# A Deep Reinforcement Learning Approach to Resource Management in Hybrid Clouds Harnessing Renewable Energy and Task Scheduling

Jie Zhao, Maria A. Rodríguez and Rajkumar Buyya
*Cloud Computing and Distributed Systems Laboratory*
*School of Computing and Information Systems*
The University of Melbourne, Australia
Email: zhao.j4@student.unimelb.edu.au, {maria.read, rbuyya}@unimelb.edu.au

***Abstract*—The use of cloud computing for delivering application services over the Internet has gained rapid traction. Since the beginning of the COVID-19 global pandemic, the work from home scheme and increased business presence online have created more demand for computing resources. Many enterprises and organizations are expanding their private data centres and utilizing hybrid or multi-cloud environments for their IT infrastructure. Because of the ever-increasing demand for computing resources, energy consumption and carbon emission have become a pressing issue. Renewable energy sources have been recognized as clean and sustainable alternatives to fossil-fuel based brown energy. However, due to the intermittent nature of availability of renewable energy sources, it brings many challenges to automatically and efficiently schedule tasks under renewable energy constraints and deadlines. Task scheduling with traditional heuristic algorithms are not able to adapt quickly with changing energy availability and stochastic task arrival. In this regard, this work aims at building a novel scheduling policy with deep reinforcement learning, which automatically applies scheduling techniques like workload shifting and cloud -bursting in a geographically distributed hybrid multi-cloud environment consists of multiple private and public clouds. Our primary goals are maximizing renewable energy utilization and avoiding deadline constraint violations. We also introduce user configurable hyper-parameters to enable multi-objective scheduling on cloud cost, makespan and utilization. Our experiment results show that the proposed scheduling approach can achieve the aforementioned objectives dynamically to varying renewable energy availability.**

## I. Introduction

Cloud computing was predicted to be the 5th utility in 2009, and now the vision has been realized [1]. During the past decade, cloud computing has seen massive adoption and rapid expansion [2]. This trend has been growing even faster after the COVID-19 global pandemic took hold, thanks to the massive work-from-home scheme and increasing online business presences. Many enterprises and organizations are migrating workloads to public clouds while still utilizing their existing on-premise private clouds, which is known as hybrid clouds. According to the Flexera 2021 State of the Cloud Report Survey [3], 99% of 750 participating enterprises have adopted at least one public or private cloud; and 92% are using multi-cloud and hybrid cloud.

With this ever-increasing demand, energy consumption and carbon emission of Cloud Data Centres (CDC) emerged as a pressing issue for the environment. According to International Energy Agency (IEA), global data centre electricity consumption was around 200 Tera-Watt hours (TWh) in 2019 and is projected to grow to 270 TWh in 2022 [4]. If all CDCs are combined as a virtual country, this projected figure would be ranked at No. 16 and more than the national energy consumption of Australia [5]. In addition, CDCs are hugely contributing to the global emission, which has a larger impact of environment. Hence, policymakers around the world are setting targets to reduce energy consumption and carbon emission. Many countries, such as the UK, France, Denmark and New Zealand, have set target and legislation to reach Net-Zero emission target by 2050; this means simultaneously reducing and removing greenhouse gas emission to reach a zero-sum total carbon emissions [6]. In this regard, the IT industry and computing research community must play its role to help to achieve this goal, therefore, sustainable computing has become an important research trend in recent years. [7]

One essential way to realize the goal of sustainable cloud is to utilize clean renewable energy sources like solar, wind or hydroelectric energy. Naturally, renewable energy availability is intermittent due to constantly changing weather conditions, e.g. in the case of solar energy, production is significantly lower on a cloudy day or nil at night-time. Thanks to recent technological advancements and cost reduction in battery storage like LG Solar [8], Tesla Powerwall [9], Samsung SDI [10], etc., batteries can help to stabilize renewable energy supply fluctuation. Also, since the cost of such a system continues to decrease and help from government initiatives is available across the globe, small to medium businesses are increasingly interested in powering their on-premise computing infrastructure with solar energy and batteries. This brings a unique opportunity for utilizing renewable energy for two reasons: 1) These private clouds are usually small to medium scale and can be powered entirely with renewable energy; 2) With regard to workload, it is quite common in both industry and academia to see computational tasks with flexible start time but strict deadlines(e.g. accounting department is doing

last month's reconciliation but only need the result by 15th of this month.)

A recent study by Xu et al. [11] shows that an optimal matching of energy demand with supply provides the best ratio of renewable energy utilization. However, optimising renewable energy usage while orchestrating the workload to satisfy the various quality of service (QoS) metrics remains an open research challenge. Task scheduling is an NP-Hard combinatorial optimization problem, and adding requirements like satisfying QoS and deadline while under renewable energy constraints makes it even more challenging.

To create scheduling policies that are autonomously adaptive to the environment, recently, Reinforcement Learning (RL) has caught a lot of attention in the research community. By combining deep neural network (DNN) as a function approximator, deep reinforcement learning (DRL) has been proved to work well in the previously infeasible RL problem [12]. Moreover, many works have recently explored applying DRL for task scheduling [13]–[15], which has shown feasibility of such solutions in real cloud environments. In this regard, we set out to employ the DRL technique to tackle the aforementioned scheduling problem and challenges with a combination of one or more existing techniques mentioned above in an autonomous and self-adaptive way. We propose a model-free approach based on proximal policy optimization (PPO) [16] to maximize renewable energy while still satisfying hard deadline constraints. We use stochastic workload size and arrival time. To evaluate our policy, we compare it to classical heuristic, and deep Q-learning network(DQN) based policies.

The main contributions of the paper are as follows:

- We propose a model-free PPO-based task scheduling algorithm in a geographically distributed heterogeneous hybrid cloud environment to harness renewable energy while still satisfying deadline constraints.
- We propose DRL approach to match demand with renewable energy supply through automatic workload shifting, and use cloud-bursting when resource is insufficient.
- We study a combination of neural network architecture and DRL algorithms in the mentioned scheduling and decision-making problem and compared their performance in terms of convergence and various metrics;
- We experiment with real-world solar energy data and workload with stochastic characteristics in a controlled simulation environment.

The remainder of the paper is structured as follows. In Section III-B, we present an overview of the studied problem and describe the proposed system architecture. After that, related studies are summarized in Section II. Section III presents the problem formulation of the scheduling and control problem in this paper. It is then followed by experimental setup, evaluation results, observations and discussions in Section V. Section VI concludes the work and discusses future research directions.

## II. Related Work

In this section, we explain the related works relevant to the background of our research problem. Figure 1 illustrates solar

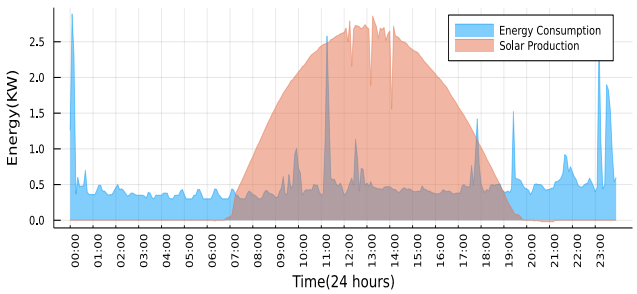


Fig. 1. An Example of Solar Energy Utilization Inefficiency

generation and consumption pattern during a 24-hour period. The centre area is electricity generated by a solar system, and the other area is power consumption by a small scale private cloud. This example shows a utilization inefficiency when renewable energy is at its peak, and the system is still consuming brown energy when renewable energy is not available. Assuming task start time are flexible, an optimal solution is to shift tasks into time slots when more renewable energy is available.

The research community has a long history of proposing new approaches for building green data centres and energy-aware scheduling algorithms to solve this problem (See Table I. In their 2009 paper, Steward and Shen [28] briefly emphasized the importance of coordinated research of resource management and renewable energy utilization. Goiri et al. proposed GreenSlot in 2011 and implemented a renewable energy-aware scheduler as an extension of SLURM. [29] They further evolved their idea and developed a small scale micro green data centre named Parasol [17], which utilized solar energy to process MapReduce jobs. However, their studies did not consider geographically distributed data centres. By shifting the workload to geographically distributed data centres, renewable energy utilization can be maximized while still guaranteeing acceptable QoS levels. More recent works employed various techniques like VM consolidation [**?**], [19], online VM migration and dynamic placement [20], [30], fuzzy logic [22], optimal workload distribution [31], distributed DVFS [23], Brownout [24], Powercapping [32] etc. Xu et. al [11] formulated the scheduling problem as a Markov Decision Process(MDP) and solved it with value iteration. Most of the previous works are either use heuristics or mathematical analysis to produce scheduling policies, however, heuristics are generally hand-crafted by experts and does not adapt well in environments such as Cloud workloads and renewable energy availability.

Recently, using DRL as a resource management and task scheduling technique was sparked by Mao et al. in their paper titled DeepRM [13], they formulated task packing problem into a learning problem and used the policy gradient method to achieve good performance. Since the climate change issue gained many traction lately, many works have proposed DRL based techniques to utilize renewable energies or improve energy efficiency, e.g, DRL-Cloud [25], RLScheduler [33],

TABLE I
COMPARISON OF RELATED WORK

| Work | Approach | Geo Distributed | Renewable Energy | Cloud Cost | Deadline | Techniques | Workload Type | Decision Level |
|---|---|---|---|---|---|---|---|---|
| [17] | MILP with Gurobi Solver | ✗ | ✓ | ✗ | ✗ | Energy Prediction, Solver | MapReduce, Mixed | Single |
| [18] | Greedy Algorithm | ✗ | ✓ | ✗ | ✓ | Energy Prediction, Greedy Algorithm | Scientific Workflow | Single |
| [19] | Heuristic | ✗ | ✗ | ✗ | ✓ | VM Consolidation | PlanetLab | Single |
| [20], [21] | Heuristic | ✓ | ✓ | ✗ | ✓ | VM Migration and Dynamic Placement | Lublin-Feitelson | Single |
| [22] | Fuzzy Logic | ✓ | ✓ | ✗ | ✗ | Load Balancing | Google Trace | Single |
| [23] | DRL(Q-Learning) | ✓ | ✓ | ✗ | ✗ | Distributed DVFS | Google Trace | Single |
| [24] | Approximate MDP | ✗ | ✓ | ✗ | ✗ | Brownout | PlanetLab | Single |
| [11] | Heuristic | ✗ | ✓ | ✗ | ✓ | VM Consolidation, Scaling, Brownout, Shift | Mixed | Multiple |
| [25] | DRL(DQN) | ✗ | ✗ | ✗ | ✓ | Energy Efficient Scheduling | Google Trace | Multiple |
| [26] | DRL(DDQN) | ✗ | ✗ | ✗ | ✓ | Energy Efficient Scheduling | MiBench | Single |
| [27] | DRL(A3C) | ✓ | ✗ | ✓ | ✓ | Energy Efficient Scheduling | Bitbrain | Single |
| This Work | DRL(PPO) | ✓ | ✓ | ✓ | ✓ | DVFS, Workload Shift, Cloud Bursting | Stochastic | Single |

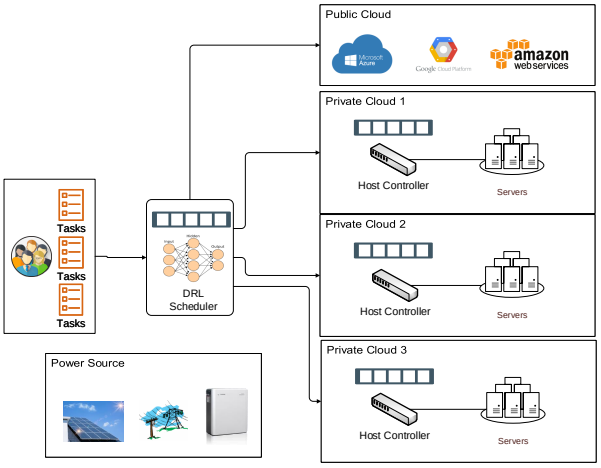


Fig. 2. System Architecture

DQL-EES [26], A3C-R2N2 [27]. These aforementioned works all try to minimize SLA violation instead of considering it as a hard constraint. However, many workloads in both industry and academia have a flexible start time but strict deadline. Our trained policy automatically uses cloudbursting at the right time to guarantee deadline constraints are met in case that renewable energy or computational resources are insufficient in local data centres.

## III. System Architecture and Model

### A. System Architecture and Challenges

In this section, we present the proposed system architecture and then describe problem formulation. Figure 2 depicts the system architecture considered in our study. We consider a hybrid cloud model comprised of $m$ geographically distributed private data centres (DC) $DC^L = \{DC_1^L, DC_2^L, \ldots, DC_m^L\}$ and a public cloud denoted as $DC_P$. Each DC consists of $n$ DVFS-enabled hosts $H_{m,n} = \{H_{11}, H_{12}, \ldots, Hmn\}$ and equipped with renewable energy source with backup source such as battery or grid. In private DCs, heterogeneous servers are modelled, where some servers have lower CPU frequency but higher energy efficiency, while others are much faster but resource-hungry and anywhere in between. We use container orchestration platforms provided by CSPs such AWS Elastic Container Services (ECS) or Microsoft Azure Containers in the public cloud. These allow a user to dynamically provision and scale down to zero with no associated cost.

The goal of our work is to automatically shift workload to maximize renewable energy utilization and control public cloud cost while still meeting strict QoS requirement (deadline). We propose reaching this objective by intelligently scheduling tasks to different servers based on energy availability and deadline constraints. To achieve the QoS objective while there is no adequate renewable energy or computing capacity, we propose using a public cloud service provider in a dynamically on-demand provisioned manner. Usually, performance, energy and cloud cost are conflicting objectives, and there is always a trade-off to be made in a multi-objective optimization problem. We use configurable hyper-parameters to allow configuration of preference, which will be discussed further in Section IV.

To achieve an optimal scheduling policy, a few challenges need to be addressed:

1) *Workload Shifting*: The system needs to decide when to start tasks based on their resource requirement, deadline constraint and available renewable energy so that they are executed efficiently without exceedingly using brown energy.
2) *Task Assignment*: In a heterogeneous environment, for example, ARM64 based servers run slowly but are more energy efficient (performance per watt) than their X86 counterpart; multi-core performing CPU consume more energy but completes tasks much faster, which server to place a task is also a hard problem due to deadline constraints;
3) *Cloud Provisioning*: In case of insufficient local resource availability, the right timing to provision more resources in a public cloud is also a non-trivial task. Provisioning of public cloud resources too early may incur unnecessary cost because local resources may become available before the deadline; in contrast, provisioning too late may cause deadline violation.
4) *Cloud Scaling*: Related to the challenge above, what type of computing resource to provision is also challenging. Higher performance VM cost more but may result in a resource wastage, while lower ones may be too slow to complete the assigned tasks on time.
5) *Stochastic Environment*: Aforementioned challenges be-

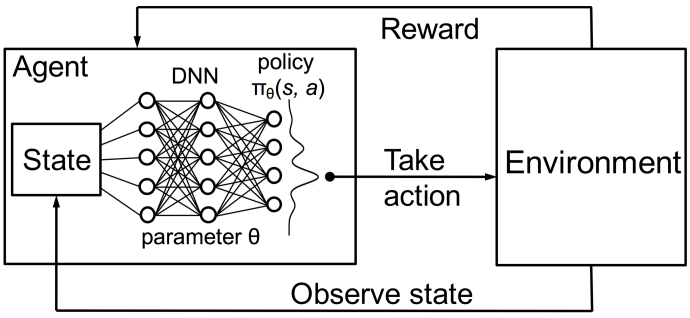


Fig. 3. Concept Diagram of DRL

come even more difficult when considering an environment with stochastic job arrival. For example, the scheduler decides to delay the current task based on the current situation, but another demanding task arrives; QoS violation is inevitable in this situation; however, DRL can be used to minimize regret similar to a multi-armed bandit problem.

### B. Background of DRL

Due to the aforementioned challenges, explicitly programming the scheduler is a non-trivial task. Hence, we sought a DRL-based approach to automatically learn an adaptive optimal policy for scheduling problems. DRL agents can sample from an environment or a simulator therefore the agents need only to be guided by a reward function.

The basic conceptual diagram of DRL is shown on Figure 3. In a typical DRL system, the main components are the environment and the agent. [34] The agent has its own view of the environment called observations or states. Depending on the current state, the agent takes action steps based on its DNN output and receives a reward from the environment. The learning process is for the agent to take a series of actions to maximize its cumulative reward over a period of time and update its DNN to make better decisions.

In our system, the agent is the DRL scheduler making scheduling and control decisions, and the environment includes the private and public cloud data centre, renewable energy generator, and tasks to be scheduled. The DRL scheduler observes the system state and takes series of steps to achieve the design objectives. It receives tasks from users and decides either to shift the workload by holding it in the scheduler's queue, sending it to a server in a private DC, or providing a resource in a public cloud. After taking action, the agent receives a reward from the environment in the training process and tries to maximize the reward it receives in a certain time period. At each DC level, the servers can be controlled by rule-based host controllers, sending the server to sleep mode or power off idle servers. On task arrival, the host controller sends commands to wake up or power up the destination server through IPMI or PDM commands.

TABLE II
MODEL PARAMETERS.

| Symbol | Definition |
|---|---|
| $DC_m^L, DC_m^P$ | $m$th private/local (L) or public (P) Data Centre |
| $P_m^{DC}$ | Total Power Consumption of $m$th Data Centre |
| $P_{m,i}$ | Power Consumption of $i$th host in $m$th Data Centre |
| $H_{m,i}$ | $i$th host in $m$th Data Centre |
| $C$ | Per second cost of energy and cloud $C^{green}, C^{brown}, C^{cloud}$ |
| $RE_m^{DC}$ | Total Available Renewable Energy at $m$th Data Centre |
| **Task Parameters** | |
| $\lambda_n$ | Arrival rate of task $n$ |
| $S_n$ | Task Size $n$ |
| $R_n$ | Resource Requirements for Task $n$, $R_n^{CPU}$, $R_n^{Mem}$, $R_n^{Storage}$ |
| $CT_n$ | Completion Time for Task $n$ |
| $D_n$ | Deadline constraint of task $n$ |
| **Hyper Parameters** | |
| $TS$ | Scheduling Time Slot |

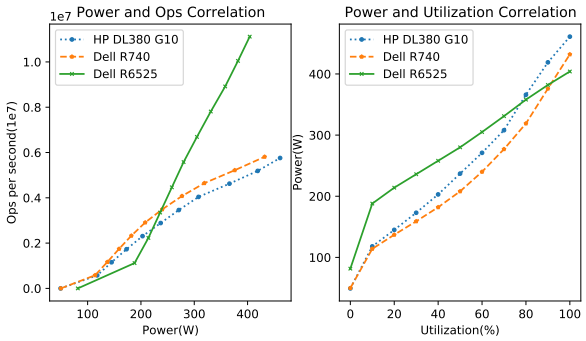


Fig. 4. Utilization, Power and Ops Correlation

### C. System Model

In this Section, we describe the system model. The notations used in this paper can be found in Table II.

*a) Assumptions:* We assume that once tasks are received by servers in a private CDC, it will complete them. We do not consider task migration or task failure mitigation in this study. We assume each private data centre has battery or grid to power through the nighttime when all servers are idle, in sleep mode or powered off.

*b) Energy Consumption Model:* We adopt the host power model from real-world server measurement results published according to SPECPower and choose three recent models with CPU from Intel and AMD [35]. Many studies show that although physical server power consumption is determined by various components, they have a linear correlation with CPU load. [19], [21]. However, as shown on Figure 4 based on real measurement, we argue that the lower 10% utilization is non-linear. For example, at idle, HP DL380 consumes 49.3W but the figure jumps more than double to 118W at 10%. Hence, to improve accuracy, we assume a linear relationship within each 10% interval.

$$P_i^{server} = \begin{cases} P_i^{idle} + (P_i^{0.1} - P_i^{idle}) \times u_i & u_i \leq 0.1 \\ P_i^{0.1} + (P_i^{max} - P_i^{0.1}) \times u_i & 0.1 < u_i \leq 1 \\ P_i^{off} & otherwise \end{cases} \quad (1)$$

where $P_i$ denotes the power consumption of host $i$ at CPU utilization $u_i$ in the data centre; $u_i \in [0, 1]$ represents the CPU utilization of host $i$. We adopt the server size Java operation per second (SSJOPS) metric from SPECPower for our task and server model for computing the server load. Since the numbers are quite large for the latest servers, we used MSSJOPS in our simulator, where 1 MSSJOPS = $10^6 \times SSJOPS$.

*c) Non-IT Component Power Model:* In addition to server power consumption, other non-IT components like cooling systems in data centres also consume significant energy. To calculate non-IT power consumption, we use a simple analytical model based on Power Usage Efficiency (PUE) in our simulator. PUE is defined as

$$PUE = \frac{P^{total}}{\sum_{i=1}^{n} P_i} \tag{2}$$

where $P^{total}$ denotes the total power consumption of the whole DC and power consumed by all servers. In our simulator, we set this variable to a constant accordingly to the latest PUE figure published by Google: $PUE = 1.11$. [36] Therefore, total energy consumption of $m$th data centre can be calculated as $P_m^{dc} = \sum_{k=1}^{i} PUE \times P_k$.

*d) Workload Model:* We consider an immutable container-based computational intensive workload. The container workload may have simple tasks or complex workflows inside its execution logic; however, the containerized workload can be modelled after a simple demand of computing resources from a modelling perspective. Each host can run many containers at a certain time period up to the limit of available CPU cores on each host. We assume required resources are known and specified by users or the users provide an estimate when submitting the tasks. For each task, it has arrival time $\lambda_n$, size $S_n$, resource requirements $\{R_n^{CPU}, R_n^{Mem}, R_n^{Storage}\}$ and a deadline $D_n$.

*e) Cost Model:* We represent per-second cost as $C^{green}, C^{brown}, C_t^{Cloud}$ for green renewable energy, brown energy and cloud monetary resource cost. The cost of executing a task was calculated in the following two conditions:

1) Private: In case the scheduler sends a task to a server in a private DC, and if that DC's energy consumption is below its available renewable energy, we consider $C^{green}$ as 0, since the renewable energy infrastructure has already been invested and renewable energy is available freely.
2) Public: If we need to send the task to a public cloud, the total cost is computed based on provisioned cloud VM type. Brown energy price is computed for the agent to make a cost-effective decision on whether to send the task to the public cloud or execute it locally using brown energy.

*f) Optimization Objectives:* The major objectives of this work are to maximize total renewable energy utilization; in other words, control DC power consumption to minimize renewable energy wastage across data centres; at the same time, minimize public cloud cost when renewable energy is not enough.

$$Minimize: \sum_{i=1}^{m} \int (RE_{m(t)}^{DC} - P_{m(t)}^{DC}) d(t) \tag{3}$$

$$Minimize: \sum_{k=1}^{n} C_k^{Cloud} \tag{4}$$

$$Subjectto: CT_n <= D_n \tag{5}$$

In addition, we introduce a few other metrics like makespan and server utilization as hyper-parameters, further detail will be explained in reward metrics in Section IV.

## IV. DRL Scheduler Design and Implementation

In this section, we present the design and implementation of our proposed DRL-based task scheduler. Due to the aforementioned challenges, we use a model-free approach. The DRL agent learns to perform optimally with the guidance of a reward function and state observations. This model-free approach differentiates itself from heuristic and meta-heuristic based methods. DRL technique does not require any prior knowledge, while (meta)heuristics need expert knowledge to write program logic to achieve optimization objectives explicitly. Firstly, we describe our renewable energy and deadline constraint problem formulation as a series of decision problems; afterwards, we detail the design and implementation of our algorithm.

*Environment:* We implemented our scheduler using Python 3.8, OpenAI Gym [37] and PyTorch [38]. In addition, we utilized Stable Baseline 3 (SB3) [39], a commonly used, open-source and well-tested DRL library, as our training framework and base algorithms implementation. We adopted the Proximal Policy Optimization (PPO) [16] algorithm in this work.

*Observation Space:* We used a continuous state space in our design, represented by a large vector of float numbers normalized to a $[-1, 1]$ space. This helps to stabilize the training process and makes it a lot easier to debug. The elements of the vector are described in sequence as below.

- Time of Day: As we harness renewable energy like solar, time of day is an important metric. We normalized it into a continuous space $[-1, 1]$ where $-1$ means 00:01 at the current day and 1 means 00:00 at the next day. The normalization is not a necessity but rather a design choice. We found it is easier to compute various metrics if we use a symmetrical normalization, since the highest production of solar energy is usually around noon(value 0).
- Renewable Energy: A vector of predicted available renewable energy in the next time slot $T_d$; the values contained in the vector are normalized with the full solar system capacity as a parameter.
- DC Power Consumption: A vector of power consumption of each DC in the scheduling time slot; the values are also normalized with the full system capacity.

- Incoming Job: A vector of incoming job attributes, also normalized by a set of parameters of allowed maximums.
- Server Load: Load of all servers in private data centre combined into a single large vector, computed based on every server's currently executing tasks and all tasks in the queue.
- Public Cloud Cost: Cost of public cloud in the last time slot $T_d$; cloud cost observations are normalized by a parameter of configurable daily maximum;

Server specification such as power consumption and speed at idle and maximum load is static variables, and dynamic power consumption can be calculated with server load. We consider these static metrics as common knowledge between the scheduling agent and the environment. It helps to reduce observation space.

*Action Space:* In our model, we define a discrete action space with $n$ actions described below. The scheduler makes a decision on job arrival, or when a job in its queue reaches start time, then it takes the following actions:

1) Action 0: The scheduler does nothing; it pushes the current incoming job by $T_d$ seconds back into its processing queue and wait for already scheduled jobs to be processed;
2) Action $[1, m * i] \in N$: Flattened index of $i$th server in $m$th private data centres, the scheduler sends task to $i$th server and remove it from queue;
3) Action $[m * i + 1, n]$: The scheduler sends the task to a type of public cloud VM decided by the action.

*Reward Function:* We define and normalize our reward function as a float number in a $[-1, 1]$ space since we find it works well in our experiments. Since the agent is guided by a single real value, reward shaping is necessary for the algorithm to know how good or bad an action is. The agent receives a positive reward when it performs well, e.g. having energy consumption under a renewable budget, scheduling to the right server at the right time, public cloud spending is under budget, etc. On the contrary, it receives a negative reward when it makes a bad decision as a penalty. The reward function design is inspired by Tuli et al. [27]). In their work, the authors defined several loss functions normalized to $[0, 1]$ space and as a convex combination and used a few parameters to enable customization of scheduling goal. We adopted a similar strategy but normalized to a symmetric space with several objective weight sums to $[-1, 1]$ space. Since we use both positive reward and punishment, this design choice allows easier distinguish between reward and penalty and stabilizes the training process. The reward metrics we use in this study are listed as below:

1) Renewable Energy Utilization (REU) is defined as the difference between power consumption and available renewable energy. If the agent uses more energy than available renewable energy, the environment gives a penalty, or otherwise 0. A positive reward is not given since completing tasks is also given rewards; it is to avoid double-rewarding.
2) Cloud Cost Reward (CCR) is defined as the normalized value of cloud cost in the last scheduling time step in relation to a user-defined maximum daily budget. When the agent spends more on the public cloud, it receives a larger penalty.
3) Deadline Violation (DV) is defined as a proportional task that violated deadline constraint in relation to total completed jobs. Since we consider deadline as a hard constraint, we set the weight of this metric to a higher value than all other rewards.
4) Makespan Reward (MR) is defined as total time used to complete tasks in a trajectory. The weight of this reward is set to a rather small value since our main objective is REU, CCR and DV. During our experiments, we found if a longer makespan is not penalized, at some stage, our agent shows a procrastination syndrome which means delay everything until the last minute.
5) Utilization Reward (UR) is defined as a balance of utilization of servers, and we penalize imbalanced task assignment if some servers are getting too many tasks. This penalty is also introduced for a similar reason as MR, if the penalty is not present and the deadline allows, the agent tends to send many tasks to the most efficient server for the highest reward while the others are sitting idle.
6) Task Reward(TR) is defined as the reward an agent received for completed a task. A higher reward is given for completing a harder task. Since a harder task naturally consume more energy or incur a higher cloud cost, the agent needs to find the right balance between this reward and other penalties, resulting in a better policy. Also, with the discount factor $\gamma$, the agent will complete the same task earlier to get a higher reward.

To allow user configurable settings for a multi-objective optimization, we introduced hyper-parameters similar to [27]. Thus,

$$\begin{aligned} TotalReward = a \times REU + b \times CCR + c \times DV \\ + d \times MR + e \times UR + f \times TR \\ a, b, c, d, e, f \geq 0 \\ a + b + c + d + e + f = 1 \end{aligned} \tag{6}$$

Because each individual reward are normalized, the total reward an agent can receive in each step is still bounded in a symmetrical $[-1, 1]$ space.

## V. Model Training And Performance Evaluation

In this section, we describe our experiment settings, training procedure and evaluation.

### A. Dataset

For deep reinforcement learning-based research, it is common to run the algorithm training in a simulated environment, because training in a real setup is too slow and often infeasible for large complex production environments [33]. Our experiment uses the Pareto distribution to generate job

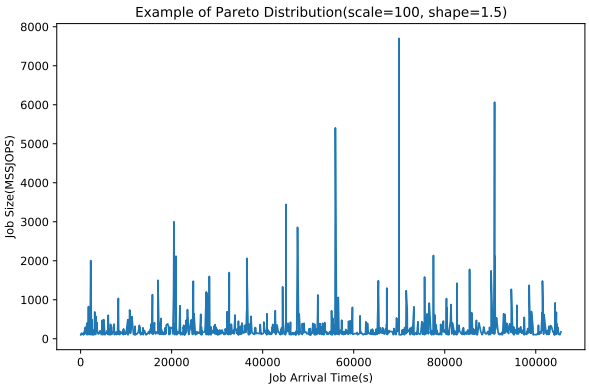


Fig. 5. Pareto Job Size with Poison Arrival

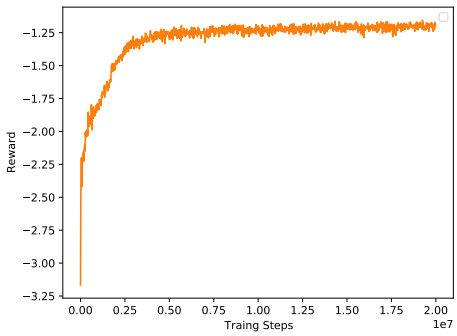


Fig. 6. Reward During Training

size and the Poisson distribution to produce job arrival time. It allows us to study our proposed policy's performance under different loads while mimicking real-world workload characteristics such as stochastic but steady arrival rate, with some occasional bursts. An illustration of generated tasks with $scale = 100, shape = 1.5$ is shown in Figure 5. We trained our policy with a few sets of generated tasks and validated using a different seed.

### B. Environment Setup

The infrastructure considered in our study is shown in table III, we consider 3 DCs equipped with 2 servers of each type in each DC. For public cloud, we use 4 types of cloud instances in our model, m5.$\{x, xl, 2xl, 4xl\}$large which costs \$0.12/h, \$0.24/h, \$0.48/h, \$0.96/h, respectively. The M5 type of general-purpose instance is equipped with Intel Xeon Platinum 8175M. Since this CPU is specifically made for AWS by Intel and provides no publicly available data, we assume its performance is slightly slower than the Intel Xeon 8180 in the same generation. Hence, we use the 8180 measurement data minus a 5% margin.

### C. Training Process

We use Spartan HPC [40] equipped with Xeon E5-2650 v4(2.2 GHz) and multiple NVidia T100 GPUs for hyper-parameters and neural network architecture search. First, we use the HPC cluster to try a few combinations of hyperparameters and neural network architectures and run the training process for a limited epoch/time steps to see its metrics and convergence behaviour. If the combination's performance looks promising, we further conduct a longer training on a workstation machine with AMD Ryzen 5900X(3.7GHz boost to 4.8 GHz), NVidia GTX 3070 (8G) graphic card and 64G RAM. The reason behind this choice is that we found simulation of the DRL environment is CPU bound; using a CPU with a higher base clock can significantly accelerate the training process (about 2.5x in our case). Also, the new TensorCore in NVidia's Ampere architecture [41] improves training speed significantly.

For our model training, the hyper-parameter that works well are as follow: $learning\_rate = 1e-4, batch\_size = 64, layers = [256, 128, 64, 32], optimizer = Adam, \gamma = 0.95, n_{step} = 1024$. Other hyper-parameters are using default implementation in SB3 [39].

The baseline algorithms used in the comparison include Round Robin (RR), Maximum Renewable Least Utilization First (MR-LUF) and DQN. The advantages of heuristic algorithms are their simplicity to understand and implement. For example, RR chooses hosts in order regardless of their utilization. In MR-LUF, we first find the data centre with the maximum difference between renewable energy and schedule the task to the server with the least load in that DC. We also compared with a DQN based DRL algorithm which is commonly used in many studies.

Figure 6 shows the reward function during training. The agent receives maximum reward at around $3 \times 10^6$ steps and doesn't improve much further with more time for training. We are interested to see if the policy can be improved further, hence, we continue the training until $2 \times 10^7$. The figure clearly indicates the policy has converged and further training didn't improve the policy.

### D. Results and Discussion

To evaluate the trained agent, we run an experiment with 2000 jobs based on the probability distribution described in Section IV but with a different seed, hence, a new sequence of workload size and arrival time is generated. Then, we compare the performance with baseline algorithms in terms of energy consumption and cloud cost in a 24-hour period.

Figure 7 shows the cumulative summary of energy consumption for the experiment. Some jobs are pushed to the next day by the DRL agents, and 1764 jobs are completed at the end of the day. These delayed jobs are taken out of the result for a fair comparison, and their energy consumption records are also removed. RR, MRLUF, DQN and Proposed algorithm consume $9307W, 10409W, 7430W, 6806W$ of energy, respectively. Hence, the proposed DRL policy reduce energy consumption by 26.87%, 34.61%, 8.40% compared to RR, MRLUF, and DQN, respectively. The result clearly indicates both DRL based agents can help save energy by

TABLE III
CONFIGURATION OF HOSTS IN THE EXPERIMENT SET UP

| Name | Processor | Cores | RAM | SPEC Power(W)/SSJOPS for different utilization | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 0% | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
| Dell PowerEdge R6525 | AMD EPYC 7702 2.0GHz | 64C/128T | 128GB | 81.6 | 188 | 214 | 236 | 258 | 280 | 305 | 331 | 358 | 382 | 404 |
| | | | | 0 | 1117236 | 2228757 | 3348739 | 4463523 | 5577282 | 6688097 | 7809749 | 8917380 | 10044691 | 11115782 |
| Dell R740 | Intel Xeon Platinum 8280 2.7GHz | 28C/56T | 188GB | 49.8 | 114 | 137 | 159 | 182 | 208 | 240 | 277 | 319 | 376 | 432 |
| | | | | 0 | 580574 | 1165221 | 1745535 | 2325614 | 2909373 | 3491467 | 4077942 | 4648443 | 5217814 | 5811114 |
| HP DL380 Gen10 | Intel Xeon Platinum 8180 2.50GHz | 28C/56T | 96GB | 49.3 | 118 | 145 | 173 | 203 | 237 | 271 | 308 | 366 | 419 | 461 |
| | | | | 0 | 575883 | 1154607 | 1730201 | 2307605 | 2885565 | 3459701 | 4039806 | 4624500 | 5187730 | 5758036 |

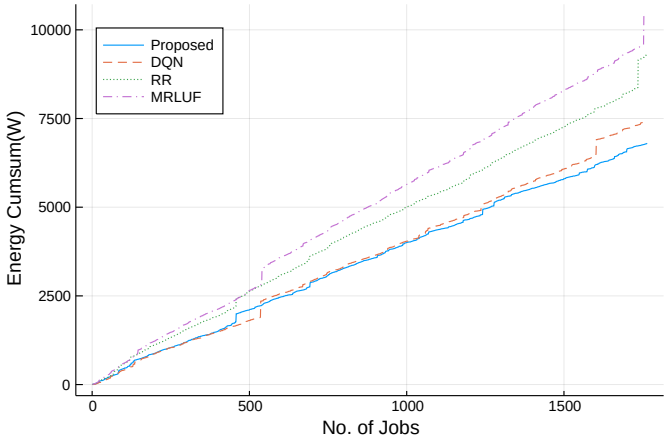


Fig. 7. Cumulative Summary of Job Energy

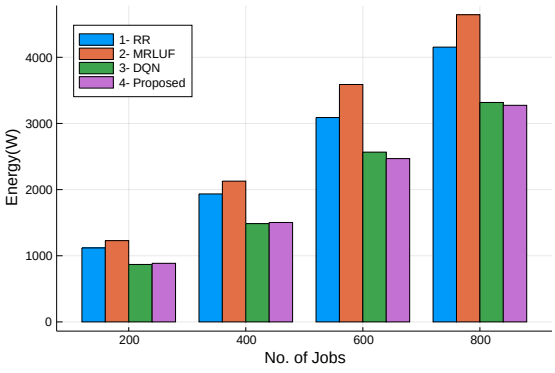


Fig. 8. Energy Consumption after Different Number Job Completion

shifting tasks to when more renewable energy is available. Figure 8 shows cumulative energy after n job completion. The trend continues after 800 jobs; hence, the diagram is truncated for clarity and conserving space. This diagram exhibits the same energy-saving as Figure 8, when more and more jobs come, the proposed DRL based workload shifting technique will save more energy.

Figure 9 shows the public cloud cost of the experiment. Note that we are not using cloudbursting in our baseline heuristic algorithms because we focus on a model-free approach in this work. Therefore, RR and MRLUF are not showing in the diagram. We found DQN and the proposed algorithm perform similarly in terms of overall performance because they are both trained to convergence. PPO based agent saves a bit more energy while sending more tasks to the public cloud.

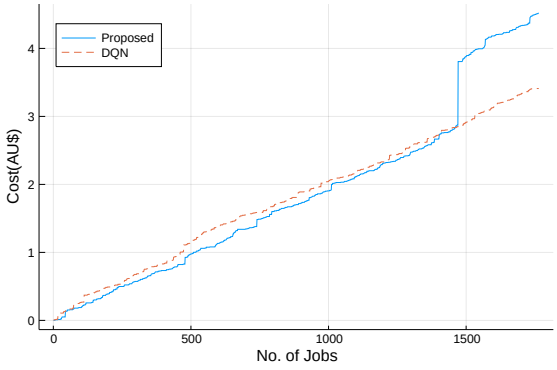


Fig. 9. Cumulative Summary of Job Cost

Also, it converges much faster than DQN in training therefore saves energy. We consider energy a primary concern in this work as long as the cloud costs are controlled under a certain budget. Hence, we consider the proposed PPO based algorithm superior to the DQN based one.

In comparison to the baseline algorithms, our work shows DRL-based algorithms can learn a better scheduling policy in complex environments without a model. However, the approach still has its limitations. In practical application, if the infrastructure changes, we will require model retraining. For example, adding or removing a physical server or introducing newer public cloud tiers will change the action space. Furthermore, although we used stochastic workload and arrival time in our experiment, the workload still satisfies certain characteristics. In the real world, some unseen workload may occur from time to time which may cause the policy to perform poorly.

## VI. CONCLUSIONS AND FUTURE WORK

In this work, we proposed a PPO-based DRL approach to optimize utilization of renewable energy in a hybrid multi-cloud environment. The proposed approach comprises the complex heterogeneous multi-cloud scheduling problem under renewable and deadline constraint. Our experiments show DRL is a promising technique for controlling complex system and make sequential decisions in the cloud resource management field. Model-free DRL can self-learn a better scheduling policy without expert knowledge and explicit programming like traditional heuristics and meta-heuristic based approaches.

Despite this, carefully designing the reward function and developing simulators in DRL training still require much expertise in the problem domain. Hence, as part of future work,

we will explore the possibility of using machine learning to determine the reward function automatically. Although DRL-based approach can adapt workload to some extent, changing of observation and action space or neural network architecture requires model retraining. To apply and test it in real-world test-beds, this process need to be automated and the scalability of proposed approach need to be further explored.

Acknowledgement: We thank Shashikant Ilager for proof-reading and suggestions to improve the quality of this paper.

## REFERENCES


[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[2] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. S. Netto, A. N. Toosi, M. A. Rodriguez, I. M. Llorente, S. D. C. D. Vimercati, P. Samarati, D. Milojicic, C. Varela, R. Bahsoon, M. D. D. Assuncao, O. Rana, W. Zhou, H. Jin, W. Gentzsch, A. Y. Zomaya, and H. Shen, "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–38, 2019.

[3] 2021 STATE OF THE CLOUD REPORT. [Online]. Available: https://info.flexera.com/CM-REPORT-State-of-the-Cloud

[4] Data centres & networks - fuels & technologies. [Online]. Available: https://www.iea.org/fuels-and-technologies/data-centres-networks

[5] List of countries by electricity consumption. Page Version ID: 1009400219. [Online]. Available: https://en.wikipedia.org/w/index.php?title=List_of_countries_by_electricity_consumption&oldid=1009400219

[6] (2020) UK net zero target. [Online]. Available: https://www.instituteforgovernment.org.uk/explainers/net-zero-target

[7] S. S. Gill and R. Buyya, "A taxonomy and future directions for sustainable cloud computing: 360 degree view," *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–33, 2018.

[8] Solar batteries and solar battery storage | LG solar energy australia. [Online]. Available: https://www.lgenergy.com.au/products/battery

[9] Powerwall | tesla. [Online]. Available: https://www.tesla.com/powerwall

[10] Samsung SDI ESS(energy storage system) - index | samsung SDI. [Online]. Available: https://www.samsungsdi.com/ess/index.html

[11] M. Xu, A. N. Toosi, and R. Buyya, "A self-adaptive approach for managing applications and harnessing renewable energy for sustainable cloud computing," *IEEE Transactions on Sustainable Computing*, no. 01, pp. 1–1, Aug 2020.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks - HotNets '16*. ACM Press, 2016, pp. 50–56.

[14] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh, "Learning scheduling algorithms for data processing clusters," in *Proceedings of the ACM Special Interest Group on Data Communication*. ACM, 2019, pp. 270–288.

[15] H. Mao, S. B. Venkatakrishnan, M. Schwarzkopf, and M. Alizadeh, "Variance reduction for reinforcement learning in input-driven environments," *arXiv preprint arXiv:1807.02264*, 2018.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[17] Í. Goiri, W. Katsak, K. Le, T. D. Nguyen, and R. Bianchini, "Parasol and greenswitch: Managing datacenters powered by renewable energy," *ACM SIGPLAN Notices*, vol. 48, no. 4, pp. 51–64, 2013.

[18] I. Goiri, R. Beauchea, K. Le, T. D. Nguyen, M. E. Haque, J. Guitart, J. Torres, and R. Bianchini, "GreenSlot: Scheduling energy consumption in green datacenters," in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–11, ISSN: 2167-4337.

[19] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[20] A. Khosravi, S. K. Garg, and R. Buyya, "Energy and carbon-efficient placement of virtual machines in distributed cloud data centers," in *European Conference on Parallel Processing*. Springer, 2013, pp. 317–328.

[21] A. Khosravi, L. L. Andrew, and R. Buyya, "Dynamic vm placement method for minimizing energy and carbon cost in geographically distributed cloud data centers," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 183–196, 2017.

[22] A. N. Toosi and R. Buyya, "A fuzzy logic-based controller for cost and energy efficient load balancing in geo-distributed data centers," in *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2015, pp. 186–194.

[23] C. Xu, K. Wang, P. Li, R. Xia, S. Guo, and M. Guo, "Renewable energy-aware big data analytics in geo-distributed data centers with reinforcement learning," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 205–215, 2018.

[24] M. Xu and R. Buyya, "Energy efficient scheduling of application components via brownout and approximate markov decision process," in *International Conference on Service-Oriented Computing*. Springer, 2017, pp. 206–220.

[25] M. Cheng, J. Li, and S. Nazarian, "Drl-cloud: Deep reinforcement learning-based resource provisioning and task scheduling for cloud service providers," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2018, pp. 129–134.

[26] Q. Zhang, M. Lin, L. T. Yang, Z. Chen, and P. Li, "Energy-efficient scheduling for real-time systems based on deep q-learning model," *IEEE transactions on sustainable computing*, vol. 4, no. 1, pp. 132–141, 2017.

[27] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using a3c learning and residual recurrent neural networks," *IEEE Transactions on Mobile Computing*, 2020.

[28] C. Stewart and K. Shen, "Some joules are more precious than others: Managing renewable energy in the datacenter," in *Proceedings of the workshop on power aware computing and systems*. IEEE, 2009, pp. 15–19.

[29] Í. Goiri, K. Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "Greenslot: scheduling energy consumption in green datacenters," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–11.

[30] A. Khosravi, L. L. Andrew, and R. Buyya, "Dynamic vm placement method for minimizing energy and carbon cost in geographically distributed cloud data centers," *IEEE Transactions on Sustainable Computing*, vol. 2, no. 2, pp. 183–196, 2017.

[31] C. Qu, R. N. Calheiros, and R. Buyya, "Mitigating impact of short-term overload on multi-cloud web applications through geographical load balancing," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 12, p. e4126, 2017.

[32] Faqiang Sun, Huawei Li, Yinhe Han, Guihai Yan, and Jun Ma, "PowerCap: Leverage performance-equivalent resource configurations for power capping," in *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*, 2016, pp. 1–8.

[33] D. Zhang, D. Dai, Y. He, F. S. Bao, and B. Xie, "Rlscheduler: an automated hpc batch job scheduler using reinforcement learning," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020, pp. 1–15.

[34] Part 1: Key concepts in RL — spinning up documentation. [Online]. Available: https://spinningup.openai.com/en/latest/spinningup/rl_intro.html

[35] K.-D. Lange, M. G. Tricker, J. A. Arnold, H. Block, and S. Sharma, "SPECpower_ssj2008: driving server energy efficiency," in *Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '12. Association for Computing Machinery, 2012, pp. 253–254. [Online]. Available: https://doi.org/10.1145/2188286.2188329

[36] Efficiency – data centers – google. [Online]. Available: https://www.google.com/about/datacenters/efficiency/

[37] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI gym," 2016.

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[39] (2021) Stable baselines 3. [Online]. Available: https://github.com/DLR-RM/stable-baselines3

[40] L. Lafayette, G. Sauter, L. Vu, and B. Meade, "Spartan performance and flexibility: An hpc-cloud chimera," *OpenStack Summit, Barcelona*, vol. 27, 2016.

[41] NVIDIA ampere architecture: The heart of the modern data center. [Online]. Available: https://www.nvidia.com/en-au/data-center/ampere-architecture/