Towards a Holistic Controller: Reinforcement Learning for Data Center Control

Albin Heimerson
Lund University
Dept. of Automatic Control
albin.heimerson@control.lth.se

Rickard Brännvall[*]
RISE, Research Institutes of Sweden
Dept. of Computer Science
rickard.brannvall@ri.se

Johannes Sjölund[*]
RISE, Research Institutes of Sweden
Dept. of Computer Science
johannes.sjolund@ri.se

Johan Eker[†]
Ericsson Research
johan.eker@ericsson.com

Jonas Gustafsson
RISE, Research Institutes of Sweden
jonas.gustafsson@ri.se

ABSTRACT

The increased use of cloud and other large scale datacenter IT services and the associated power usage has put the spotlight on more energy-efficient datacenter management. In this paper, a simple model was developed to represent the heat rejection system and energy usage in a small DC setup. The model was then controlled by a reinforcement learning agent that handles both the load balancing of the IT workload, as well as cooling system setpoints. The main contribution is the holistic approach to datacenter control where both facility metrics, IT hardware metric and cloud service logs are used as inputs. The application of reinforcement learning in the proposed holistic setup is feasible and achieves results that outperform standard algorithms. The paper presents both the simplified DC model and the reinforcement learning agent in detail and discusses how this work can be extended towards a richer datacenter model.

CCS CONCEPTS

• **Computer systems organization → Sensors and actuators**; • **Hardware** → *Enterprise level and data centers power issues*; *Temperature control*; • **Computing methodologies → Reinforcement learning**; Modeling and simulation.

KEYWORDS

reinforcement learning, power optimization, load balancing, datacenter

ACM Reference Format:
Albin Heimerson, Rickard Brännvall, Johannes Sjölund, Johan Eker, and Jonas Gustafsson. 2021. Towards a Holistic Controller: Reinforcement Learning for Data Center Control. In *The Twelfth ACM International Conference on Future Energy Systems (e-Energy '21), June 28-July 2, 2021, Virtual Event, Italy.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3447555.3466581

[*]also at Luleå University of Technology
[†]also at Lund University

1 INTRODUCTION

The transition into cloud solutions and external IT-service has been ongoing for several years, as Microsoft Azure, AWS, Google Cloud and many other services have emerged during the last decade. New cloud and other IT services continue to pop up, putting increased demand on hosting platforms and datacenters. With the emergence of edge computing, supported by 5G-networks and new mobile services, an increase in the growth of locally (regionally) hosted services is expected.

In parallel with the expected growth in the IT domain, a societal push towards a green electrification of the society is ongoing, with e.g. electric cars becoming increasingly common. Green electrical production from solar and wind power plants is increasing and introducing new challenges to the electric grid, as the production is difficult to control, and in most cases cannot be stored. Urbanization creates even more intensified power concentration in cities, which further challenges the power grid capacities. All in all, power availability will be an increased challenge in many (most) places in the world. Cloud services and the underlying IT infrastructure needs to find ways to reduce, or at least limit, its energy and power demand in a smart way, to avoid expensive interruption of services, failures and prevention of future expected growth.

Inside a typical datacenter, there are hundreds to several thousand servers that can compute jobs related to the various services that the datacenter is hosting. In a modern virtualized environment like cloud service, IT workloads can in many cases be freely moved in between servers within, or sometimes even across, datacenters, allowing for power and energy optimization. The operating conditions of the datacenter, where the temperature is the most pronounced, and commonly discussed, also have a significant impact on the operational efficiency of the datacenter [9]. Finding the ideal temperature in the datacenter can potentially save a lot of power in the cooling process, especially if the cooling system is supported with free cooling capability. Having the ability to move IT-loads between servers inside the datacenter, to e.g. avoid hot spots, and dynamically operate the facility cooling system depending on the current conditions, allows for interesting control and optimization research.

This paper introduces the concept of applying reinforcement learning (RL) on a datacenter model, where the RL methods have

This work was supported by Vinnova grant ITEA3-17002 (AutoDC).

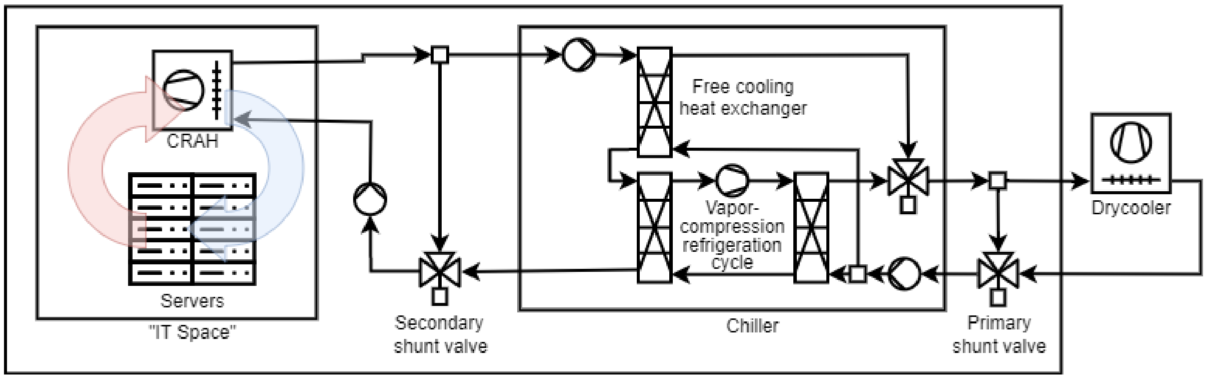

Figure 1: Conceptual heat rejection schematic of the physical model.

the ability to adjust both the cooling system and the distribution of IT workloads.

Outline. The outline of the paper is as follows. It starts with a problem formulation in Section 2. Section 3 presents a brief overview of related work on power optimization for datacenters as well as reinforcement learning applications for cloud services. Our simplified datacenter model is described in section 4. The control algorithm for managing the cooling equipment is presented in section 5 together with the results. Finally, future work is presented in section 6.

2 PROBLEM FORMULATION

In this work, simultaneous control is applied on multiple layers in a DC stack and a reinforcement learning (RL) agent is deployed that acts on all of them in a holistic manner. The assumption is that a controller acting over multiple layers will be able to find new and more efficient strategies than previous work, where the focus was often on controlling a single part of the system.

The results show that it is possible, and indeed useful, to deploy an RL agent that considers multiple layers in the stack. In this case, the layers included are the IT workload scheduling layer and the cooling layer. A simulated stack with simplified cooling and IT workload scheduler is used which optimizes energy usage while adhering to constraints used as standards in industry. It is shown to outperform standard algorithms in our simplified environment. The results hint that in a more advanced real-world setting, even though it will be more difficult for the RL agent to learn, it will also be able to find even more optimized strategies compared to the standard algorithms. This work is thus a stepping stone towards a more realistic and complete setup.

The datacenter. The model in this paper is based on an indirectly cooled datacenter, where Computer Room Air Handlers (CRAH) are transferring the heat generated in the IT-space by servers, into a water loop which in turn is connected to a chiller with free-cooling capability. The free-cooling functionality can be applied when the outdoor temperature is low enough to enable the dry-cooler to reject the heat from the IT without using the compression-cycle.In compressor mode, a vapor-compressor cycle lowers the temperature obtained from the dry-cooler to chill the water loop connected to the CRAH so the generated heat can be rejected. Operating the compressor in the chiller consumes a lot more power than operating the chiller in free cooling mode and also increases the risk of a failure, and should hence be avoided if possible. In Figure 1 the main components from a physical point of view are depicted.

The workload. The simulated IT workload is modeled to represent a simple 1-tier compute service, which could for example be an on-demand image recognition or inference task service. The service time as well as the load of jobs are assumed to be known in this case. This is a simplification, but previous work has been done to estimate similar values such as Ahmad et al. [1]. When the IT jobs arrive they are scheduled to execute at one of the servers. Job queues are not modeled, if a job is scheduled on a busy server then it cannot execute and is thus dropped. While this model is overly simplified, it is assumed to capture the essence of a more elaborate system where placing loads on full servers should be avoided.

3 RELATED WORK

Energy optimization over the different parts of a datacenter stack is nothing new and plenty of work has already gone into reducing the energy footprint of datacenters. Lucchese et al. [8] created non-linear models for the thermal and flow properties of servers and used model predictive control (MPC) to optimize over airflow cost while maintaining server component constraints.

Finding control strategies using data-driven methods is nothing new either, Lazic et al. [5] fitted a linear model to the DC dynamics and used MPC to improve the operational efficiency. Li et al. [6] used RL to minimize cooling energy by finding optimal facilities set-points.

Expanding beyond the hardware level, Townend et al. [14] presented a scheduler that takes multiple levels of infrastructure into account when distributing containers on a Kubernetes cluster while Baek et al. [2] use RL to balance workloads over heterogeneous servers while optimizing throughput and energy usage.

This work investigates the viability of using RL to find control policies that sense and act over multiple levels of infrastructure. This is not something the authors have encountered in any previous work, and they believe that this will allow the trained policy to become closer to optimal in a global scope. The important question then is if RL is a feasible method of finding these policies.

Table 1: States of the simplified model

Name	Description
$p^{\mathrm{SRV_i}}(t)$	Load on server i at time t.
$Q^{\mathrm{SRV_i}}(t)$	Volume flow in server i at time t.
$T_{\mathrm{in}}^{\mathrm{SRV_i}}(t)$	Air temperature into server i at time t.
$T_{\mathrm{out}}^{\mathrm{SRV_i}}(t)$	Air temperature out from server i at time t.
$Q^{\mathrm{CRAH}}(t)$	Volume flow in CRAH at time t.
$T_{\mathrm{in}}^{\mathrm{CRAH}}(t)$	Air temperature into CRAH at time t.
$T_{\mathrm{out}}^{\mathrm{CRAH}}(t)$	Air temperature out from CRAH at time t.
$T^{\mathrm{AMB}}(t)$	Ambient (outside) temperature

Thermal model for the datacenter. VanGilder et al. [15] proposed a compact model for the thermal dynamics in the datacenter with a chilled-water cooling system. It included discretized numerical models for heat sources (servers) and cooling by assuming a simple counterflow heat exchanger, which alternatively can be replaced by a quasi-steady-state model for applications when accuracy can be traded for model simplicity. This compact model is developed further in [4] that adds components for room, plenum, walls, floor and ceiling in order to better represent the complete thermal mass of a datacenter. The simplified model presented in the next section ignores the thermal mass and dynamics of equipment, facilities and cooling system, and only considers the heat and mass transfer associated with the cooling airflows.

4 THE SIMPLIFIED DC MODEL

In order to start testing the agent, a very simple model is used for a few reasons. Because the model is fast to simulate, fast RL model training and hyperparameter search can be achieved by running experiments in parallel. It is also easier to understand the RL agent's behavior in a simpler environment which can facilitate the construction of better and more efficient policies.

The simple model is designed to capture a few important aspects of a small datacenter, such as

- Heat generated by placing IT workload on servers.
- CRAH units that capture heat generated by the servers.
- Heat rejection using either free cooling or compressor-based cooling.
- Hot air from servers, which can flow back into server inlets (re-circulation) or into the CRAH units.
- Cold air from CRAH, which can flow back into the CRAH inlets (bypass) or over into the servers.

The re-circulation and bypass flows are common causes of inefficiencies in datacenter cooling, and decreasing their influence is an important target for optimization. In a modern datacenter, this recirculation and air mixture is often avoided using containment solutions where cold and hot air are separated using lightweight walls. However, this model assumes there is no containment.

With that in mind, N servers are modeled without any spatial distribution and cooled by a single CRAH unit, i.e. the air inlets for the servers will have a homogeneous temperature as will all the outlets. This is used as a first stepping stone to modeling more advanced flow.

Table 2: Constants used in the simplified model

Name	Description	Value
C_v	Volumetric heat capacity	1183 J/(Km^3)
R	CPU to air heat transfer	$2.54 \cdot 10^{-3}$ Km^3/J
T_i	Integral control constant	-12820 K/(m^3/s)
$p_{\mathrm{fan}}^{\mathrm{CRAH}}$	CRAH fan max power usage	1323 W
$Q_{\mathrm{min/max}}^{\mathrm{CRAH}}$	Range for CRAH flow	0.1-2.1 m^3/s
$T_{\mathrm{min/max}}^{\mathrm{CRAH}}$	Range for CRAH outlet	18-27 °C
$p_{\mathrm{fan}}^{\mathrm{SRV}}$	Server fan max power usage	50.4 W
$p_{\mathrm{max}}^{\mathrm{SRV}}$	Max load on a server	500 W
$Q_{\mathrm{min/max}}^{\mathrm{SRV_i}}$	Range for SRV$_\mathrm{i}$ flow	0.001-0.04 m^3/s

The model is simulated with steps of one second, and new states are updated based on previous states to allow for some dynamic behavior. The states are described in Table 1 and some constants for the environment are presented in Table 2.

The flow between CRAH and servers is modeled such that it creates either recirculation, where a fraction η of the hot airflows back into the servers, or bypass, where a fraction μ of the cold air flows back into the CRAH.

$$\eta(t) = \max\left(0, 1 - \frac{Q^{\mathrm{CRAH}}(t)}{\sum_i Q^{\mathrm{SRV_i}}(t)}\right)$$
$$\mu(t) = \max\left(0, 1 - \frac{\sum_i Q^{\mathrm{SRV_i}}(t)}{Q^{\mathrm{CRAH}}(t)}\right)$$

The flow is assumed to be mixed well and is approximated with a homogeneous temperature both at each server inlet

$$T_{\mathrm{in}}^{\mathrm{SRV}}(t+1) = (1-\eta(t))T_{\mathrm{out}}^{\mathrm{CRAH}}(t) + \eta(t)T_{\mathrm{out}}^{\mathrm{SRV}}(t)$$

and the CRAH inlet

$$T_{\mathrm{in}}^{\mathrm{CRAH}}(t+1) = (1-\mu(t))T_{\mathrm{out}}^{\mathrm{SRV}}(t) + \mu(t)T_{\mathrm{out}}^{\mathrm{CRAH}}(t).$$

The temperature out from the servers $T_{\mathrm{out}}^{\mathrm{SRV}}(t)$ is also approximated to mix well as soon as it exits the servers,

$$T_{\mathrm{out}}^{\mathrm{SRV}}(t) = \frac{\sum_i Q^{\mathrm{SRV_i}}(t)T_{\mathrm{out}}^{\mathrm{SRV_i}}(t)}{\sum_i Q^{\mathrm{SRV_i}}(t)},$$

where the individual outlet temperatures are updated as

$$T_{\mathrm{out}}^{\mathrm{SRV_i}}(t+1) = T_{\mathrm{SRV_i}}^{\mathrm{in}}(t) + \frac{p^{\mathrm{SRV_i}}(t)}{C_v Q^{\mathrm{SRV_i}}(t)}.$$

The CPU temperature in the server is updated according to

$$T_{\mathrm{SRV_i}}^{\mathrm{cpu}}(t+1) = T_{\mathrm{SRV_i}}^{\mathrm{in}}(t) + R\frac{p^{\mathrm{SRV_i}}(t)}{Q^{\mathrm{SRV_i}}(t)}$$

which will in turn affect the flow that is updated using an integral action controller,

$$Q^{\mathrm{SRV_i}}(t+1) = \mathrm{clip}\left(Q^{\mathrm{SRV_i}}(t) + \frac{60 - T_{\mathrm{SRV_i}}^{\mathrm{cpu}}(t)}{T_i}, Q_{\mathrm{min}}^{\mathrm{SRV}}, Q_{\mathrm{max}}^{\mathrm{SRV}}\right),$$

where the controller tries to keep the temperature around a setpoint of 60°C by varying the flow through the server.

The outlet temperature for the CRAH is one of the control variables, as is the CRAH flow. It is assumed that the CRAH will always be able to keep exactly this temperature and flow.

Distributing incoming load over the servers is also among the control variables. Arriving jobs have a load and a duration and each job is placed and processed on one of the servers. If the server cannot handle the added load the job is dropped.

The power used in the end is composed of two parts: the fan power from servers and CRAH, and the compressor power. The fan power follows affinity laws derived from dimensional analysis [3]

$$p^{\text{FAN}}(t) = p_{\text{fan}}^{\text{SRV}} \sum_i \left(\frac{Q^{\text{SRV}_\text{i}}(t)}{Q_{\text{max}}^{\text{SRV}}} \right)^3 + p_{\text{fan}}^{\text{CRAH}} \left(\frac{Q^{\text{CRAH}}(t)}{Q_{\text{max}}^{\text{CRAH}}} \right)^3 .$$

The compressor is modeled to turn on depending on the temperature and, when on, uses power based on the power removed from the air in the CRAH.

$$p^{\text{CMPR}}(t) = \begin{cases} 0 \text{ if } T^{\text{AMB}}(t) < T_{\text{out}}^{\text{CRAH}}(t) \\ C_v Q^{\text{CRAH}}(t) \left(T_{\text{in}}^{\text{CRAH}}(t) - T_{\text{out}}^{\text{CRAH}}(t) \right) & \text{otherwise} \end{cases}$$

This introduces a non-linearity such that power consumption increases by a large factor for CRAH temperature setpoints below the ambient temperature.

5 CONTROL USING RL

5.1 RL Intro

Reinforcement learning is an area of machine learning which focuses on learning how to optimally interact with an *environment*. An *agent* is an entity that learns and it uses a *policy* to map *states* onto *actions* when deciding what to do. To learn it also needs some measure of *reward* that should signify how good it is to be in a certain state. The agent learns by updating the policy in a way to try to maximize the cumulative reward.

The environment defines transitions between states based on actions, as well as the corresponding reward for ending up in a state. If the environment is not fully observable, it has incomplete *observations* of the state instead of the full state.

The basic RL loop used with the simplified DC model is illustrated in Figure 2.

Actor-Critic [13] methods are a category of RL-algorithms where the agent consists of an *actor* defining the policy $\pi(a \mid s, \boldsymbol{\theta})$, and a *critic* approximating a value function $\hat{v}(s \mid \boldsymbol{w})$.

The value function is an expectation over future cumulative reward, and is the unique solution to the Bellman equation

$$v(s) = \mathbb{E}_{s'}[r + v(s')]$$

where s' are possible successor states to s. It is also common to value a current reward higher than a future reward because of uncertainty, so a discount γ is added to the estimated future value. After parameterizing $\hat{v}$ with $\boldsymbol{w}$, the value can be estimated by minimizing the mean squared bellman error through gradient updates of $\boldsymbol{w}$ as

$$J(\boldsymbol{w}) = \left(r + \gamma\hat{v}(s' \mid \boldsymbol{w}) - \hat{v}(s \mid \boldsymbol{w})\right)^2$$
$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha_{\boldsymbol{w}} \nabla J(\boldsymbol{w})$$

where $\alpha_{\boldsymbol{w}}$ is the learning rate.

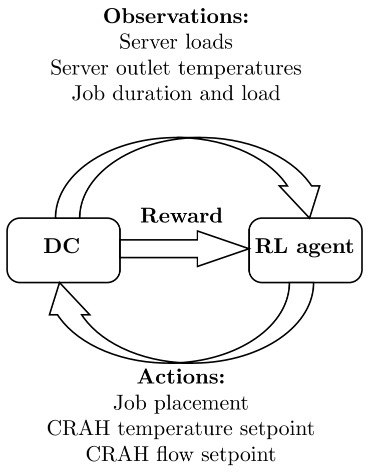

Figure 2: How the RL agent interacts with the DC.

The policy is updated using policy gradient,

$$\nabla J(\boldsymbol{\theta}) = \delta \nabla \log \pi(a \mid s, \boldsymbol{\theta})$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha_{\boldsymbol{\theta}} \nabla J(\theta),$$

this means $\boldsymbol{\theta}$ is updated to maximize the policy weighted by some estimation of advantage of this action over others. A common baseline for the advantage estimation is $\delta = r + \gamma\hat{v}(s' \mid \boldsymbol{w}) - \hat{v}(s \mid \boldsymbol{w})$, so if the action resulted in more value than expected (probably a good action) then $\delta > 0$ and the update will be weighted positively.

Proximal Policy Optimization (PPO) [11] is an actor-critic method that aims to reduce the step-size of policy updates to make the policy more stable.

The output of the policy network is converted to a distribution, allowing actions to be sampled in a non-deterministic way. It also allows for calculating the probability of a specific action under a specific policy which allows for comparing the action distributions between the previous and current policy. The policy loss

$$J^{CLIP}(\boldsymbol{w}) = \hat{\mathbb{E}}_t \left[\min\left(r_t(\boldsymbol{w})\hat{A}_t,\ \text{clip}(r_t(\boldsymbol{w}), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right],$$

is then using the fraction between these distributions

$$r_t(\boldsymbol{w}) = \frac{\pi(a_t \mid s_t)}{\pi_{old}(a_t \mid s_t)}.$$

to slow down the distributional shift when doing gradient updates. $\hat{A}_t$ is an advantage estimate showing how much more reward is gained compared to what is expected by the current value function. It is calculated using using GAE(1) [10].

5.2 RL Model

Ray rllib [7] is used as the RL platform along with their implementation of PPO adapted to fit with environments observations and actions used in this work. Figure 3 depicts the structure of the policy and value network used by the agent and how it relates to the observations and actions.

Observations from the environment are the current server loads and temperatures out from the servers, as well as load (p_{job}) and

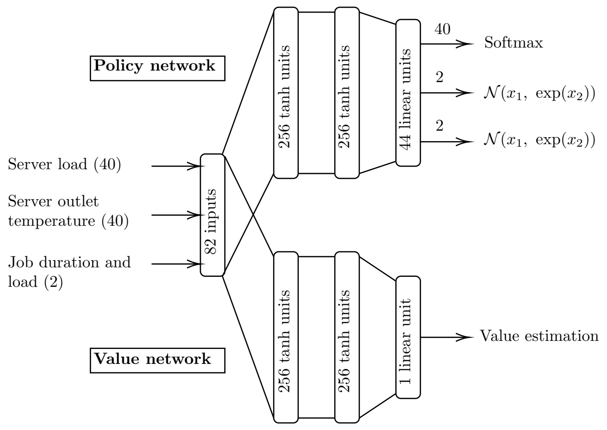

Figure 3: Structure of fully connected neural network layers used for the RL agent.

duration (D_{job}) of the incoming job

$$\left([T_{\text{out}}^{\text{SRV}_\text{i}}(t)],\ [p^{\text{SRV}_\text{i}}(t)],\ p_{\text{job}},\ D_{job} \right).$$

The observations are fed into both the policy and the value networks which are implemented as dense neural networks. Both networks have two hidden layers each with 256 units and tanh as the activation function.

Actions are sampled by the agent based on the most recent observation. Each action contains a server index where the incoming load should be placed together with flow and temperature setpoints for the CRAH unit

$$\left(i,\ T_{\text{out}}^{\text{CRAH}},\ Q^{\text{CRAH}} \right).$$

Rewards from the environment are designed to facilitate the agent in learning how to reduce cooling power while minimizing the number of dropped jobs (J_{drop}) and also adhering to an accepted standard from industry of keeping the cold isle below 27°C

$$C_1 (p^{\text{FAN}}(t) + p^{\text{CMPR}}(t))\Delta t + C_2 J_{drop} + C_3 \sum_i^N [T_{\text{in}}^{\text{SRV}_\text{i}} > 27]$$

There is an explicit weighting between reward terms which will punish not adhering to the cold-isle temperature or dropping jobs quite a bit harder than the cost of energy expenditure. This is done to encourage the agent to learn how to adapt to those terms first and then optimize energy usage based on those bounds.

There are several things that can be improved upon from this model, but this is a first step to show that the idea is viable. For one the algorithm selected (PPO) is not very sample efficient and might not be optimal for environments that are not possible to simulate very quickly. It was chosen since not all algorithms are able to handle a mix of discrete and continuous actions out of the box, and this was a convenient choice for the simple environment. It is also likely that the dense neural network model is not optimal for the current application, and much more efficient learning can come out of models adapted for the problem. It was used as a first attempt to interact with the described datacenter model, and see if any learning will come out of it. Future work will include how to better facilitate this learning and improve upon the methods used here, to make this able to scale to bigger problems and learn in a more efficient manner.

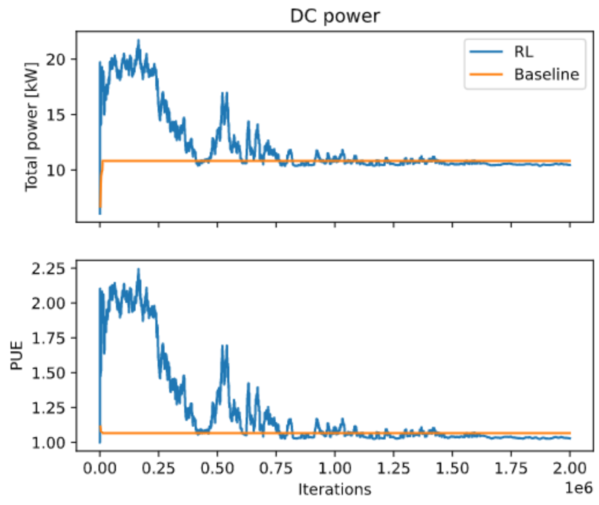

Figure 4: Total power usage and PUE for the simulated DC. The resulting shape is largely from the compressor being very expensive to run, see Figure 5. Before the RL agent learns to avoid this, the majority of the cost will come from the workload (around 10kW) and the compressor.

5.3 Result

This work starts off modeling a single rack with 40 servers and a single CRAH unit. Each job adds a load of 20W and has a duration of 400s (2.2Wh) with an arrival rate of one job per second. The ambient temperature is set to $T^{\text{AMB}}(t) = 20°\text{C}$. The performance of the RL agent is compared to a baseline method where the load balancing is handled by placing incoming jobs on the lowest load server and keeping the CRAH at 80% fan power and 22°C.

The data is sampled every 200 iterations and a moving average is created over the 100 most recent samples.

Figure 4 shows how the RL agent learns with time to control the DC to reduce the total power (and thus also the cooling power) to a lower level than the baseline algorithm. The final PUE values are 1.028 and 1.067 respectively for the RL and baseline policies. This is a rather small difference, but since it is not possible to achieve a PUE below 1 it is still a rather big relative improvement on a policy that is already running very close to optimal. To have a PUE this low might not be very realistic for most DCs, and is an artifact of the simplified model where many inefficiencies are removed.

Further, Figure 5 shows the compressor is causing a majority of the energy inefficiency at the beginning of the simulation. With time the RL agent learns to adapt the CRAH temperature setpoint so that the compressor will be left turned off most of the time.

For the workload used the average load on each server will be around 250 W assuming no jobs are dropped. In this simple model,

The code for both the model simulation and RL agent training is available at https://github.com/albheim/rldc-flowsim/releases/tag/article

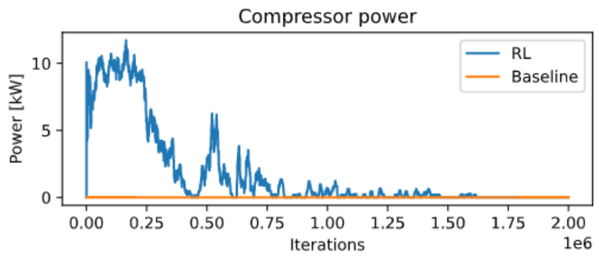

Figure 5: Compressor power is a large cost when on. Over time, the averaged power goes to zero.

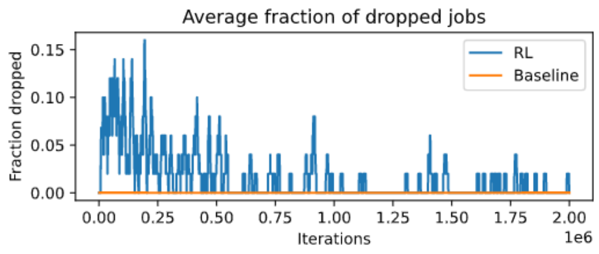

Figure 6: Fraction of jobs dropped over each averaging period.

there are no hot spots and an even distribution can be shown to be optimal from an energy perspective. Figure 6 shows what fractions of jobs are dropped from being placed on a full server. After the initial phase of dropping jobs, the agent learns to somewhat balance the load and ends up with a 250 W mean and a standard deviation of around 70 W over all servers.

In conclusion, the RL agent learns a competitive policy for the simulated environment. It balances the load with minimal servers being overloaded and manages the CRAH setpoints in a manner so as to minimize the compressor usage (and thus energy used) while still keeping the cold isle below the maximum.

6 SUMMARY AND FUTURE WORK

In this work, a simple model of the airflow and energy consumption in a small DC setup was developed. This model was then controlled by an RL agent that did both load balancing for incoming jobs and setpoint handling for the CRAH unit. The results were compared to some baseline algorithms which showed that the RL agent can learn to do both jobs at the same time and reach a level where it uses less power for cooling than the baseline.

This will hopefully just become more clear in a richer environment including features such as changing outside temperature and varying loads, where a simple strategy might not always fare as well, but an RL strategy will be more adaptive and gain a deeper understanding of the environment.

Future work. As next steps, the authors plan to allow for more varying features, such as changing outside temperature and varying loads, as well as, use models developed by Sjölund et al. [12] for training and validating the RL agent. The relative performance of the RL agent and baseline can then be investigated under simulation of a more realistic data center environment, to elicit whether the RL agent adapts better to increased complexity. Scaling of the model to multiple racks comes with challenges when the action and state spaces become very large andthe richer and more realistic models of the environment are also much slower to run. So to be able to train the RL agent in a reasonable time, sample efficient ways of learning are needed. Two approaches to this will be explored in future work. The first one looks at different RL algorithms able to make use of data that was not necessarily collected by the current policy. The algorithm could then be more sample efficient. The second one looks at the network structure of the policy. Imbuing domain knowledge into the network can simplify the knowledge representation and also make for a more efficient learning process.

REFERENCES

[1] Mumtaz Ahmad, Songyun Duan, Ashraf Aboulnaga, and Shivnath Babu. 2011. Predicting Completion Times of Batch Query Workloads Using Interaction-Aware Models and Simulation. In *Proceedings of the 14th International Conference on Extending Database Technology - EDBT/ICDT '11*. ACM Press, Uppsala, Sweden, 449. https://doi.org/10.1145/1951365.1951419
[2] J. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel. 2019. Managing Fog Networks Using Reinforcement Learning Based Load Balancing Algorithm. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. 1–7. https://doi.org/10.1109/WCNC.2019.8885745
[3] E. Buckingham. 1914. On Physically Similar Systems' Illustrations of the Use of Dimensional Equations. *Physical Review* 4, 4 (oct 1914), 345–376. https://doi.org/10.1103/physrev.4.345
[4] C.M. Healey, J.W. VanGilder, M. Condor, and W. Tian. 2018. Transient Data Center Temperatures after a Primary Power Outage. *Proceedings of the 17th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, ITherm 2018* (2018), 865–870. https://doi.org/10.1109/ITHERM.2018.8419583
[5] Nevena Lazic, Tyler Lu, Craig Boutilier, MK Ryu, Eehern Jay Wong, Binz Roy, and Greg Imwalle. 2018. Data center cooling using model-predictive control. (2018).
[6] Yuanlong Li, Yonggang Wen, Kyle Guan, and Dacheng Tao. 2018. Transforming Cooling Optimization for Green Data Center via Deep Reinforcement Learning. *arXiv:1709.05077 [cs]* (July 2018). arXiv:cs/1709.05077
[7] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Jennifer Dy and Andreas Krause (Eds.), Vol. 80. PMLR, Stockholmsmässan, Stockholm Sweden, 3053–3062. http://proceedings.mlr.press/v80/liang18b.html
[8] Riccardo Lucchese, Jesper Olsson, Anna-Lena Ljung, Winston Garcia-Gabin, and Damiano Varagnolo. 2017. Energy savings in data centers: A framework for modelling and control of servers' cooling. *IFAC-PapersOnLine* 50, 1 (2017), 9050–9057.
[9] M. K. Patterson. 2008. The effect of data center temperature on energy efficiency. In *2008 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*. 1167–1174. https://doi.org/10.1109/ITHERM.2008.4544393
[10] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2018. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *arXiv:1506.02438 [cs]* (Oct. 2018). arXiv:cs/1506.02438
[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).
[12] Johannes Sjölund, Mattias Vesterlund, Nicolas Delbosc, Amirul Khan, and Jon Summers. 2018. Validated Thermal Air Management Simulations of Data Centers Using Remote Graphics Processing Units. In *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 4920–4925.
[13] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.
[14] P. Townend, S. Clement, D. Burdett, R. Yang, J. Shaw, B. Slater, and J. Xu. 2019. Invited Paper: Improving Data Center Efficiency Through Holistic Scheduling In Kubernetes. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. 156–15610. https://doi.org/10.1109/SOSE.2019.00030
[15] James W. VanGilder, Christopher M. Healey, Michael Condor, Wei Tian, and Quentin Menusier. 2018. A Compact Cooling-System Model for Transient Data Center Simulations. In *2018 17th IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*. IEEE. https://doi.org/10.1109/itherm.2018.8419515