 1: Comparison of HUNTER with related work (✓means that the corresponding feature is present).

Work	Holistic	Dynamic	Technical Approach	Sustainability Models			QoS and other Optimization Parameters				
				Energy	Thermal	Cooling	Temperature	Time	SLA Violation Rate	Cost	Energy
TOPSIS [20]	✓		Threshold Based	✓	✓		✓				✓
MALE [21]	✓		Memory Mapping	✓	✓		✓				✓
CRUZE [6]	✓		Cuckoo Optimization	✓	✓	✓	✓	✓	✓	✓	✓
MITEC [8]		✓	Genetic Algorithm	✓	✓		✓				✓
PADQN [12]		✓	Deep Q Learning	✓			✓	✓			✓
ANN [10]		✓	Neural Network	✓				✓			✓
SDAE-MMQ [13]		✓	Autoencoders	✓				✓	✓		✓
HDIC [9]	✓	✓	NARX Network	✓	✓		✓				✓
HUNTER	✓	✓	Surrogate Modelling	✓	✓	✓	✓	✓	✓	✓	✓

Meta-Heuristic Methods. Our previous work, CRUZE [6], aimed to reduce the total cloud energy consumption whilst maximizing reliability of the system. It utilizes efficient design models with respect to energy, reliability, capacity and cooling. To generate a scheduling decision, CRUZE uses a Cuckoo optimization approach. Another work, FECBench [22] provides performance interference prediction models for services of cloud providers to develop resource management techniques. The authors have constructed a process pipeline to construct multi-resource stressors using machine learning. To minimize prohibitive profiling costs, the authors have explored multi-dimensional resource metrics with minimal experimental runs using design of experiments (DoE) that significantly minimizes prohibitive profile cost. In similar efforts, the MALE algorithm [21] was recently introduced to minimize energy consumption in a cloud datacenter by reducing memory consumption and contention. This is achieved by mapping memory requirements of virtual machines to cloud hosts using a predefined best-fit criteria. Similarly, TOPSIS [20] presents a set of heuristics to significantly reduce energy consumption using thermal features that are recorded from cooling devices and servers. This approach uses a threshold based load-balancing technique to prevent thermal hotspots and minimize failures due to overheating. A similar work, MITEC [8] uses a genetic algorithm to optimize scheduling decisions and updates the energy and thermal models to tune the fitness scores for each allocation decision. Other works in this category propose autonomic cloud resource management mechanisms for the execution of batch and interactive workloads by leveraging the multiple resource layers and host heterogeneity to reduce energy consumption [23, 24, 25]. However, most methods in this category including CRUZE, Ella-W and GRANITE do not adapt in volatile settings. Still, we include the CRUZE and MITEC methods as a baseline in our experiments to represent this category.

Reinforcement Learning. In recent years, several machine learning (ML) based schedulers have been proposed that aim to optimize energy consumption of CDCs. Reinforcement learning (RL) is an sub-field within ML that models the system as an interactive environment using QoS parameters to dynamically modify the scheduling policy [26]. One of the most versatile RL techniques is the deep Q learning (DQL). Here, a deep neural network is used to estimate a long-term reward (commonly referred to as the Q value) for each state. Many recent works, for instance PADQN [12] and SDAE-MMQ [13], formulate the scheduling problem as a RL problem and utilize deep Q learning to produce task placement decisions [27]. Here, the decision is modelled as the state of the RL system with actions as task migration or allocation decisions. Each action changes the model state and gives a reward in the form of a QoS score. More advanced DQL based approaches use sophisticated neural networks to predict the Q values for each scheduling decision. SDAE-MMQ uses a stacked denoising autoencoder as a value network and MiniMax-Q instead of vanilla Q-learning [13]. Another work, HDIC [9], uses a nonlinear auto-regressive network with exogenous inputs (NARX) as a value network. Advanced neural models typically take a long time to train and infer Q values for large-scale state inputs. Other works directly use a deep neural network to produce a task allocation or migration decision [28]. For instance, a recent ANN approach uses an artificial neural network to produce a softmax output for each task [10]. Taking the argmax for each output gives us the scheduling decision for each task. The ANN is trained using a supervised learning framework with actions being rewarded using QoS metrics like energy consumption and execution time. Such approaches usually scale well with the number of tasks or hosts in the system, but are unable to capture inter-task dependencies to efficiently handle task placement. Moreover, Q-learning based methods are known to be slow to adapt in volatile settings [14]. We use the PADQN, ANN, SDAE-MMQ and HDIC methods as baselines in our experiments as these are empirically the best methods this category.

These approaches focus on particular perspectives of cloud resources management, e.g. computing or network. Unlike these works, our approach considers resource management in a holistic manner by considering energy, thermal and cooling characteristics whilst reducing scheduling and task migration overheads.

3. The HUNTER Scheduler

3.1. System Model

Figure 1 shows the system model considered in this work. Motivated from prior work [29, 14], we consider the following components of the CDC.

- **Cloud Users:** The users share workloads as jobs to the CDC (more details in Section 3.2). The data is collected

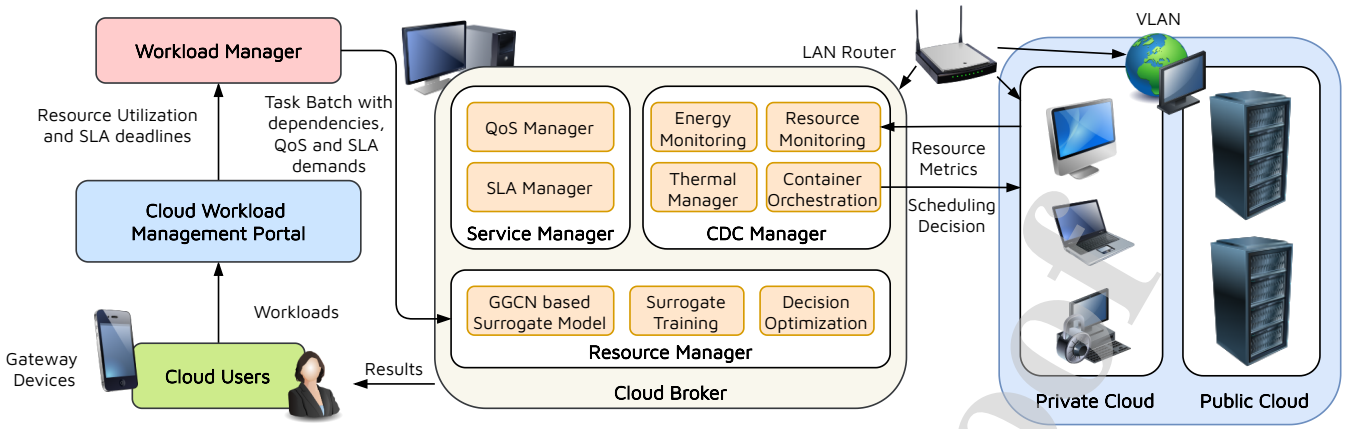


Figure 1: System Model of HUNTER

using IoT sensors and passed on to the CDC using gateway devices like smartphones and tablets [14].

- **Cloud Workload Management Portal:** A graphical user interface for cloud users to interact with the system for the submission of workloads along with their SLA and QoS details.
- **Workload Manager:** Initially, the workload manager processes all the incoming workloads. An admission controller realizes all workloads as container instances [6, 14].
- **Cloud Broker:** The central cloud server that allocates incoming jobs to various compute resources (cloud worker nodes). It consists of the following components:
  - *Service Manager:* Contains two elements, SLA and QoS managers that manage the heterogeneous cloud services while processing workloads. The QoS manager contains the information about QoS requirements for different workloads, while the SLA manager contains the information about an agreement signed between a cloud user and a provider based on QoS requirements.
  - *CDC Manager*: Continually monitors the resource utilization of all active tasks and hosts in the system. It also monitors the QoS parameters (including the energy and thermal characteristics of cloud hosts) and also performs the task allocation and migration. In this work, we assume tasks as container instances and task migration as the transfer and restoration of container checkpoints.
  - *Resource Manager:* Decides the schedule for each task in the system. The resource manager includes the sustainability models for energy, thermal and cooling parts of the CDC. For resource scheduling, the manager contains a GGCN based surrogate model that estimates QoS parameters. It performs training and on-the-fly tuning of the GGCN model to adapt in non-stationary settings. This manager also runs an exploration strategy that checks the QoS scores for a set of allocations and chooses the best one as the scheduling decision (more details in Section 3.5).
- **Cloud Hosts:** The cloud broker is connected to a heterogeneous set of worker nodes. Some nodes are present in the same Local Area Network (LAN) as the broker, called the private cloud. Others are present in a geographically distant location and connected using a virtual LAN (VLAN). As is common practice, we assume that private-cloud nodes are resource constrained but offer low latency services, and public-cloud nodes have abundant resources but have high communication latency.

The HUNTER scheduler resides as the Resource manager in the Cloud Broker, taking tasks as inputs from the Workload Manager (see Figure 1). HUNTER uses resource metrics from the Resource Monitor and executes scheduling decision through Container Orchestration (as tasks are realized as containers in our system).

3.2. Workload Model and Problem Formulation

As common in prior work, we assume that generating scheduling decisions is a discrete-time control problem [14, 30]. We divide the timeline into equal duration intervals, with the t-th interval denoted as I_t (starting from $i = 0$). We assume a fixed number of host machines and denote the set of cloud hosts by H. The workloads are in the form of jobs J_t, where each job $j_i \in J_t$ is composed of multiple tasks $j_i = \{t_0, \ldots, t_{|j_i|}\}$. There are no precedence constraints among tasks that belong to the same job, but the QoS metrics are calculated at the job level instead of the task level. Thus, it is important to consider inter-task dependencies while scheduling. All new jobs created at the interval I_t are denoted as N_t, with all active jobs being denoted as A_t. A job is considered to be active if at least one task of that job is being executed in the cloud environment. If no task of a job $j \in N_t$ can be allocated to a cloud node then it is added to a wait queue W_t. All created jobs that are not active and are not in the wait queue are considered to be completed and we can calculate their metrics like response time and SLA violation.

We consider the problem of maximizing the QoS objective score accumulated across all intervals in a bounded time experiment. We denote the QoS score for interval I_t by O_t and

consider a total n intervals in an experiment. We denote the utilization metrics of all hosts in interval I_{t-1} as U_t. Now using U_t, we need to predict a scheduling decision S_t. All tasks for jobs in $N_t \cup W_t \cup A_t$ are called feasible tasks. Thus the problem can be formulated as:

$$\begin{aligned} \underset{S_t}{\text{maximize}} \quad & \sum_{t=0}^{n} O_t \\ \text{subject to} \quad & \forall\, t, S_t : P_t \cup Q_t \rightarrow H, \\ & \forall\, t, P_t = \text{ set of feasible tasks in } N_t \cup W_t \cup A_t, \\ & \forall\, t, Q_t = \text{ set of active tasks in the system} \end{aligned} \tag{1}$$

In the rest of the discussion we consider these symbols only for a single interval and drop the t subscript for notational convenience.

3.3. Sustainability Models

In this work, to decouple the different aspects of sustainable resource management [6], we have designed three different models: energy, thermal and cooling. For completeness, we reproduce the formulae from our prior work [6], with necessary adaptations for the new formulation.

3.3.1. Energy Model

This model is designed to encapsulate all parameters related to the energy consumption, ranging from compute devices to the cooling components [31]. The total energy of a CDC is calculated as

$$E_{\text{Total}} = E_{\text{Computing}} + E_{\text{Cooling}}. \tag{2}$$

The computing system consists of hosts and its energy consumption includes that of the different components like CPU, RAM, disk, network and peripherals. Thus, $E_{\text{Computing}}$ can be defined as

$$E_{\text{Computing}} = E_{\text{Processor}} + E_{\text{Storage}} + E_{\text{Memory}} + E_{\text{Network}} + E_{\text{Extra}}. \tag{3}$$

Processor. Here, $E_{\text{Processor}}$ represents the processor's energy consumption, which is calculated by adding the idle and dynamic consumption of all cores. Thus,

$$E_{\text{Processor}} = \sum_{r=1}^{cores} E^r_{\text{dynamic}} + E^r_{\text{idle}}, \tag{4}$$

where E^r_{dynamic} and E^r_{idle} are the dynamic and idle energy consumption of the r-th core. Here, E_{dynamic} is calculated using

$$E_{\text{dynamic}} = \frac{E^{\text{linear}}_{\text{dynamic}} + E^{\text{non-linear}}_{\text{dynamic}}}{2}. \tag{5}$$

$E^{\text{linear}}_{\text{dynamic}}$ is calculated as

$$E^{\text{linear}}_{\text{dynamic}} = CV^2 f, \tag{6}$$

where C is CPU capacitance, f is CPU clock frequency, and V is CPU voltage. $E^{\text{non-linear}}_{\text{dynamic}}$ is calculated using

$$E^{\text{non-linear}}_{\text{dynamic}}(h_j) = \mu_1 \cdot U_j + \mu_2 \cdot U_j^2, \tag{7}$$

where μ_1 and μ_2 are non-linear model parameters and U_j is CPU utilization of host h_j.

Storage. E_{Storage} represents the energy consumption of storage devices to store data. The data read and write operations account for the energy consumption in such devices

$$E_{\text{Storage}} = E_{\text{ReadOperation}} + E_{\text{WriteOperation}} + E_{\text{Idle}}. \tag{8}$$

E_{Memory} represents the energy consumption of the main memory (RAM/DRAM) and cache memory (SRAM), which is calculated using

$$E_{\text{Memory}} = E_{\text{SRAM}} + E_{\text{DRAM}}. \tag{9}$$

Network. E_{Network} represents the energy consumption of networking equipment such as routers, switches and gateways, LAN cards, etc., and is calculated as

$$E_{\text{Network}} = E_{\text{Router}} + E_{\text{Switches}} + E_{\text{Gateways}} + E_{\text{LAN cards}}. \tag{10}$$

Peripherals. E_{Extra} represents the energy consumption of other parts, including the current conversion loss and others and is calculated as

$$E_{\text{Extra}} = E_{\text{motherboard}} + \sum_{f \in F} E^f_{\text{connector}} \tag{11}$$

where $E_{\text{motherboard}}$ is energy consumed by motherboard(s) and $\sum_{f \in F} E^f_{\text{connector}}$ is energy consumed by a connector (port) running at the frequency f, where the set of port frequencies is denoted by F.

3.3.2. Cooling Model

In the cooling model, E_{Cooling} denotes the energy consumed by cooling devices to maintain the temperature of a cloud datacenter, which is calculated using

$$E_{\text{Cooling}} = E_{\text{AC}} + E_{\text{Compressor}} + E_{\text{Fan}} + E_{\text{Pump}}, \tag{12}$$

where E_{AC} is the energy consumption of the air-conditioner inside the cloud-datacenter, $E_{\text{Compressor}}$ is the energy used by the compressor, E_{Fan} is that of the fans attached to the radiators and E_{Pump} is that of the pump within the all-in-one (AIO) water cooling solution.

3.3.3. Thermal Model

To design the thermal model, we use the Computer Room Air Conditioning (CRAC) model and RC (where R and C are thermal resistance (k/w) and heat capacity (j/k) of the host respectively) used as a time-constant to estimate the temperature of the CPU for each host (T_{cu}) [6, 7]. Thus,

$$T_{\text{cu}} = PR + Temp_{\text{inlet}} + T_{\text{initial}} * e^{-\text{RC}}, \tag{13}$$

where the inlet temperature ($Temp_{\text{inlet}}$) is calculated using CRAC model (T_{cu}); the RC model is used to calculate CPU temperature (T_{CPU}); P is the dynamic power of host. T_{inital} is the initial temperature of the CPU, which is taken as the ambient temperature of the datacenter [7].

The detailed description of the thermal model and the various metrics is given in our previous works [6].

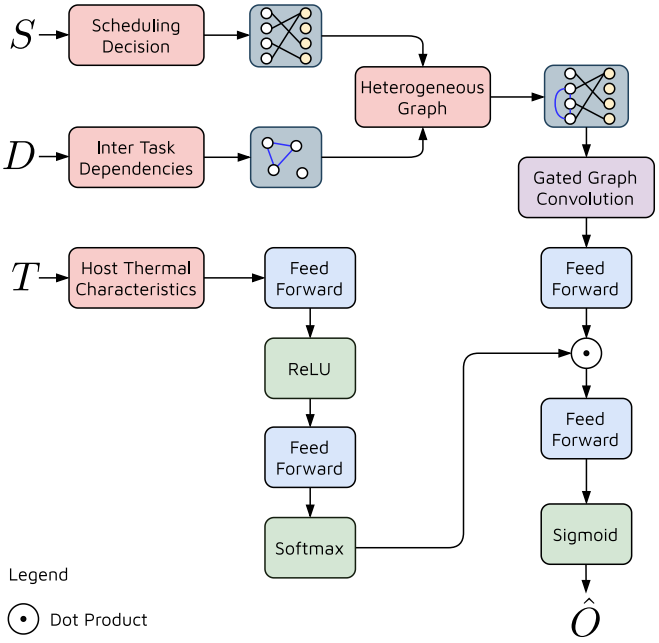


Figure 2: GGCN based surrogate model of HUNTER. The three inputs to the model and the graph structure are shown in red. Feed-forward and graph convolution operations are shown in blue and purple respectively. All activations are shown in green and all data structures are shown in grey.

3.4. GGCN based Surrogate Model

As described in Section 1, we model the inputs of our surrogate model using a geometric graph representation (see Figure 2). We form two graphs D and S. The former represents the inter task dependencies and the latter represents the bi-partite graph corresponding to the scheduling decision. $D = (V_D, E_D)$, where V_D denotes the tasks as nodes and E_D denotes the inter-task dependency in terms of the jobs they belong to as undirected edges. Each task has a feature vector corresponding to the Instructions per Second (IPS), RAM, Disk and Bandwidth consumption. The RAM, Disk and Bandwidth consumption also include the read and write speeds of such tasks. Each edge (t_p, t_q) in E_D is an unordered pair such that tasks t_p and t_q belong to the same job $j \in J$. $S = (V_S, E_S)$ is a bi-partite graph with nodes of two types: tasks and hosts. The edges of the graph $(t_p, h_r) \in E_S$ correspond to the allocation decision of the current state, where task t_p is allocated to host h_r. Similar to D, each task has a feature vector corresponding to the IPS, RAM, Disk and Bandwidth consumption. The feature vectors of the host machines consist of the IPS, RAM, Disk and Bandwidth consumption and capacities.

To perform graph convolutions, we combine S and D into a single heterogeneous graph, where the edge set now becomes $E_D \cup E_S$, such that each edge now also has an edge type (task dependence or allocation relation). This hetero-graph is then sent to the GGCN model to run convolution operations. The GGCN model executes convolutions across the edges of the graph where the convolution operations are weighted using a Gated Recurrent Unit (GRU). The convolution operations allow the model to share information across different tasks and

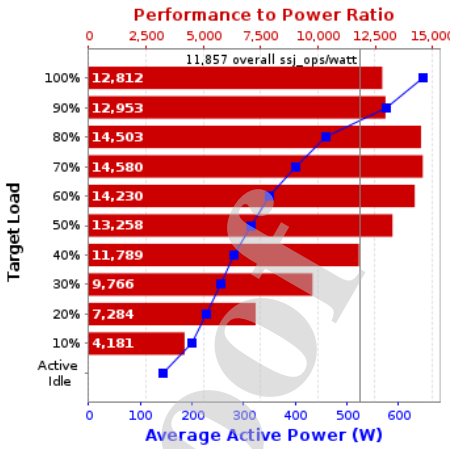


Figure 3: A sample power to load curve for a cloud server. Reproduced with permission. Source: SPEC benchmark power profile repository `https://www.spec.org/power_ssj2008/results/res2021q2/power_ssj2008-20210528-01098.html`

hosts whilst inferring a latent representation of the scheduling decision. This information sharing helps the downstream operations to explicitly leverage the inter-task dependencies and the implications of an allocation or migration decision on host utilization characteristics. The GRU based weighting allows the model to be flexible with respect to the extent to which the feature vectors of hosts and tasks should be combined. Formally, the message passing leads to *graph-to-graph* updates

$$\begin{aligned} r_i^0 &= \text{Tanh}\left(W\, e_i + b\right), \\ x_i^k &= \sum_{j \in n(i)} W^k r_j^{k-1}, \\ r_i^k &= \text{GRU}\left(r_i^{k-1}, x_i^k\right). \end{aligned} \tag{14}$$

Here, W, b are parameters of the feed-forward layer within the GGCN network, e_i is the feature-vector (described previously) of a node $i \in V_D \cup V_S$ in the heterogeneous graph and k varies from 1 to p. Also, the messages for task i are aggregated over one-step connected neighbors $n(i)$ over the p convolutions, resulting in an embedding r_i^p for each task node in the graph. The stacked representation for all tasks is represented as r^p. Convolutions across these edge types help as the dependence of task utilization characteristics with the allocated hosts is the maximum and much lower for hosts on which the task is not allocated. This allows the scheduling decision to properly manage task and hosts utilization characteristics while also considering the changing demands of tasks. We generate the graph encoding e^S by passing r^p through a feed-forward layer as

$$e^S = \text{FeedForward}(r^p). \tag{15}$$

We also capture the thermal-characteristics of the host machines in terms of their current temperatures (T_{cu}) and the power to load profile[10] (see Figure 3 for a sample performance

[10]We use Standard Performance Evaluation Corporation (SPEC) power consumption models to generate power to load performance curves of our cloud hosts. URL: `https://www.spec.org/power_ssj2008/`

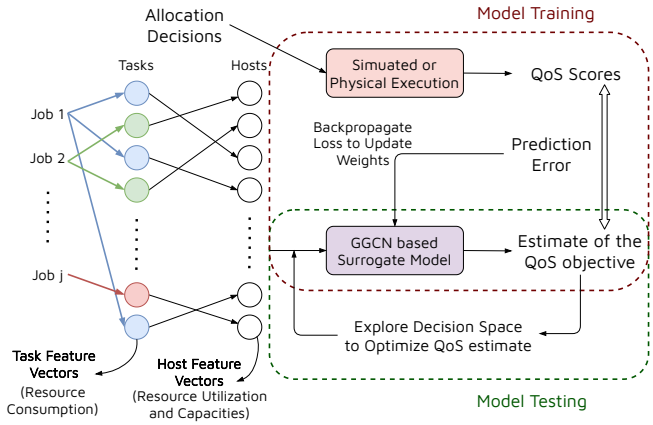


Figure 4: Graphical representation of the GGCN model.

to power profile). We model the thermal profile and current temperature of all hosts as vectors ($Temp$) and pass through a FeedForward network

$$e^T = \text{ReLU}(\text{FeedForward}(\text{ReLU}(\text{FeedForward}(Temp)))). \tag{16}$$

We then use Bahdanau style self-attention [32] to generate an estimate of the QoS objective. This allows the model to focus only on those hosts that can potentially become thermal hotspots.

$$\hat{O} = \text{Sigmoid}(\text{FeedForward}(e^S \cdot \text{Softmax}(e^T))). \tag{17}$$

The sigmoid operator allows us to generate an output within $[0, 1]$ to allow training with normalized QoS scores. The GGCN model is agnostic to the QoS objective in general; however, in our experiments we use energy, temperature and SLA violation rates to train and fine-tune the model (see Section 3.5). To train the GGCN model, we use the Mean-Square-Error (MSE) loss between the predicted and ground-truth QoS scores.

3.5. Using GGCN Model for Scheduling

We now present the *modus operandi* of the proposed HUNTER resource scheduling technique using the GGCN network as a surrogate model (summarized in Algorithm 1). Figure 4 shows a graphical representation of the HUNTER scheduler. The input for the scheduler is the resource utilization metrics of the hosts and tasks that need to be allocated or migrated and the thermal characteristics of the cloud hosts. These are obtained from the resource monitoring and the thermal management services in the cloud broker (see Figure 1). These metrics are then combined to generate the S and D graphs and the T vector as described in Section 3.4. The allocation in the S graph is obtained from the scheduling decision of the previous interval for I_t such that $t > 0$ and a random allocation otherwise (line 2 in Alg. 1). Thus, from the input $[S, D, T]$ we obtain an estimate of the QoS score $\hat{O}$. Now to generate a scheduling decision, we choose subsets of all tasks and hosts, each of size K. We sort the tasks based on the power consumption of their hosts, breaking ties using the CPU utilization and consider top K such tasks (line 4 in Alg. 1). Now, for each such high energy-resource

Algorithm 1 HUNTER Scheduler

Require:
Pre-trained GGCN model f
Convergence threshold ϵ
Consideration parameter K; Learning rate γ

1: **procedure** HUNTER(s)
2: $S, D, T \leftarrow ResourceMonitor()$
3: $O \leftarrow f([S, D, T])$
4: $Tasks \leftarrow$ get top K tasks based on power consumption
5: $Hosts \leftarrow$ get bottom K hosts based on performance to power ratio
6: **for** $(t \in Tasks)$ **do**
7: host $\leftarrow \arg\max_{h \in Hosts} f([S(t,h), D, T])$
8: **if** (allocation of t to h is feasible)
9: Allocate or migrate t to h.
10: **else**
11: Add t to wait queue.
12: **end if**
13: **end for**
14: $O = 1 - (\alpha \cdot AEC + \beta \cdot AT + \gamma \cdot SLAV)$
15: Datapoint $\leftarrow ([S, D, T], O)$
16: Backpropagate f using Datapoint and MSE loss
17: **end procedure**

consumption task, we choose a target host where it can be migrated. To do this, we sort all hosts in terms of the *performance to power ratio* and choose the bottom K hosts (line 5 in Alg. 1). Now, we consider all $K \times K$ combinations and choose the host h that maximizes the objective score estimate obtained from the surrogate model. We denote the updated S graph with task-host allocation (t, h) by $S(t, h)$ (line 7 in Alg. 1).

To choose the value of the K parameter, we leverage the network transfer constraints. Consider the router bandwidth (in MB/s) in a CDC to be denoted by B and the size of a scheduling interval (in seconds) by I_S. Also, let us denote the minimum size of a container in the CDC in MB by R. Then we define

$$K = \frac{B \times I_S}{R}. \tag{18}$$

Intuitively, this denotes the upper-bound of the number of tasks that can be migrated in the scheduling interval. If we try to migrate more than K tasks in any decision, the tasks would take more than I_S seconds, rendering the migration useless as new decisions are taken every I_S seconds. For a typical setting with bandwidth of 100-MB/s, scheduling interval of 300-seconds and container size of 3000-MB, $K = 10$ allowing us to check only 100 task-to-host allocations. This can be significantly more efficient than checking all task-host combinations, which may be in thousands.

The motivation behind using the performance-to-power ratio to sort hosts is as follows. Consider the sample performance to power ratio profile shown in Figure 3. This ratio indicates the amount of CPU computational performance we get for each watt of power consumed. It is apparent that this ratio is highest for 70% CPU load and reduces for higher or lower CPU loads. Most cloud servers have similar trends in their power profiles,

with a sweet spot around 70-80%. Higher CPU load can be easily avoided by capping the constraint checker in the scheduler to not allocate tasks to host with 80% CPU load. However, choosing the optimal target host for lower CPU loads is challenging due to the heterogeneity of the host power profiles. This key insight of using the performance-to-power ratio allows us to minimize the number of migrations as well as the time to explore different scheduling options.

Further, to adapt in volatile settings, at every scheduling interval we train the neural approximator using back-propagation. To do this, we obtain the latest QoS objective score from the QoS manager of the cloud broker and fine-tune the weights of the GGCN model by back-propagating the MSE loss between the predicted and true QoS scores. The ground-truth QoS score is obtained as (lines 14-16 in Alg. 1)

$$O = 1 - (\alpha \cdot AEC + \beta \cdot AT + \gamma \cdot SLAV), \tag{19}$$

where AEC, AT and $SLAV$ denote the average normalized energy consumption, average normalized temperature and SLA violation for the leaving tasks in the previous interval. Here, α, β, γ are convex-combination weights. To minimize the metrics of energy, temperature and SLA violations, we maximize O. Continuous training of the model allows it to quickly adapt to dynamic workloads and also consider changing scheduling decisions and use these as well to consider task migrations and allocations. As shown in line 8 in Alg. 1, for each container, we check if allocation to the host corresponding to the maximum QoS score of the GGCN model is feasible, if yes we allocate/migrate the container to this host else add it to the wait queue to be processed in the next interval.

Computational Complexity. Assuming that the inference of a deep neural network is an $O(1)$ operation, we provide the computational complexity in the Big-O notation. Assume $p = |P \cup Q|$ active tasks and $q = |H|$ hosts in the system. Selecting $K \leq q$ hosts and tasks from these sets based on the previously described metric is an $O(q \log K)$ and $O(p \log K)$ operation. Checking all $K \times K$ task to host allocations is a $O(K^2)$ operation. Overall, the computational complexity of checking the various allocation choices is $O(K^2 + q \log K + p \log K)$. This is significantly better than checking all possible task-host combinations, *viz*, $O(pq)$ in the typical case where $p > q$.

4. Performance Evaluation

We now describe how we evaluate the HUNTER scheduler and compare it against the state-of-the-art baselines: PADQN, CRUZE, MITEC, ANN, SDAE-MMQ and HDIC as described in Section 2.

4.1. Evaluation Setup

We have tested our proposed approach in both real and simulated cloud environments using the COSCO framework [14] and the CloudSim toolkit [19]. We keep the size of the scheduling interval as 5 minutes or 300 seconds and run our experiments for 100 scheduling intervals to generate QoS results. For statistical significance, we average over 5 runs. The first is a *physical setup* where we have used 10 Azure VMs in a distributed cloud setup as described below.

- **Private Cloud.** 6 Azure machines, four of type B2s (dual-core CPU with 4 GB RAM) and two of type B4ms (quad-core with 16 GB RAM). They were instantiated in the London, UK Azure datacenter.
- **Public Cloud.** 4 Azure machines, two of type B4ms (quad-core with 16 GB RAM) and two of type B8ms (octa-core with 32 GB RAM). They were instantiated in the Virginia, USA Azure datacenter.

We also tested on a *simulated* platform with 5 times the instances as described above to give a total of 50 hosts. The former allows more accurate testing of our approach while the latter allows large-scale experimentation. We use the SPEC power benchmarks to determine the energy consumption of the datacenters as done in prior work [14]. We used the R and C values in (13) as 0.5 and 0.03 based on prior work [33]. We followed the same implementation details that were used and described in previous work [14, 30, 34].

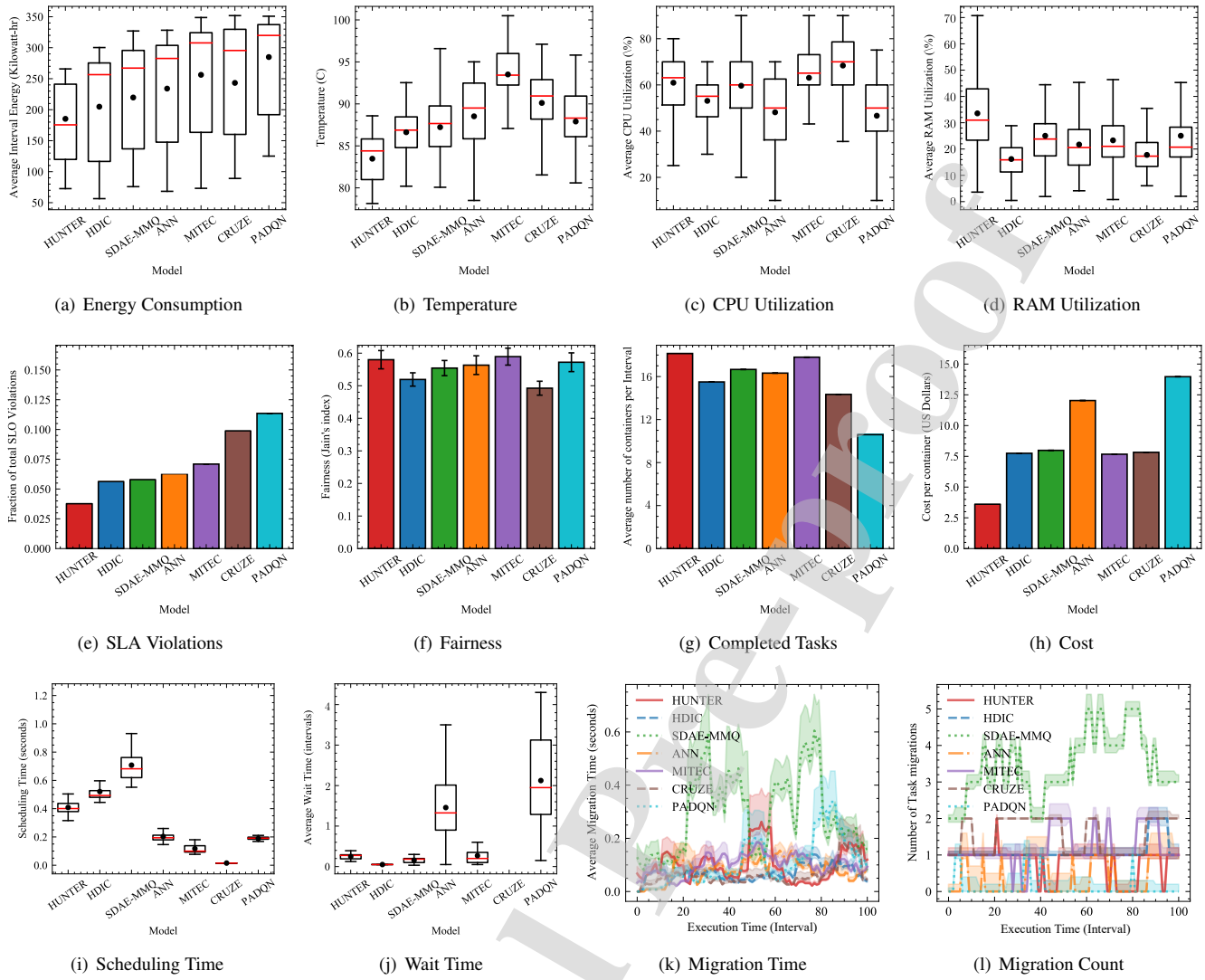
4.2. Workloads

For our physical experiments we use the *DeFog* benchmarking applications for their diverse and non-stationary workloads [35]. DeFog consists of various compute intensive AI applications like *Yolo*, *PocketSphinx* and *Aeneas*. The include workloads in the form of object detection in images, natural language processing, audio-text synchronization and speech recognition. We encapsulate these workloads as Docker containers to execute in our cloud servers. At the start of each scheduling interval we create $Poisson(\lambda)$ jobs with $\lambda = 1.2$. The jobs are sampled uniformly from the three applications of *Yolo*, *PocketSphinx* and *Aeneas*. We divide the input batch of each job into 3 to 5 parts and send them to separate containers (each container acts as a task).

In our simulated setup, we use the popular dynamic traces from the BitBrain dataset to emulate a large-scale execution [36]. The dataset consists of performance metrics of more than a thousand hosts in a heterogeneous CDC. These traces are collected from the Alibaba distributed datacenter, which is very popular for providing services to perform business computations and manage hosting of industrial applications. Insurers (Aegon), credit card operators (ICS) and many major banks (ING) are the main customers of this datacenter. Further, various financial computing applications (e.g. Algorithmics and Towers Watson) related to credit worthiness domain are hosted here. Moreover, traces are divided into two categories: *Rnd* and *fastStorage*. To allow diverse workloads, we use both traces in our experiments. These traces consist of time-series models of CPU, RAM, Disk and Bandwidth utilization characteristics. As in the physical setup, we create jobs using the the $Poisson(5)$ distribution with each job is sampled uniformly at random from the *Rnd* and *fastStorage* categories and has 3 to 5 tasks. the λ parameter is chosen based on prior work [14].

4.3. Model Training

The GGCN model takes as an input, the utilization matrix of the active tasks and the capacity matrix of the target hosts. This



(a) Energy Consumption (b) Temperature (c) CPU Utilization (d) RAM Utilization

(e) SLA Violations (f) Fairness (g) Completed Tasks (h) Cost

(i) Scheduling Time (j) Wait Time (k) Migration Time (l) Migration Count

Figure 5: Comparison of HUNTER against baselines on physical setup with 10 hosts

includes the metrics like CPU, RAM, Disk and Network Bandwidth. We also include the SLA deadline as part of the task utilization matrix. To train the model we first run a random scheduler to cover as much of the state space as possible. We run this for a 1000 scheduling intervals and create a dataset of the form $\{([S, D, T], O)\}$.

Details for Reproducibility: We pass the input through a 4 layer GGCN model with 64 nodes each and initialize the hidden state of the GRU by a zero vector. We use AdamW optimizer with a learning rate of 10^{-4} to train our model and use early-stopping as our convergence criterion [37]. All model training and experiments were performed on a system with configuration: Intel i7-10700K CPU, 64GB RAM, Nvidia GTX 1060 and Windows 11 OS.

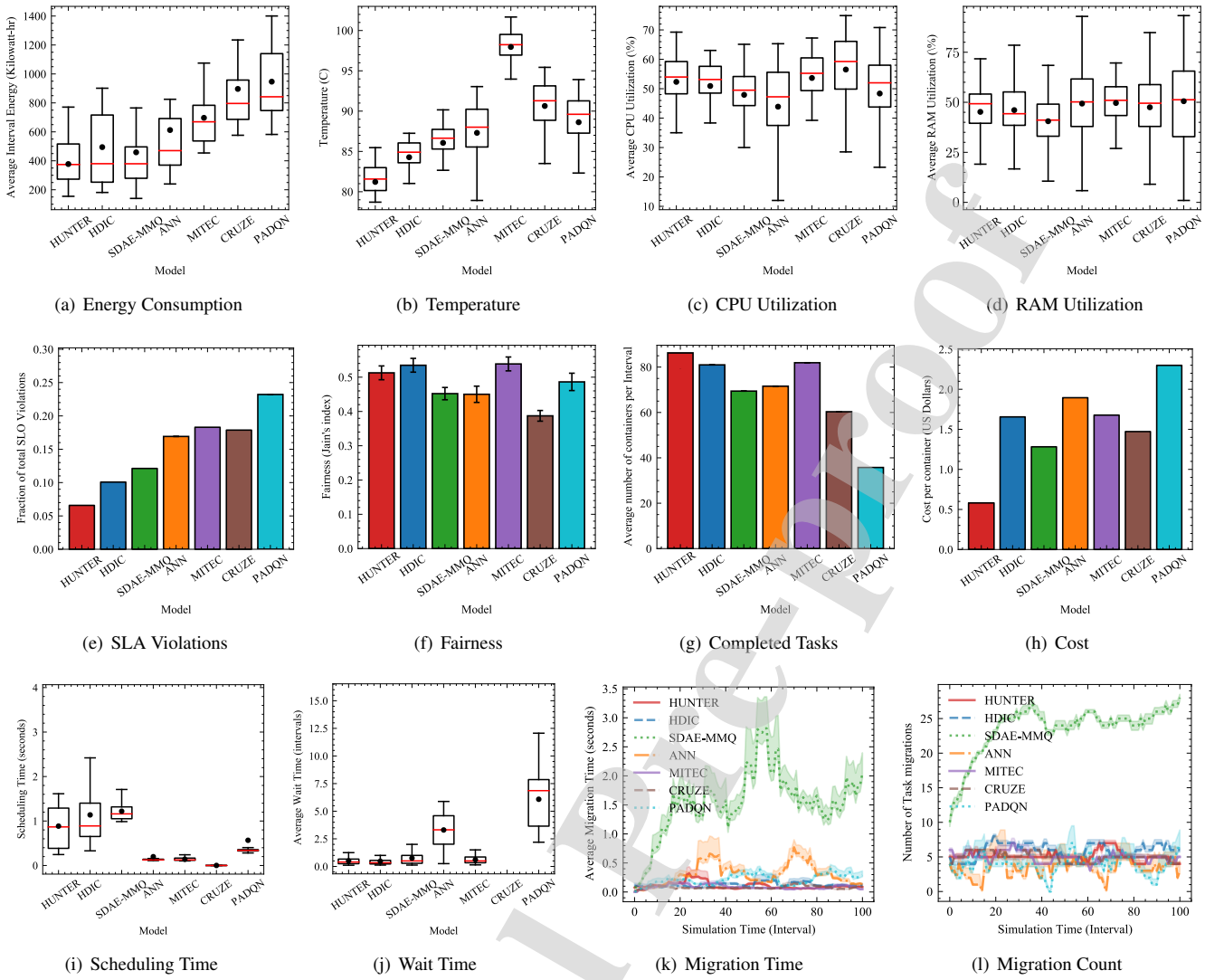
4.4. Evaluation Metrics

To compare the proposed HUNTER method against the baseline methods, we use the following metrics:

- *Energy consumption* given as E_{total} in (2).
- *SLA Violations* which is given as

$$\frac{\sum_i SLAV_i}{\sum_i j_i},$$

  where $SLAV_i$ is the 1 if SLA of job j_i is violated else 0.
- *Average Response Time* which is the mean response time for all completed jobs in an experiment
- *Datacenter Temperature* given by (13).
- *Cost* is given by $Time \times Price$. We use the Microsoft Azure pricing calculator to obtain the cost of execution per hour (in US Dollars) `https://azure.microsoft.com/en-gb/pricing/calculator/`.
- *Fairness* is given by the Jain's fairness index [14].
- *Scheduling Time* is the average time to generate a scheduling decision.
- *Wait Time* is given as the average time a job spends in the waiting queue.
- *Migration Time* is the average time a task spends in container migration.



(a) Energy Consumption (b) Temperature (c) CPU Utilization (d) RAM Utilization

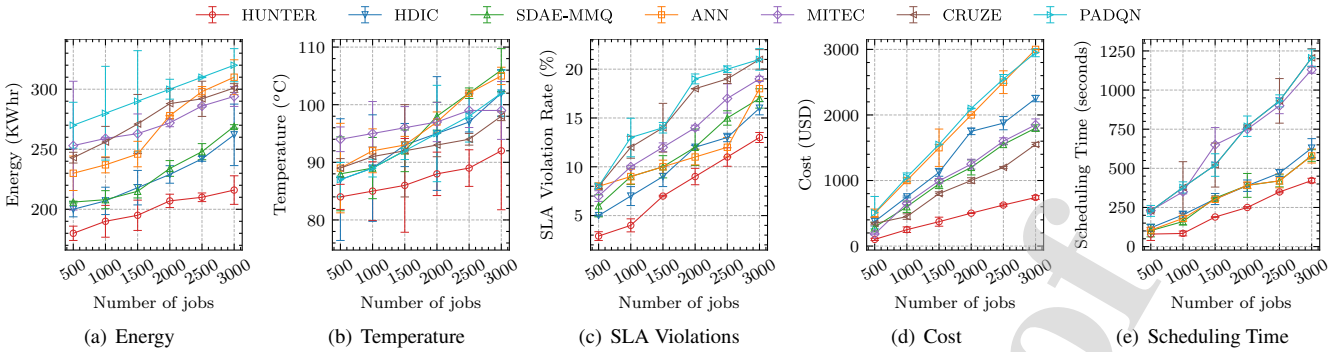
(e) SLA Violations (f) Fairness (g) Completed Tasks (h) Cost

(i) Scheduling Time (j) Wait Time (k) Migration Time (l) Migration Count

Figure 6: Comparison of HUNTER against baselines on simulator with 50 hosts

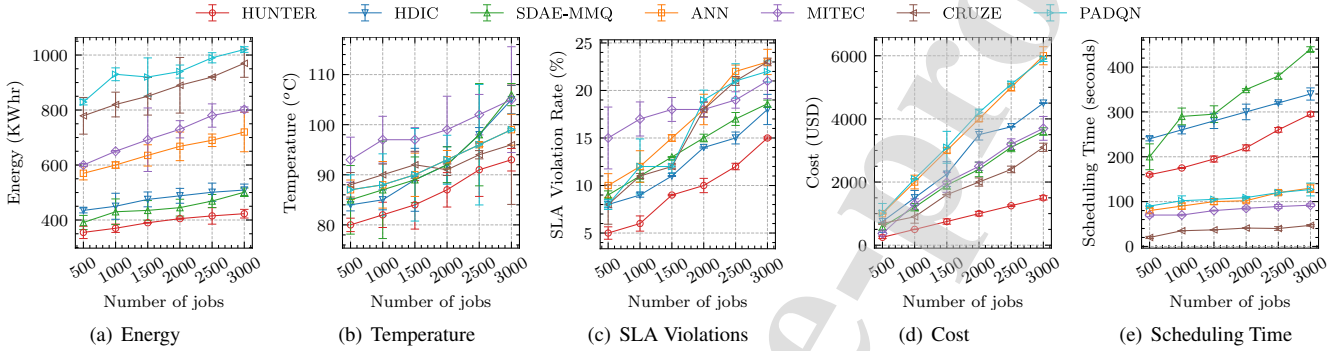
4.5. QoS Results

Figures 5 and 6 show the QoS parameters on the COSCO framework and CloudSim simulator respectively. Figures 5(a) and 6(a) show the energy consumption in a scheduling interval averaged over the number of tasks. Among the baselines, HDIC and SDAE-MMQ provide the most energy efficient policies. As we go from 10 to 50 hosts, the gaps in the energy consumption among schedulers increase, showing how robust the models are in minimizing energy consumption in large-scale setups. Overall, HUNTER gives the lowest energy consumption, reducing by up to 11.90% compared to the best baselines (HDIC in physical and SDAE-MMQ in simulated testbeds). This is because of the high number of tasks that complete execution in case of HUNTER and the use of the performance-to-power ratio (Figures 5(g) and 6(g)). Figures 5(b) and 6(b) show that HUNTER gives the lowest average temperature for both setups, giving a reduction of up to 3.47% compared to the best baselines, ~ 3°C (HDIC). This is because of the thermal-aware attention operation in the GGCN based surrogate model that allows HUNTER to emphasize scheduling for hosts that could act as thermal hostpots. Figures 5(c), 5(d), 6(d) and 6(d) show the CPU and RAM utilization of all models. All models have similar resource utilization metrics, with some cases where the RAM consumption in the HUNTER approach is quite high. Checking the execution traces in case of the HUNTER scheduler shows that this is due to the strict load-balancing rules to prefer keeping the number of containers in hosts to maintain the highest performance to load. Migrations based on such an approach can, in rare cases, lead to slight resource contention at the cost of minimizing energy or temperature. This is primarily when there are sudden spikes in task resource demands, having a cascading effect on other tasks running in the same host. Avoiding such cases is left as part of the future work. Figures 11(d) and 6(e) show that the proposed approach is able to reduce SLA violations by up to 35.41% compared to the best baselines (HDIC). This is primarily due to the accurate QoS objective prediction, allowing the model to minimize the SLA violation rates by checking the QoS estimate for several placement



(a) Energy (b) Temperature (c) SLA Violations (d) Cost (e) Scheduling Time

Figure 7: Sensitivity analysis of HUNTER and baselines with increasing number of workloads on physical setup with 10 hosts.



(a) Energy (b) Temperature (c) SLA Violations (d) Cost (e) Scheduling Time

Figure 8: Sensitivity analysis of HUNTER and baselines with increasing number of workloads on simulated setup with 50 hosts.

choices. Figures 5(f) and 6(f) show that all models have comparable fairness index values. Figures 11(e) and 6(h) show the average cost per task for each model. The HUNTER method has the lowest average cost giving up to 53.86% compared to the best baseline (SDAE-MMQ). PADQN has very high cost due to the excessive migration overheads as shown by Figures 5(k), 5(l), 6(k), 6(l). HUNTER like many other baselines has low wait times (Figs. 5(j) and 6(j)). Compared to the best baselines (HDIC and SDAE-MMQ), HUNTER has 42.78% lower scheduling time (Figs. 5(i) and 6(i)).

5. Analyses

5.1. Sensitivity Analysis

We now show how various models scale with the number of workloads. The results in the previous section were time bound, *i.e.*, for 100 scheduling intervals. Now we show how the QoS parameters vary with the number of workloads (see Figs. 7 and 8). Figures 7(a) and 8(a) show the variation in the consumption of energy with increasing number of jobs. HUNTER consumes up to 19.8% less as compared to best baseline models (HDIC and SDAE-MMQ). Overall, the rise in energy consumption with number of jobs for HUNTER is not as high as other baseline methods. This is because HUNTER uses the performance to power profiles of cloud hosts to maintain optimal performance while minimizing energy consumption. Figures 7(b) and 8(b) show the change in the temperature with the variation of job quantity. The value of temperature in HUNTER is 5.5% less than CRUZE because HUNTER uses CRAC-based cooling management [7] that avoids overloading and underloading of resources and can switch off idle resources automatically. Figures 7(c) and 8(c) show the change in the SLA violation rate with the variation of number of jobs. The value of SLA violation rate in HUNTER is up to 42.12% lower as compared to the HDIC baseline. Figures 7(d) and 8(d) show the change in the cost with the variation job count. HUNTER gives up to 63.75% less cost as compared to CRUZE and SDAE-MMQ. This is primarily due to the optimal performance to load management in the HUNTER scheduler. Figures 7(e) and 8(e) show the distribution of the scheduling time with the job count. There is a sharp increase in PADQN model as DQN scales poorly with time [14]. HUNTER has a higher scheduling time compared to the heuristic based baselines: CRUZE and MITEC. However, compared to the best baselines in terms of energy, temperature and cost, *i.e.*, SDAE-MMQ and HDIC, HUNTER has up to 56.12% lower scheduling times.

5.2. Comparison between Simulated and Physical Setups

We now compare the QoS metrics for the HUNTER approach for the simulated (CloudSim) and physical setups (COSCO) (see Figure 9). Clearly, all QoS parameters increase with the rise of the number of jobs. Figures 9(a), 9(b), 9(c), 9(d) and 9(e) show a performance comparison of simulated and physical setup for energy consumption, host temperature, SLA violation rate, cost and scheduling time. Naturally, we get higher energy consumption, SLA violation rates and scheduling times for the simulated setup as there are five times the number of hosts in

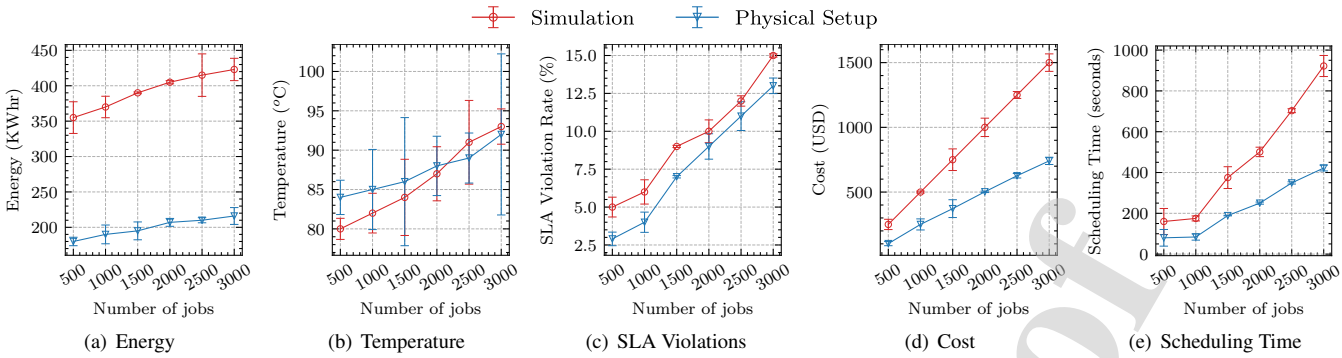


Figure 9: Comparison of QoS metrics using the HUNTER scheduler on simulated and physical platforms.

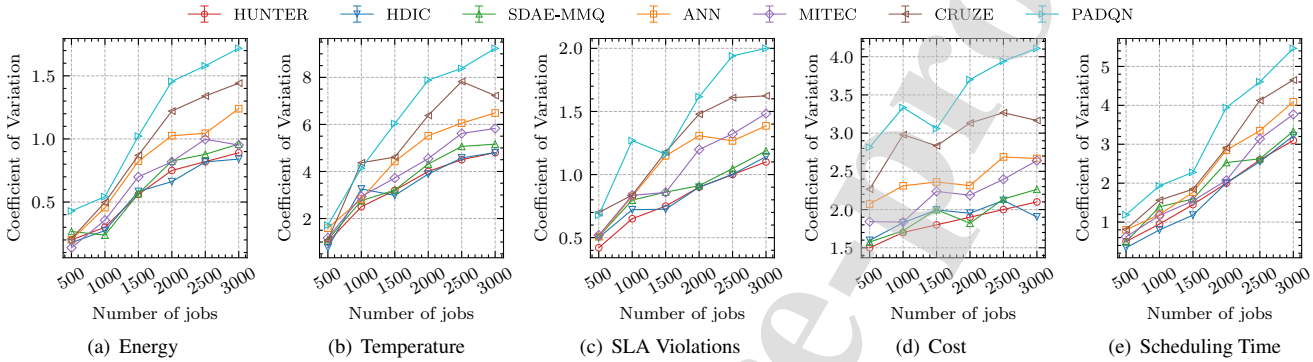


Figure 10: Coefficient of Variance for various metrics on the simulated platform.

Table 2: Comparison of training and inference times (in seconds) between HUNTER and baseline methods on simulated setup with 50 hosts.

Model	Training Time	Inference Time
HUNTER	908 ± 12	0.88 ± 0.13
HDIC	1193 ± 68	1.12 ± 0.28
SDAE-MMQ	2058 ± 102	1.17 ± 0.05
ANN	102 ± 4	0.20 ± 0.01
MITEC	-	0.19 ± 0.03
CRUZE	-	0.01 ± 0.01
PADQN	340 ± 81	0.23 ± 0.02

that of the physical setup. COSCO allows us to conduct more accurate experiments which give us less noisy fine-tuning as the model adapts to volatile workloads. This describes why the average rise in the temperature with the number of jobs is lower for the physical setup. Moreover, due to the imprecise computation of the resource utilization metrics for the tasks and hosts, the experiments of CloudSim simulator gives results that have high deviation from the ones conducted on the physical platform. Scalability wise we are able to show that HUNTER is able to scale well when the number of workloads or host machines is large.

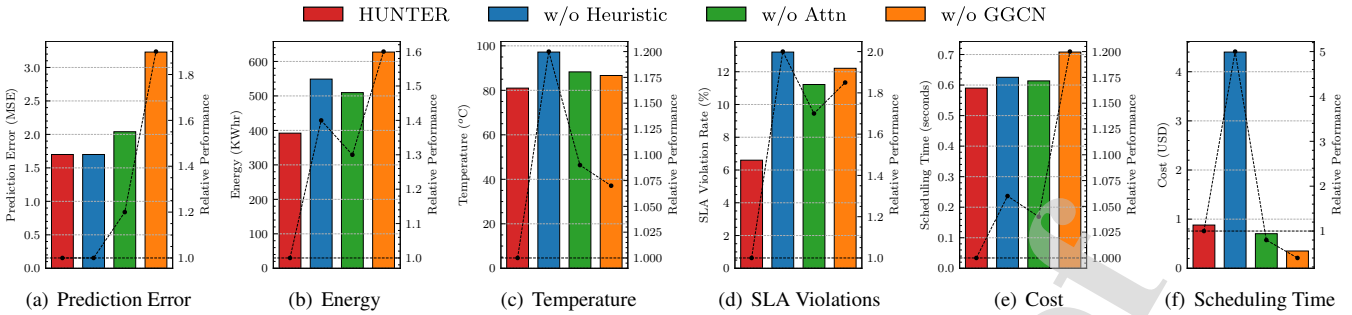
5.3. Analysis of Model Training and Inference Times

Table 2 compares the training and inference times of the HUNTER approach with the baseline methods. CRUZE and MITEC do not have any training overheads as they run online without training any AI model or neural network. We also test the training time for the GGCN model. Compared to various other prior works which rely on deep reinforcement learning (HDIC, SDAE-MMQ, ANN and PADQN) that take up to 2058 ± 102 seconds, HUNTER takes only 908 ± 12 seconds to train its model, giving a training overhead reduction of 55.87% compared to SDAE-MMQ and 23.88% compared to HDIC. This is negligible compare to the discrete time interval of 300 seconds used in our orchestration controllers and hence it is feasible to adopt the HUNTER approach in dynamically changing environments.

When comparing the inference times, the best baselines (HDIC and SDAE-MMQ) have a relatively high inference times of up to 1.17 seconds. HUNTER gives a scheduling time of 0.88 seconds, 24.78% lower than these baselines.

5.4. Statistical Analysis

The Coefficient of Variation (CoV) is used to analyse statistical significance of the experiments as it measures the distribution of the QoS metrics around the mean-value. Moreover, CoV gives an overall analysis of HUNTER's robustness to environment volatility. Figures 10(a), 10(b), 10(c), 10(d) and 10(e) show the CoV of energy consumption, temperature, SLA violation rate, execution cost and scheduling time with the increase in number of jobs. The range of CoV is (0.2–0.89%) for energy consumption, (0.42–1.1%) for SLA violation rate, (1.5–2.1%) for cost, (0.5–3.1%) for scheduling time and (1–6%) for temperature.



(a) Prediction Error (b) Energy (c) Temperature (d) SLA Violations (e) Cost (f) Scheduling Time

Figure 11: Ablation Analysis of different model components of HUNTER for the simulated setup with 50 hosts. The bar graphs show absolute values. The line graphs show the performance relative to HUNTER.

HUNTER has comparatively low CoV indicates that the model is able to handle dynamic workloads well and is robust enough to handle environment non-stationarity [6].

5.5. Ablation Analysis

To study the relative importance of each component of the model, we exclude every major one and observe how it affects the performance of the scheduler. An overview of this ablation analysis is given in Figure 11. First, we consider the HUNTER scheduler without the top-K heuristic and check all task-to-host allocations (*w/o Heuristic* model). Clearly, this gives a much higher scheduling time (Fig. 11(f)) and has worse effect on the other QoS metrics due to its high overheads. Second, we consider a model without the thermal-aware attention, *i.e.*, we only use GGCN part of the deep surrogate model (*w/o Attn* model). Here we see that the average temperature increases significantly, also impacting energy and cost. The other model we consider is replacing the GGCN network with a completely feed-forward one (*w/o GGCN* model). Here we see a significant increase in the MSE prediction error (Fig. 11(a)) leading to higher temperature, cost, SLA violations and energy consumption.

6. Conclusions and Future Work

In this paper, we have proposed a Gated Graph Convolution Network (GGCN) based holistic resource management scheduling technique called HUNTER. Our scheduler enables energy-efficient utilization of cloud servers and reduces thermal hotspots. HUNTER achieves this by adding cooling specific energy and temperature models, unseen in previous approaches. Further, using a GGCN based deep surrogate model allows HUNTER to quickly generate QoS estimates, avoiding significant costs in testing various scheduling decisions. HUNTER uses performance with load as a heuristic to effectively balance the load on cloud hosts, giving maximum compute power while reducing energy consumption. This heuristic also allows HUNTER to quickly explore the scheduling search space and quickly converge to a decision. Extensive experiments on both physical and simulated testbeds show that HUNTER outperforms baselines in most QoS metrics. Furthermore, the small values of the coefficient of variation of energy and temperature indicate that HUNTER is efficient in resource management while handling dynamic workloads. HUNTER optimizes five key performance parameters, *viz*, temperature, energy consumption, cost, SLA violation and time. The experiments demonstrate that the HUNTER performs better than existing AI based (HDIC, SDAE-MMQ, ANN and PADQN) and heuristic algorithm (CRUZE and MITEC) based resource schedulers.

This work can be extended by factoring in parameters that relate to scalability, security and reliability and their energy ramifications. Future work may also consider how cooling management can be further enhanced by capturing domain specific tactics for cooling; IoT and Fog/Edge computing reliant domains such as agriculture, healthcare and smart homes are among the candidate application domains to consider. Currently, HUNTER only decides the appropriate placement decisions for tasks; however it can be extended to also decide the AC or fan settings in the cases of deadline constrained or bursty workloads. Finally, HUNTER can use the concept of serverless edge computing to effectively scale applications.

Software Availability

The code is available at `https://github.com/imperial-qore/COSCO/tree/ggcn`. The Docker images used in the experiments are available at `https://hub.docker.com/u/shreshthtuli`.

CRediT authorship contribution statement

Shreshth Tuli: Conceptualization, Data curation, Investigation, Methodology, Software, Visualization, Validation, Formal analysis, Writing - original draft. **Sukhpal Singh Gill:** Methodology, Software, Investigation, Visualization, Validation, Writing - original draft. **Minxian Xu:** Writing - review & editing. **Peter Garraghan:** Writing - review & editing. **Rami Bahsoon:** Writing - review & editing. **Schahram Dustdar:** Writing - review & editing. **Rizos Sakellariou:** Writing - review & editing. **Omer Rana:** Writing - review & editing. **Rajkumar Buyya:** Writing - review & editing. **Giuliano Casale:** Supervision, Visualization, Writing - review & editing. **Nicholas R. Jennings:** Conceptualization, Writing - original draft, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Shreshth Tuli is supported by the President's Ph.D. Scholarship at the Imperial College London. This research work is partially supported by the EPSRC Research Grant (EP/V007092/1). The authors thank Muhammad Hassaan Anwar for his suggestions at the initial stages of this work.

References

[1] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, K. Pentikousis, Energy-efficient cloud computing, The computer journal 53 (7) (2010) 1045–1051.

[2] J. Shuja, A. Gani, S. Shamshirband, R. W. Ahmad, K. Bilal, Sustainable cloud data centers: a survey of enabling techniques and technologies, Renewable and Sustainable Energy Reviews 62 (2016) 195–214.

[3] K. W. Chun, H. Kim, K. Lee, A study on research trends of technologies for Industry 4.0; 3D printing, artificial intelligence, big data, cloud computing, and internet of things, in: Advanced multimedia and ubiquitous engineering, Springer, 2018, pp. 397–403.

[4] P. Wankhede, M. Talati, R. Chinchamalatpure, Comparative study of cloud platforms-Microsoft Azure, Google Cloud Platform and Amazon EC2, Journal of Research in Engineering and Applied Sciences.

[5] E. Pakbaznia, 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.

[6] S. S. Gill, P. Garraghan, V. Stankovski, G. Casale, R. K. Thulasiram, S. K. Ghosh, K. Ramamohanarao, R. Buyya, Holistic resource management for sustainable and reliable cloud computing: An innovative solution to global challenge, Journal of Systems and Software 155 (2019) 104–129.

[7] M. T. Chaudhry, T. C. Ling, A. Manzoor, S. A. Hussain, J. Kim, Thermal-aware scheduling in green data centers, ACM Computing Surveys (CSUR) 47 (3) (2015) 1–48.

[8] A. Akbari, A. Khonsari, S. M. Ghoreyshi, Thermal-aware virtual machine allocation for heterogeneous cloud data centers, Energies 13 (11) (2020) 2880.

[9] S. MirhoseiniNejad, G. Badawy, D. G. Down, Holistic thermal-aware workload management and infrastructure control for heterogeneous data centers using machine learning, Future Generation Computer Systems 118 (2021) 208–218.

[10] M. Sharma, R. Garg, An artificial neural network based approach for energy efficient task scheduling in cloud data centers, Sustainable Computing: Informatics and Systems 26 (2020) 100373.

[11] N. Fareghzadeh, M. A. Seyyedi, M. Mohsenzadeh, Toward holistic performance management in clouds: taxonomy, challenges and opportunities, The Journal of Supercomputing 75 (1) (2019) 272–313.

[12] Y. Ran, H. Hu, X. Zhou, Y. Wen, DeepEE: Joint optimization of job scheduling and cooling control for data center energy efficiency using deep reinforcement learning, in: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2019.

[13] B. Wang, F. Liu, W. Lin, Energy-efficient vm scheduling based on deep reinforcement learning, Future Generation Computer Systems.

[14] S. Tuli, S. Poojara, S. N. Srirama, G. Casale, N. Jennings, COSCO: Container Orchestration using Co-Simulation and Gradient Based Optimization for Fog Computing Environments, IEEE Transactions on Parallel and Distributed Systems.

[15] Z. Zhao, G. Verma, C. Rao, A. Swami, S. Segarra, Distributed scheduling using graph neural networks, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 4720–4724.

[16] L. Ruiz, F. Gama, A. Ribeiro, Gated graph recurrent neural networks, IEEE Transactions on Signal Processing 68 (2020) 6303–6318.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems 32 (1) (2020) 4–24.

[18] Y. Zhang, X. Wang, X. Liu, Y. Liu, Ł. Zhuang, F. Zhao, Towards better cpu power management on multicore smartphones, in: Proceedings of the Workshop on Power-Aware Computing and Systems, 2013, pp. 1–5.

[19] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and experience 41 (1) (2011) 23–50.

[20] E. Arianyan, H. Taheri, S. Sharifian, Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers, Computers & Electrical Engineering 47 (2015) 222–240.

[21] B. Liang, X. Dong, Y. Wang, X. Zhang, Memory-aware resource management algorithm for low-energy cloud data centers, Future Generation Computer Systems 113 (2020) 329–342.

[22] Y. D. Barve, S. Shekhar, A. Chhokra, S. Khare, A. Bhattacharjee, Z. Kang, H. Sun, A. Gokhale, FECBench: A holistic interference-aware approach for application performance modeling, in: 2019 IEEE International Conference on Cloud Engineering (IC2E), IEEE, 2019, pp. 211–221.

[23] M. S. A. Shuvo, M. A. R. Munna, S. Sarker, T. Adhikary, M. A. Razzaque, M. M. Hassan, G. Aloi, G. Fortino, Energy-efficient scheduling of small cells in 5G: A meta-heuristic approach, Journal of Network and Computer Applications 178 (2021) 102986.

[24] A. A. Butt, S. Khan, T. Ashfaq, S. Javaid, N. A. Sattar, N. Javaid, A cloud and fog based architecture for energy management of smart city by using meta-heuristic techniques, in: 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), IEEE, 2019, pp. 1588–1593.

[25] G. G. Kumar, P. Vivekanandan, Energy efficient scheduling for cloud data centers using heuristic based migration, Cluster Computing 22 (6) (2019) 14073–14080.

[26] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018.

[27] D. Zeng, L. Gu, S. Pan, J. Cai, S. Guo, Resource management at the network edge: A deep reinforcement learning approach, IEEE Network 33 (3) (2019) 26–33.

[28] J. N. Witanto, H. Lim, M. Atiquzzaman, Adaptive selection of dynamic vm consolidation algorithm using neural network for cloud resource management, Future generation computer systems 87 (2018) 35–42.

[29] A. Madej, N. Wang, N. Athanasopoulos, R. Ranjan, B. Varghese, Priority-based fair scheduling in edge computing, in: 2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC), IEEE, 2020, pp. 39–48.

[30] D. Basu, X. Wang, Y. Hong, H. Chen, S. Bressan, Learn-as-you-go with megh: Efficient live migration of virtual machines, IEEE Transactions on Parallel and Distributed Systems 30 (8) (2019) 1786–1801.

[31] S. MirhoseiniNejad, H. Moazamigoodarzi, G. Badawy, D. G. Down, Joint data center cooling and workload management: A thermal-aware approach, Future Generation Computer Systems 104 (2020) 174–186.

[32] D. Bahdanau, K. H. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: 3rd International Conference on Learning Representations, ICLR 2015, 2015.

[33] M. Wolf, The physics of computing, Elsevier, 2016.

[34] K. Alwasel, D. N. Jha, E. Hernandez, D. Puthal, M. Barika, B. Varghese, S. K. Garg, P. James, A. Zomaya, G. Morgan, et al., IoTsim-SDWAN: A simulation framework for interconnecting distributed datacenters over software-defined wide area network, Journal of Parallel and Distributed Computing 143 (2020) 17–35.

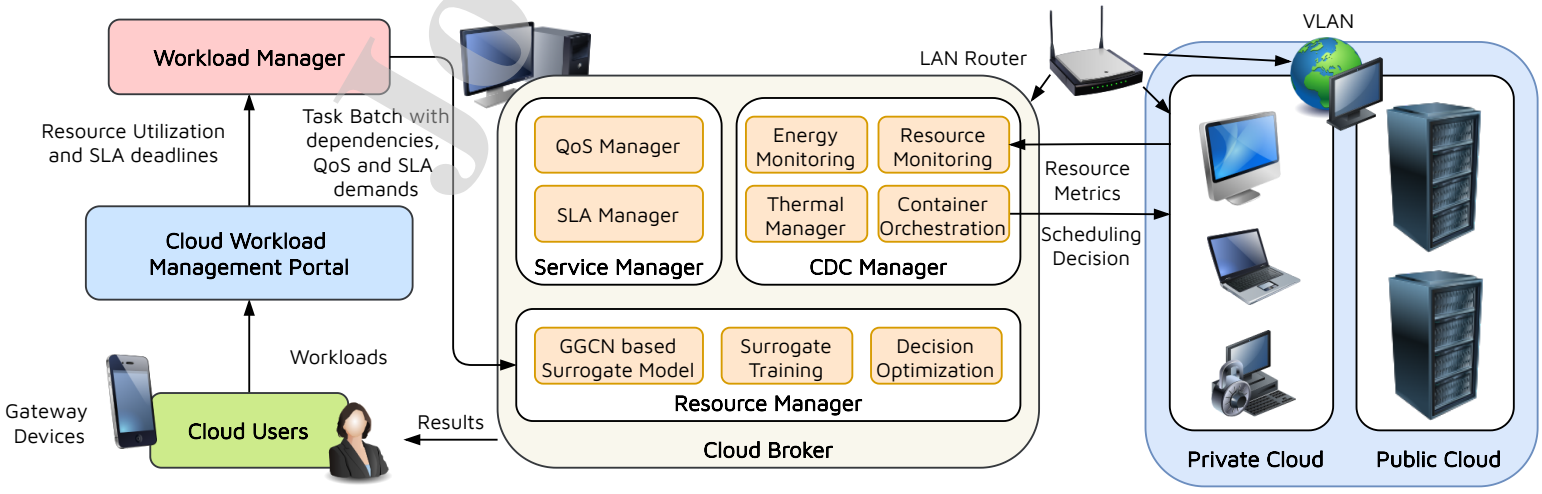
[35] J. McChesney, N. Wang, A. Tanwer, E. de Lara, B. Varghese, Defog: fog computing benchmarks, in: Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, 2019, pp. 47–58.

[36] S. Shen, V. Van Beek, 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.

[37] I. Loshchilov, F. Hutter, Decoupled weight decay regularization, in: International Conference on Learning Representations, 2018.

Highlights

- Artificial Intelligence based holistic resource management approach called HUNTER
- HUNTER is a thermal and energy-aware approach for resource scheduling in CDCs
- HUNTER manages resources (servers, memory, storage, network, cooling) holistically
- HUNTER is tested using a simulated and a physical cloud environment
- HUNTER optimizes energy consumption, SLA violation, time, cost and temperature



Author's Biography

Shreshth Tuli is a President's Ph.D. Scholar at the Department of Computing, Imperial College London, UK. Prior to this he was an undergraduate student at the Department of Computer Science and Engineering at Indian Institute of Technology - Delhi, India. He has worked as a visiting research fellow at the CLOUDS Laboratory, School of Computing and Information Systems, the University of Melbourne, Australia. He is a national level Kishore Vaigyanik Protsahan Yojana (KVPY) scholarship holder from the Government of India for excellence in science and innovation. His research interests include Internet of Things (IoT), Fog Computing and Deep Learning.

Sukhpal Singh Gill is a Lecturer (Assistant Professor) in Cloud Computing at School of Electronic Engineering and Computer Science, Queen Mary University of London, UK. Prior to this, Dr. Gill has held positions as a Research Associate at the School of Computing and Communications, Lancaster University, UK and also as a Postdoctoral Research Fellow at CLOUDS Laboratory, The University of Melbourne, Australia. Dr. Gill is serving as an Associate Editor in IET Networks Journal. His research interests include Cloud Computing, Fog Computing, Software Engineering, Internet of Things and Healthcare. For further information, please visit http://www.ssgill.me

Minxian Xu is currently an assistant professor at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He received the BSc degree in 2012 and the MSc degree in 2015, both in software engineering from University of Electronic Science and Technology of China. He obtained his PhD degree from the University of Melbourne in 2019. His research interests include resource scheduling and optimization in cloud computing. He has coauthored 20+ peer-reviewed papers published in prominent international journals and conferences, such as ACM Computing Surveys, IEEE Transactions on Sustainable Computing, IEEE Transactions on Automation Science and Engineering, Journal of Parallel and Distributed Computing, Concurrency and Computation: Practice and Experience, International Conference on Service-Oriented Computing. His Ph.D. Thesis was awarded the 2019 IEEE TCSC Outstanding Ph.D. Dissertation Award. More information can be found at: minxianxu.info.

.

Peter Garraghan is a Lecturer (Assistant Professor) and EPSRC Fellow in Distributed Systems. His research expertise is empirically studying and designing high performance, resilient, and sustainable distributed systems at scale (Cloud computing, Machine Learning systems, Datacentres, core network infrastructure) in the face of societal and environmental change. His research places strong emphasis on conducting analysis, design, and evaluation via experimentation on systems both in laboratory and production. Peter has published over 50 articles, has industrial experience building large-scale production distributed systems, and has worked and collaborated internationally with the likes of Alibaba Group, Microsoft, BT, STFC, CONACYT, and the UK datacenter and IoT industry. He is the recipient of the prestigious EPSRC Early-career Fellowship (2021 - 2025), and his research on sustainable computing and future AI systems has featured in the media including the BBC and the Daily Mail.

Rami Bahsoon is a Senior Lecturer (Associate Professor) at the School of Computer Science, University of Birmingham, UK. Bahsoon's research is in the area of software architecture, cloud and services software engineering, self-aware software architectures, self-adaptive and managed software engineering, economics-driven software engineering and technical debt management in software. He co-edited four books on Software Architecture, including Economics-Driven Software Architecture; Software Architecture for Big Data and the Cloud; Aligning Enterprise, System, and Software Architecture. He was a Visiting Scientist at the Software Engineering Institute (SEI), Carnegie Mellon University, USA (June-August 2018) and was the 2018 Melbourne School of Engineering (MSE) Visiting Fellow of The School of Computing and Information Systems, the University of Melbourne (August to Nov 2018). He holds a PhD in Software Engineering from University College London (2006) and was MBA Fellow in Technology at London Business School (2003–2005). He is a fellow of the Royal Society of Arts and Associate Editor of IEEE Software - Software Economies.

Schahram Dustdar is Full Professor of Computer Science heading the Research Division of Distributed Systems at the TU Wien, Austria. He holds several honorary positions: University of California (USC) Los Angeles; Monash University in Melbourne, Shanghai University, Macquarie University in Sydney, and University of Groningen (RuG), The Netherlands (2004-2010). From Dec 2016 until Jan 2017 he was a Visiting Professor at the University of Sevilla, Spain and from January until June 2017 he was a Visiting Professor at UC Berkeley, USA. From 1999 - 2007 he worked as the co-founder and chief scientist of Caramba Labs Software AG in Vienna (acquired by Engineering NetWorld AG), a venture capital co-funded software company focused on software for collaborative processes in teams. Caramba Labs was nominated for several (international and national) awards: World Technology Award in the category of Software (2001); Top-Startup companies in Austria (Cap Gemini Ernst & Young) (2002); MERCUR Innovation award of the Austrian Chamber of Commerce (2002). He is founding co-Editor-in-Chief of the new ACM Transactions on Internet of Things (ACM TIoT) as well as Editor-in-Chief of Computing (Springer). He is an Associate Editor of IEEE Transactions on Services Computing, IEEE Transactions on Cloud Computing, ACM Transactions on the Web, and ACM Transactions on Internet

Technology, as well as on the editorial board of IEEE Internet Computing and IEEE Computer. Dustdar is recipient of the ACM Distinguished Scientist award (2009), the IBM Faculty Award (2012), an elected member of the Academia Europaea: The Academy of Europe, where he is chairman of the Informatics Section, as well as an IEEE Fellow (2016).

Rizos Sakellariou obtained his PhD from the University of Manchester in 1997. Since then he held positions with Rice University and the University of Cyprus, while currently he is with the University of Manchester leading a laboratory that carries out research in High-Performance, Parallel and Distributed systems, which over the last 10 years has hosted more than 30 doctoral students, researchers and long-term visitors. Rizos has carried out research on a number of topics related to parallel and distributed computing, with an emphasis on problems stemming from efficient resource utilization and workload allocation and scheduling issues. He has published over 140 research papers, His research has been supported by several UK and EU projects and has been on the Program Committee of over 150 conferences and workshops. He values collaboration and a strong work ethic.

Omer Rana is a Professor of Performance Engineering in School of Computer Science & Informatics at Cardiff University and Deputy Director of the Welsh e-Science Centre. He holds a PhD from Imperial College. His research interests extend to three main areas within computer science: problem solving environments, high performance agent systems and novel algorithms for data analysis and management. Moreover, he leads the Complex Systems research group in the School of Computer Science & Informatics and is director of the "Internet of Things" Lab, at Cardiff University. He has published over 310 papers in peer-reviewed international conferences and journals. He serves on the Editorial Board of IEEE Transactions on Parallel and Distributed Systems, ACM Transactions on Internet Technology, and ACM Transactions on Autonomous and Adaptive Systems. He has served as a Co-Editor for a number of journals, including Concurrency: Practice and Experience (John Wiley), IEEE Transactions on System, Man, and Cybernetics: Systems, and IEEE Transactions on Cloud Computing.

Rajkumar Buyya is a Redmond Barry Distinguished Professor and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He has authored over 850 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. He is one of the highly cited authors in computer science and software engineering worldwide (h-index=149, g-index=322, 116,500+ citations). "A Scientometric Analysis of Cloud Computing Literature" by German scientists ranked Dr. Buyya as the World's Top-Cited (#1) Author and the World's Most-Productive (#1) Author in Cloud Computing. Dr. Buyya has led the establishment and development of key community activities, including serving as foundation Chair of the IEEE Technical Committee on Scalable Computing and five IEEE/ACM conferences. These contributions and international research leadership of Dr. Buyya are recognized through the award of "2009 IEEE Medal for Excellence in Scalable Computing" from the IEEE Computer Society TCSC. He served as the founding Editor-in-Chief of the IEEE Transactions on Cloud Computing. He is currently serving as Co-Editor-in-Chief of Journal of Software: Practice and Experience, which was established 50+ years ago. For further information on Dr. Buyya, please visit his cyberhome: www.buyya.com

Giuliano Casale joined the Department of Computing at Imperial College London in 2010, where he is currently a Reader. Previously, he worked as a research scientist and consultant in the capacity planning industry. He teaches and does research in performance engineering and cloud computing, topics on which he has published more than 100 refereed papers. He has served on the technical program committee of over 80 conferences and workshops and as co-chair for several conferences in the area of performance and reliability engineering, such as ACM SIGMETRICS/Performance and IEEE/IFIP DSN. His research work has received multiple awards, recently the best paper award at ACM SIGMETRICS. He serves on the editorial boards of IEEE TNSM and ACM TOMPECS and as current chair of ACM SIGMETRICS.

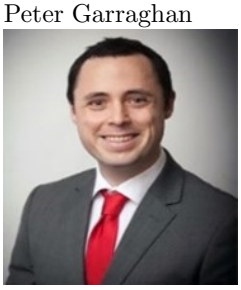
Nicholas R. Jennings is the Vice-Provost for Research and Enterprise and Professor of Artificial Intelligence at Imperial College London. He is an internationally-recognised authority in the areas of AI, autonomous systems, cyber-security and agent-based computing. He is a member of the UK government's AI Council, the governing body of the Engineering and Physical Sciences Research Council, and chair of the Royal Academy of Engineering's Policy Committee. Before Imperial, he was the UK's first Regius Professor of Computer Science (a post bestowed by the monarch to recognise exceptionally high quality research) and the UK Government's first Chief Scientific Advisor for National Security.

Shreshth Tuli

Sukhpal Singh Gill

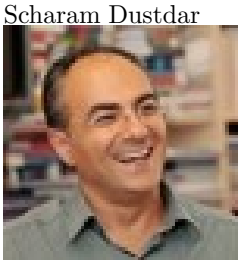
Minxian Xu

Peter Garraghan



Rami Bahsoon

Scharam Dustdar



Rizos Sakellariou

Omer Rana

Rajkumar Buyya

Giuliano Casale

Nicholas R. Jennings