Thermal and Energy Management with Fan Control Through Offline Meta-Reinforcement Learning

Shao-Yu Yen, Yen-Ru Lai, Fu-Chieh Chang, and Pei-Yuan Wu
Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan
{p09942a02, r09942079, d09942015, peiyuanwu}@ntu.edu.tw

Abstract

Reinforcement learning has garnered significant attention across various fields, including computer vision, natural language processing, and robotics. In this work, we explore the potential of applying reinforcement learning to open-world agents through an empirical study of three distinct offline meta-reinforcement learning approaches for fan control, with a focus on thermal and energy management. Our models enable adaptive fan speed control, which not only protects devices from overheating but also effectively reduces power consumption. To better evaluate the performance in open-world scenarios, we go beyond the industry-standard steady-state test by conducting a CPU-stress test that simulates a more dynamic and unpredictable deployment environment. Compared to commercially available techniques, our solution achieves up to a 21% reduction in power consumption on a real 2U-server under the worst thermal conditions. This approach demonstrates the broader applicability of meta-reinforcement learning in the thermal and energy management of server systems, particularly in open-world settings.

1 Introduction

AI and IoT advancements have led to server clusters in data centers, posing significant thermal management challenges for performance and fault tolerance. Excessive heat from high power consumption hardware threatens service reliability and shortens device lifespans, necessitating increased cooling efforts which in turn require more energy. Amid global environmental concerns, it's vital to enhance server cooling efficiency to minimize energy use and prevent overheating.

Recent studies have applied ML to improve thermal management. For example, neural networks combined with PID controllers[8] have optimized airflow in vacuum systems, while a thermal-aware load-balancing algorithm [1] predicted CPU temperatures for better fan control. Notably, reinforcement learning, particularly Q-Learning[13, 2], has been effective to minimize future temperatures and reduce fan power consumption, showcasing significant energy savings compared to traditional methods.

Research primarily uses online RL for server fan control, benefiting from simulated data for quick policy training. Yet, these studies often ignore critical variables like airflow and hardware differences, affecting fan control's success and performance. In our experience, applying online RL on actual servers can cause stability issues, including heat crashes, and adapting online RL to different hardware setups is also challenging, as a single server may have over 10 distinct configurations, each needing a week for engineers to adjust the fan curve appropriately.

Consequently, Offline meta-RL methods stand out by learning from existing data, allowing for quick adaptation to new settings without needing large new datasets. These methods, enhanced by some online fine-tuning [9], significantly improve performance by applying previously trained policies to gather more data, optimizing the overall effectiveness of the implementation.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

This paper presents an empirical study involving three distinct offline meta-reinforcement learning approaches for the task of fan control, focusing on thermal and energy management. The chosen approaches include BCQ [6] in conjunction with MAML [4] (MAML-BCQ), MACAW [15], and FOCAL [10]. In previous studies, [2] conducted comparisons with fixed fan policies. The data collection methods in [7] and [28] involved a software-simulated environment and a server-like environment, respectively. However, our methodology encompasses both offline learning and subsequent online fine-tuning, all conducted on a real commercial server, thus expanding the scope beyond previous works. Our server environment is equipped with 2 CPUs, 32 DIMMS, and 4 fan zones, In our experiment, we consider 4 different CPU configurations and 2 testing scenarios. Moreover, our investigation incorporates a broader set of server information than previous studies. The inclusion of CPU power as an input enables our RL model to preemptively adjust fan speeds, averting CPU temperature escalation. This proactive control mechanism facilitates swift self-adaptive temperature management and enhances energy efficiency.

The experiments conducted in previous RL-based fan control literature are often limited and may not reflect real-world scenarios properly, [2] examines the transition from IDLE to CPU stress, yet it is compared to fixed fan policies, which is not a decent real-world benchmark. Additionally, [7] explores the transition from IDLE to CPU stress and compares it with [21], which however does not represent a fan control approach. Our experiments thus delve into more dynamic scenarios on a commercial server, where our fan policies are compared with the commercially available fan policy on the server.

Notably, this study represents a novel approach in the field by integrating both energy and thermal considerations into server fan control via an offline RL-based algorithm, an area largely unexplored in existing literature. To contribute to the field, we compare different offline meta-RL models with commercially available techniques, emphasizing the applicability of our solution in open-world scenarios. Our contributions are summarized as follows:

- This is the first work to experiment with different offline meta-RL frameworks that leverage MAML-BCQ, MACAW, and FOCAL to address the fan control problem regarding power consumption and thermal management on a real server environment.
- To better assess the fan control performance, besides the industry-standard steady-state test, we also conduct CPU-stress test to simulate a more general deployment scenario for the server where the workload is more random and dynamic.
- We introduce CPU power and fine-tuning into the field of fan control, exploring the efficacy of capturing thermal control relevance using the reinforcement learning model, particularly during the transient of hardware loading changes.
- The efficacy of our methodologies is established through comparison with fan policy actually deployed in commercial servers under industry-standard test cases. Our fan control policy achieves 21%/19% reduction in power consumption under the steady-state/CPU-stress tests, respectively.

2 Related Work

2.1 Offline Reinforcement Learning

Offline RL methods resemble supervised learning, transforming data into models that generalize well. They allow for effective training of RL agents using pre-existing data, eliminating the need for real-time data acquisition, crucial in domains where live data gathering is difficult. Yet, offline RL faces challenges, especially when models trained on specific data encounter new situations, leading to significant extrapolation errors in predictions and value estimations. This discrepancy can cause policies to perform poorly, diverging from their initial training. We refer the reader to prior work [6] for a detailed discussion. Addressing these challenges often involves restricting certain learning aspects to curb the impact of distributional shifts [18].

BCQ [6], a leading method in offline RL, constrains the action space to match the behavior policy, avoiding queries on out-of-distribution actions. AWR [16] uses an implicit KL-divergence constraint for offline RL through two steps: value function fitting and optimizing policy improvement with the Lagrange multiplier method. BRAC [26] introduces an explicit KL-divergence penalty, ensuring

policy alignment with behavior policy and minimizing future deviations. FOCAL [10] extends BRAC for offline meta-RL, a technique we employ in our studies. ExORL [27] shows using varied unlabeled datasets with reward relabeling can boost offline RL, highlighting the need for better real-world data relabeling methods.

2.2 Offline Meta-Reinforcement Learning

Meta-reinforcement learning (meta-RL) helps agents learn quickly in new settings with few samples by training on diverse environments, facilitating rapid adaptation. Offline meta-RL merges offline learning's efficiency with meta-learning's adaptability. MACAW[15] utilizes MAML [4] and AWR's [16] principles, plus a weight transform [3] to improve MuJoCo [24] performance. FOCAL [10] , an offline actor-critic method [19], uses an autoencoder for environmental context and regularization for policy stability, achieving fast adaptation with limited data. SMAC [17], focusing on learning from offline data for real-time task application, uses self-supervision to improve adaptability but requires an online meta-training phase, leading to prolonged exploration time for practical use.

2.3 Reinforcement Learning on Thermal Management

[12] and [14] developed RL-based CPU task schedulers, with [14] employing PPO [20] for thermal and energy-efficient scheduling. [7] applied Q-learning [25] in 3D-ICE [22] simulations to manage processor temperature through DVFS, fan speed, and core activation adjustments, achieving a 19% power saving compared to [21]'s method of increasing fan speed upon thermal threshold breaches. Meanwhile, [28] used an actor-critic model with bimodal airflow-temperature sensing for temperature management. Unfortunately, those previous works only experiment with server-like simulated environments, which oversimplifies the thermal fan zones in the server and underestimates how hardware configurations affect dissipating heat.

3 Thermal Fan Control Benchmark

As mentioned in Sec. 1, the fan control problem lacked open datasets and benchmarks. To address this, we collected our data using the ASUS RS720-E10-RS12E. In Fig. 1, a 2U server is equipped with 2 CPUs, 32 DIMMs, 12 HDDs, 2 PSUs, and 4 fans. We utilized the default fan policy provided by ASUS BMC to establish our benchmark. BMC refers to Baseboard Management Controller, which is the microcontroller system migrated into the server, with functions including remote monitoring of server data, remote server control, and control of the server fan. The power consumption of two scenario is shown in Table 1. Based on these two scenario, we compare our model with BMC policy, as to be elaborated below:

Steady-state Test: The server consists of 4 test cases. Case 1 represents the IDLE mode, Case 2 represents the CPU stress mode, Case 3 represents the MEM stress mode, and Case 4 represents the CPU+MEM stress mode. As depicted in Fig. 2, each test case runs for 15 minutes, with the last 7.5 minutes representing the steady state (behind the black dotted line area). During the steady-state test, the model will maintain the fan duty within a range of $\pm 2\%$. To consider the worst-case scenario for thermal management in the steady state test, we chose the configuration consisting of 2 205W CPUs and 32 8GB DIMMs, representing the highest power consumption for CPUs of the RS720-E10-RS12E server hardware specification.

CPU-stress Test: To simulate the real-world scenarios where the server encounters unpredictable workload request traffic, in CPU-stress test we deliberately introduce frequent changes to the CPU power and stress modes in a total of 42 times within a 15-minute timeframe, as shown in Fig. 3.

Table 1: The Benchmark Power Consumption on Steady State Test and CPU Stress Test

	Sum (Watt)	Avg (Watt)
ASUS BMC on Steady-state Test	695887	644.33
ASUS BMC on CPU-stress Test	296373.5	658.6

(a) Internal structure (b) Fan Zone configuration

Figure 1: Diagrams of the RS720-E10-RS12E server. The hard drives (HDDs) are positioned in front of the fans, followed by the CPUs and dual in-line memory modules (DIMMs), and at the end are the power supply units (PSUs) and PCIs.

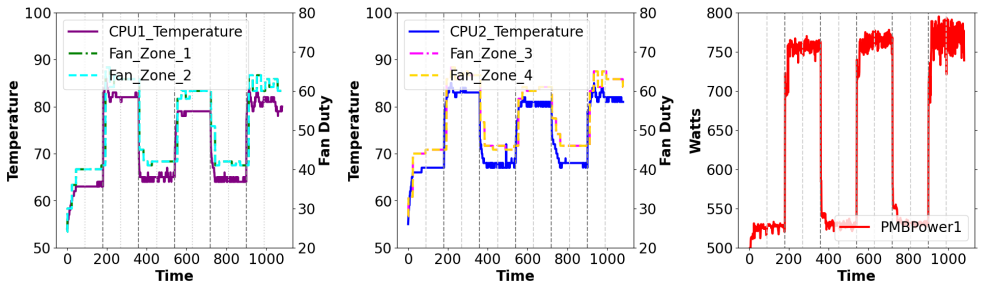


Figure 2: Benchmark Fan Policy on Steady State Test. To compare with ASUS RS720-E10-RS12E, we considered the server's internal airflow and hardware design for visualization of the benchmark. The sequence of case switches is given by: case1 $\rightarrow$ case2 $\rightarrow$ case1 $\rightarrow$ case3 $\rightarrow$ case1 $\rightarrow$ case4, where each switch between cases is indicated by a dark grey dashed line. The left and middle plots in this figure represent the variations between the fan duty of fan zones and CPU temperatures under different test cases. The right plot illustrates the changes in power supply output under various test cases. Due to the highest power consumption reaching 820 watts in the worst case (CPU+DIMM), only PSU1 is utilized as it can supply up to 1600 watts, while PSU2 was not activated.

4 Algorithmic Review

4.1 Reinforcement Learning

Reinforcement learning addresses the challenge of learning to control a dynamical system, typically represented by a Markov decision process (MDP), which can be defined by a tuple of $(S, A, T, d_0, r, \gamma)$: An agent in a state $s_t \in S$ interacts with the environment by taking an action $a_t \in A$, and the environment responds with a new state $s_{t+1} \in S$ according to a transition probability distribution $T(s_{t+1}|s_t, a_t)$ that describes the dynamics of the system. d_0 is the initial state distribution, $r : S \times A \rightarrow \mathbb{R}$ defines a reward function measuring how beneficial that interaction was toward the goal of the agent, and $\gamma \in [0, 1]$ is the scalar discount factor. The ob-

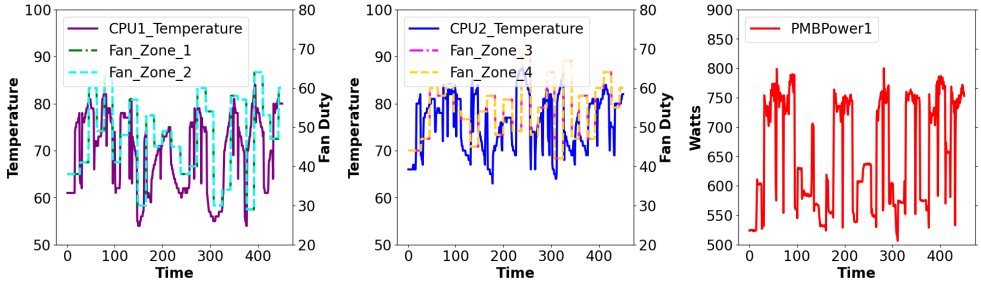


Figure 3: Benchmark Fan Policy on CPU Stress Test. The difference from Fig. 2 is the frequency of case switching. We simulated the scenario faced by servers deployed in real-world environments by frequently switching between different stress test cases.

jective is to learn a policy $\pi(a_t|s_t)$ which defines a distribution over actions conditioned on states. A trajectory is a sequence of states and actions of length H, given by $\tau = (s_0, a_0, ..., s_H, a_H)$, where H may be infinite. The probability density function for a given trajectory τ under policy π is given by $p_\pi(\tau) = d_0(s_0) \prod_{t=0}^{H} \pi(a_t|s_t) T(s_{t+1}|s_t, a_t)$. The objective function $J(\pi)$ is the expected (discounted) cumulative reward under policy π, namely: $J(\pi) = \mathbb{E}_{\tau \sim p_\pi} \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t) \right]$. The optimal policy $\pi^* = \arg\max_\pi J(\pi)$ is the policy that maximizes the expected (discounted) cumulative reward. Approaches like policy iteration and value iteration often utilize state-value function V^π and state-action-value function Q^π to find the optimal policy. The value functions offer an estimate of the expected cumulative reward attainable by following a certain policy π when starting from a specific state s (for the state-value function) or state-action pair (s, a) (for the state-action value function), namely: $V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t) | s_0 = s \right]$, and $Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{H} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right]$. The value functions satisfy the well-known Bellman equations [23]: $V^\pi(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} [Q^\pi(s, a)]$, and $Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim T(\cdot|s,a)} [V^\pi(s')]$. The advantage function $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ indicates the advantage or benefit of choosing a specific action a in comparison to the average expected performance from state s.

4.2 Offline Meta-Reinforcement Learning

Conventional offline RL algorithms require the collection of extensive new data from a new task to train the policy, which is time-consuming in practice. We can opt-in meta-learning to offline RL algorithms to enhance the agent's ability to faster adapt the policy to new tasks in the open-world scenario. Existing offline meta-reinforcement learning problems typically define a task $\mathcal{T}_i$ to represent a tuple (M_i, μ_i), where M_i represents a fixed MDP and μ_i is an unknown behavior policy. The behavior policy μ_i may vary, depending on the replay buffer of an RL agent, and it could be either an expert policy or a sub-optimal policy. In the offline setting, an offline meta-RL algorithm lacks direct interaction with the environment. Instead, it accesses static datasets of transitions $\mathcal{D}_i = \{(s_{i,t}, a_{i,t}, s'_{i,t}, r_{i,t})\}_{t=1}^{N}$ sampled from μ_i for each task, where $s'_{i,t}$ is the next state.

In this work, we implement three distinct offline meta-RL algorithms to deploy on the ASUS RS720-E10-RS12E, which are Model-Agnostic Meta-Learning (MAML) [4] combining with Batch-Constrained deep Q-Learning (BCQ) [6], called (MAML-BCQ), Meta Actor-Critic with Advantage Weighting (MACAW) [15], and Fully-Offline Context-based Actor-critic meta-RL (FOCAL) [10], all based on MAML framework to achieve meta-learning. In MAML, the adaptation learning process known as the inner-loop, involves an inner learning algorithm solving tasks defined by the buffer $\mathcal{D}_i$. Meanwhile, during meta-learning, an outer algorithm updates the inner learning algorithm to improve an outer objective, referred to as the outer-loop. We will provide a concise overview of those three algorithms in Appendix A.

5 Implementation Details and Reward Function

5.1 Implementation Details

In this project, we used PyTorch on a server with an NVIDIA GeForce RTX 3080Ti GPU to implement three models for offline meta-learning, testing four CPU power settings (130W, 150W, 190W, 205W) on an ASUS RS720-E10-RS12E server. Initially, we ran 500 online learning episodes for AWR [16] warm-up, then stored data from the next 100 episodes, totaling 108,000 time steps of server data including CPU temperatures, power inputs, thermal sensor readings, power usages, and fan duties. Data was normalized to a $[0, 1]$ range.

We tested our models to reduce server fan power usage, training them on 130W, 150W, and 190W CPU configuration across 200,000 steps in offline mode. Each step involved selecting an episode at random from these configurations and training on a 256-transition batch. We also fine-tuned the models with a 205W CPU configuration over five rounds.

For our fine-tuning approach, we initially trained our models offline on 130W, 150W, and 190W CPU data for 40,000 steps, then shifted to online fine-tune with 205W CPU configuration over 5

episodes, and collected relevant data. Then, we used this new dataset for another 40,000-step training. Repeating this fine-tuning four times greatly improved our models' adaptability and power efficiency.

Furthermore, we simplified MACAW's architecture by using fully connected layers instead of weight transform layers to lower computational demands and applied gradient clipping to prevent exploding gradients. In our FOCAL experiments, replacing LSTM encoders with fully connected layers countered the rapid CPU temperature rise during stress tests, due to LSTM's slow inference delaying fan response.

We evaluated our models using steady-state and CPU-stress tests benchmarks (see Sec. 3) on a server with 2 205W CPUs and 32 8GB DIMMs, mimicking RS720-E10-RS12E's maximum power setup for extreme thermal conditions.

5.2 Reward Function

In this work, the reward function is chosen as:

$$\mathcal{R}(s,a) = \begin{cases} -540, & \text{if } T \geq T_c, \\ -5, & \text{if } T_c > T \geq T_a, \\ \alpha(1-\sqrt{\frac{P}{P_{max}}})^2 & \\ \quad +\beta \tan^{-1}(1-\frac{F}{F_{max}}), & \text{otherwise.} \end{cases} \tag{1}$$

where T is the current CPU temperature, T_c is the CPU critical temperature, and T_a is the temperature close to the CPU critical temperature (i.e., alert temperature). In this work, we set $T_a = T_c - 5$. Notably, if $T \geq T_a$, we set the fan duty to 100% to minimize the risk of overheating. P is the output of power supply, F is the fans' power consumption.

Taking into account both the long-term power consumption of the system and the short-term power consumption of the fans, the reward function (1) is designed to achieve our goal of maximizing fan control performance while minimizing power consumption and avoiding heat crash issues. More elaborately, $(1-\sqrt{\frac{P}{P_{max}}})^2$ represents the total system power consumption, which helps our model to identify the server workloads; $\tan^{-1}(1-\frac{F}{F_{max}})$ represents the fan control aspect, with an aim to minimizing fan power within the thermal constraint, where P_{max} refers to the maximum wattage that the server's Power Supply Unit (PSU) can provide, while F_{max} denotes the highest rotation speed of the fan. However, given the conflicting objectives of minimizing power consumption and maximizing fan control performance, we introduced trade-off factors to navigate this challenge. The constants α and β, determined by their respective significance, denote the priority assigned to each objective, satisfying the constraint $\alpha + \beta = 1$. In our experiments, we set $\alpha = 0.3$ and $\beta = 0.7$ across all settings.

6 Experimental Results

6.1 Compared with ASUS BMC Fan Policy

Table 2 shows our models excel in both test scenarios, efficiently managing server fans to avoid CPU overheating and maintain stability. In the steady-state test, our top fan control policy cuts power use by 21% against the ASUS BMC policy, and in the CPU-stress test, it saves 19% versus the benchmark. The cumulative power consumption of the fan policies of the three models and BMC fan policy during the steady-state test is illustrated in Fig. 4. Additionally, we compare the MACAW fan policy and the BMC fan policy in case 1 switches to case 2, shown in Fig. 5. Besides, Fig. 4 illustrates how fine-tuning markedly enhances each model's power efficiency.

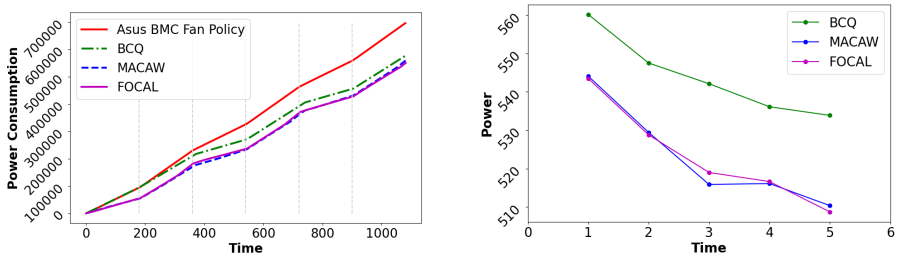
6.2 Ablation Study

In this part, we ablate the 2 key features of our proposed model to better understand our method.

Fine-tuning: Our fine-tuning tests, detailed in Table 3, demonstrated that fine-tuned models outperformed non-fine-tuned ones, achieving over 2% power savings in steady-state tests. This success is due to the models' enhanced use of online experience, proving the approach's applicability in real-world fan policy adjustments.

Table 2: ASUS BMC Fan Policy v.s. Offline Meta-RL.

Model	FT	CPU Power	Avg (Watt)	± %	FT	CPU Power	Avg (Watt)	± %
	Steady-state Test				CPU Stress Test			
ASUS BMC Policy	-	-	644.33	-	-	-	658.6	-
MAML-BCQ	Yes	Yes	533.87	-19%.	Yes	Yes	543.55	-17.4%
MACAW	Yes	Yes	510.34	-20.8%.	Yes	Yes	536.25	-18.5%
FOCAL	Yes	Yes	508.71	-21%.	Yes	Yes	533.8	-18.9%



(a) Cumulative Power Consumption on Steady-state Test

(b) Fine-tuning Improvement on Steady-state Test

Figure 4: In the left figure, our meta-RL models have significant improvements in power consumption compared with the original server fan policy. In the right figure, after each round of fine-tuning for each model, we observed a gradual improvement in the performance of power consumption throughout the process.

CPU Power Information: Experiments comparing models with and without CPU power data, as shown in Table 3, revealed that including CPU power led to up to 2.9% power savings on CPU-stress tests. Incorporating CPU power as an input is key, helping models better understand server load and improve efficiency.

7 Conclusion

In this work, we experiment with 3 different offline meta-RL approaches, namely MAML-BCQ, MACAW, and FOCAL, to the fan control problem regarding power consumption in a real server environment. To better assess the fan control performance, besides the industry-standard steady-state test, we also conduct CPU-stress test to simulate a more general deployment scenario for the server where the workload is more random and dynamic. Compared to commercially available products, our fan control policy achieves 21%/19% reduction in power consumption under the steady-state/CPU-

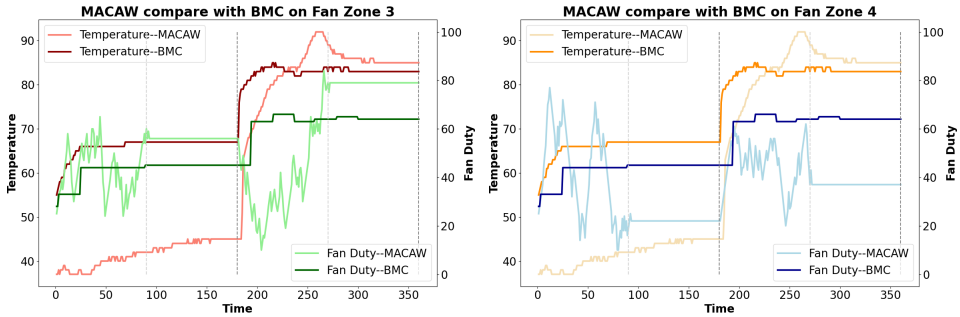


Figure 5: Fan duty companion of BMC and MACAW on Steady State Test. Compared to the BMC, the fan duty output of MACAW for Fan Zone 3 is higher than Fan Zone 4 when the case switches from IDLE Mode to CPU Stress Mode (the orange dote area), which is reasonable as Fan Zone 3 affects CPU2 more than Fan Zone 4 (cf. Fig. 1)

Table 3: Experiment results of ablation study.

Model	FT	CPU Power	Avg (Watt)	± %	FT	CPU Power	Avg (Watt)	± %
	Steady-state Test				CPU Stress Test			
MAML-BCQ	Yes	Yes	533.87	-	Yes	Yes	543.55	-
MAML-BCQ	**No**	Yes	552	+3.3%	Yes	**No**	557.8	+2.5%
MACAW	Yes	Yes	510.34	-	Yes	Yes	536.25	-
MACAW	**No**	Yes	520.8	+2%	Yes	**No**	548.69	+2.2%
FOCAL	Yes	Yes	508.71	-	Yes	Yes	533.8	-
FOCAL	**No**	Yes	520.12	+2.2%	Yes	**No**	549.78	+2.9%

stress tests, respectively. Our models effectively manage the complexities of thermal environments within the server, adeptly handling interactions among various heat sources, airflow, and transient operations. Despite the success, the limited hardware configuration we made regarding task training and inference can potentially limit offline meta-RL algorithms' robustness to common challenges such as more complex environments or stochastic environments, which opens up avenues for future work of more advanced implementation of offline meta-RL algorithms.

Acknowledgment

We extend our gratitude to the ASUS server team for their invaluable contribution to this research project(110-2622-8-002-017, 111-2622-8-002-031). Their passionate participation and insights were indispensable in the successful execution of this study. This work was supported in part by the Ministry of Education (MOE) of Taiwan under Grant NTU-113L891406; and in part by the National Science and Technology Council, R.O.C., under Grant 112-2622-8-002-022; and in part by the ASUS under Project 112CB244; and in part by the Asian Office of Aerospace Research Development (AOARD) under Grant NTU-112HT911020; and in part by the Center of Data Intelligence: Technologies, Applications, and Systems, National Taiwan University (grant nos. 113L900901/113L900902/113L900903), from the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) of Taiwan.

References

[1] Bilge Acun, Eun Kyung Lee, Yoonho Park, and Laxmikant V. Kale. Neural network-based task scheduling with preemptive fan control. In2016 4th International Workshop on Energy Efficient Supercomputing (E2SC), pages 77–84, 2016.

[2] Wen-Xiao Chu, Yun-Hsuan Lien, Kuei-Ru Huang, and Chi-Chuan Wang. Energy saving of fans in air-cooled server via deep reinforcement learning algorithm. Energy Reports, 7:3437–3448, 2021.

[3] Chelsea Finn. Learning to Learn with Gradients. PhD thesis, EECS Department, University of California, Berkeley, Aug 2018.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. InInternational conference on machine learning, pages 1126–1135. PMLR, 2017.

[5] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. InInternational Conference on Learning Representations, 2018.

[6] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. InInternational conference on machine learning, pages 2052–2062. PMLR, 2019.

[7] Arman Iranfar, Federico Terraneo, Gabor Csordas, Marina Zapater, William Fornaciari, and David Atienza. Dynamic thermal management with proactive fan speed control through

reinforcement learning. In2020 Design, Automation Test in Europe Conference Exhibition (DATE), pages 418–423, 2020.

[8] Li Jinyang and Meng Xiaofeng. Temperature decoupling control of double-level air flow field dynamic vacuum system based on neural network and prediction principle. Engineering Applications of Artificial Intelligence, 26(4):1237–1245, 2013.

[9] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643, 2020.

[10] Lanqing Li, Rui Yang, and Dijun Luo. FOCAL: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization. InInternational Conference on Learning Representations, 2021.

[11] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

[12] Shiting Lu, Russell Tessier, and Wayne Burleson. Reinforcement learning for thermal-aware many-core task allocation. InProceedings of the 25th edition on Great Lakes Symposium on VLSI, pages 379–384, 2015.

[13] Shiting (Justin) Lu, Russell Tessier, and Wayne Burleson. Reinforcement learning for thermal-aware many-core task allocation. InProceedings of the 25th Edition on Great Lakes Symposium on VLSI, GLSVLSI '15, page 379–384, New York, NY, USA, 2015. Association for Computing Machinery.

[14] Sudipa Mandal, Krushna Gaurkar, Pallab Dasgupta, and Aritra Hazra. An rl based approach for thermal-aware energy optimized task scheduling in multi-core processors. In2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID), pages 181–186. IEEE, 2021.

[15] Eric Mitchell, Rafael Rafailov, Xue Bin Peng, Sergey Levine, and Chelsea Finn. Offline meta-reinforcement learning with advantage weighting. InInternational Conference on Machine Learning, pages 7780–7791. PMLR, 2021.

[16] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177, 2019.

[17] Vitchyr H Pong, Ashvin V Nair, Laura M Smith, Catherine Huang, and Sergey Levine. Offline meta-reinforcement learning with online self-supervision. InInternational Conference on Machine Learning, pages 17811–17829. PMLR, 2022.

[18] Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. IEEE Transactions on Neural Networks and Learning Systems, page 1–0, 2023.

[19] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. InInternational conference on machine learning, pages 5331–5340. PMLR, 2019.

[20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347, 2017.

[21] Gaurav Singla, Gurinderjit Kaur, Ali K. Unver, and Umit Y. Ogras. Predictive dynamic thermal and power management for heterogeneous mobile platforms. In2015 Design, Automation Test in Europe Conference Exhibition (DATE), pages 960–965, 2015.

[22] Arvind Sridhar, Alessandro Vincenzi, David Atienza, and Thomas Brunschwiler. 3d-ice: A compact thermal model for early-stage design of liquid-cooled ics. IEEE Transactions on Computers, 63(10):2576–2589, 2014.

[23] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. MIT press, 2018.

[24] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 5026–5033, 2012.

[25] Christopher JCH Watkins and Peter Dayan. Q-learning. Machine learning, 8:279–292, 1992.

[26] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. arXiv preprint arXiv:1911.11361, 2019.

[27] Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. InICLR 2022 Workshop on Generalizable Policy Learning in Physical World, 2022.

[28] Zhen Zhang, Cheng Ma, and Rong Zhu. Thermal and energy management based on bimodal airflow-temperature sensing and reinforcement learning. Energies, 11(10):2575, 2018.

A Offline Meta-Reinforcement Learning Methods

MAML-BCQ: As mentioned in Sec. 2, existing off-policy reinforcement learning algorithms often overlook extrapolation error by selecting actions based on a learned value estimate, without assessing the accuracy of the estimate. BCQ [6] addresses the concept of batch constraint using a conditional variational auto-encoder (VAE) G_η, which is parameterized by η. The VAE generates a set of candidate actions that closely resemble the batch data, where the action with the highest value as estimated by a learned Q-network Q_θ (parameterized by θ) is selected. The policy is given by: $\pi_\theta(s) = a_n + \delta_\phi(s, a_n, \Phi)$, and $\delta_\phi(s, a_n, \Phi) = \arg\max_{\Delta a \in [-\Phi, \Phi]: a_n + \Delta a \in A} Q_\theta(s, a_n + \Delta a)$, where a_n is an action generated from G_η. The $\delta_\phi(s, a_n, \Phi)$ is a perturbation model parameterized by ϕ that outputs an adjustment to an action a in the range $[-\Phi, \Phi]$. BCQ is able to learn favorably without interacting with the environment by constraining the action selection, which ensures that the Q-function is never queried on out-of-distribution actions.

We applied MAML [4] to extend BCQ on offline meta-RL setting. The critical insight lies in considering the inner-loop as a differentiable function of the initial parameters. Consequently, the initialization can be optimized using gradient descent to serve as a strong starting point for learning tasks drawn from the task distribution. At each meta-training step, the inner loop adapts θ to a task $\mathcal{T}_i$ by computing $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(\theta)$, where $\mathcal{L}_{\mathcal{T}_i}$ is the loss function for task $\mathcal{T}_i$ and α is the step size. The outer loop updates the parameters as $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i} \mathcal{L}'_{\mathcal{T}_i}(\theta_i')$, where $\mathcal{L}'_{\mathcal{T}_i}$ is a loss function for task $\mathcal{T}_i$, which may or may not be the same as the inner-loop loss function $\mathcal{L}_{\mathcal{T}_i}$, and β is the outer loop step size. The outer-loop optimizes the initial parameter θ by taking the gradient of a loss function that depends on several gradient descent steps in the inner-loop. Therefore, the outer-loop requires taking the gradient of a gradient, which involves second-order derivatives. In practice, people often applied First-Order MAML (FOMAML) [4] which is MAML that ignores the second-order term. However, while it can speed up model training, it may discard some information from the inner loops.

MACAW: MACAW [15] draws inspiration from MAML [4] and AWR [16]. MACAW is an offline meta-reinforcement learning algorithm. The model architecture has three outputs: $\pi_\theta(\cdot|s)$, $A_\theta(s, a)$ and V_ϕ, parameterized by θ and ϕ, where $\pi_\theta(\cdot|s)$ corresponds to the predicted action, while $A_\theta(s, a)$ represents the predicted advantage output from the advantage regression block of the actor model, given both state and action. Finally, V_ϕ denotes the learned value function. Similar to AWR[16], MACAW formulates a supervised regression problem which learns the value function V_ϕ with the guidance of the Monte Carlo returns. More elaborately, MACAW adapts the value function V_ϕ by taking a few gradient descent steps aiming to minimize the sum of squared error loss: $\mathcal{L}_{critic} = \mathbb{E}_{s,a \sim \mathcal{D}_i}[(V_\phi(s) - (R_{\mathcal{D}_i}(s, a)))^2]$, where $R_{\mathcal{D}_i}(s, a)$ is the Monte Carlo return from the state s taking action a observed in $\mathcal{D}_i$.

MACAW updates its policy π_θ through $\theta_i' \leftarrow \theta - \psi \nabla_\theta \mathcal{L}_{actor}(\theta, \phi_i', \mathcal{D}_i)$, with $\mathcal{L}_{actor} = \mathcal{L}_{AWR} + \lambda \mathcal{L}_{ADV}$, ψ is the step size for the actor model, λ is the advantage regression coefficient. Here $\mathcal{L}_{AWR}(\theta, \phi_i', \mathcal{D}_i) = \mathbb{E}_{(s,a) \sim \mathcal{D}_i}\left[-\log \pi_\theta(a|s) \exp\left(\frac{1}{T}\left(R_{\mathcal{D}_i}(s, a) - V_{\phi_i'}(s)\right)\right)\right]$, where $T > 0$ is a temperature parameter, that aligns with the previous work AWR [16] (cf. Sec. 2.1). However, directly applying the same update rule of AWR [16] does not provide good universality [5], namely two distinct output values might yield the same gradient under the MAML [4] framework. To address this universal update problem, MACAW introduced $\mathcal{L}_{ADV}$ which is defined as: $\mathcal{L}_{ADV}(\theta, \phi_i', \mathcal{D}_i)$ $= \mathbb{E}_{(s,a) \sim \mathcal{D}_i}\left[\left(A_\theta(s, a) - \left(R_{\mathcal{D}_i}(s, a) - V_{\phi_i'}(s)\right)\right)^2\right]$, where $A_\theta(s, a)$ is employed to estimate the advantage, with the advantage defined as the return minus the state value: $R_{\mathcal{D}_i}(s, a) - V_{\phi_i'}(s)$. The combined gradient of $\mathcal{L}_{AWR}$ and $\mathcal{L}_{ADV}$ ensures that distinct output values will result in different gradients while updating the policy.

FOCAL: FOCAL [10] is an actor-critic model combined with an inference model, which draws inspiration from PEARL [19] and BRAC [26]. While PEARL trains the inference network to reconstruct the MDP by learning a predictive model of reward and transition, FOCAL disentangles the learning of the context encoder from the learning of the control policy. The inference network q_ω (parameterized by ω) of FOCAL aims to cluster similar data and push away dissimilar data in the embedding space Z to obtain the embedding q_ω of the data buffer $\mathcal{D}$. The inference network was updated by minimizing the following loss function: $\mathcal{L}_{dml}(x_i, x_j, q_\omega) = 1\{y_i = y_j\}\|q_i - q_j\|_2^2 + 1\{y_i \neq y_j\}\hat{\eta} \cdot \frac{1}{\|q_i - q_j\|_2^2 + \epsilon}$,

where x_i is the input state of the task, y_i is the input task, and $q_i = q_\omega(x_i)$ is the embedding vector of x_i. $\hat{\eta}$ is the distant coefficient, and $\epsilon > 0$ is a small hyperparameter added to avoid division by zero. On the other hand, FOCAL introduces a value penalty mechanism from BRAC [26], aimed at ameliorating overestimation problems in Q-values in the offline setting. Consequently, the actor and critic losses are formulated as $\mathcal{L}_{critic} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_i a'\sim\pi_\theta(\cdot|s')}\left[\left(r + \gamma\bar{Q}_\phi(s', a', \bar{z}) - Q_\phi(s, a, \bar{z})\right)^2\right]$,$\mathcal{L}_{actor} = -\mathbb{E}_{(s,a,r,s')\sim\mathcal{D}_i}\left[\mathbb{E}_{a''\sim\pi_\theta(\cdot|s)}\left[Q_\phi(s, a'', \bar{z})\right] - \hat{\delta}D_{KL}\Big(\pi_\theta(\cdot|s), \pi_b(\cdot|s)\Big)\right]$. FOCAL align with the previous work [26], the actor-critic model learns initialization weights ϕ and θ for Q-function Q_ϕ and policy π_θ, and $\bar{Q}_\phi$ denotes a soft-updated target Q-function [11] without gradients. The latent context variable $\bar{z} \in Z$ is computed by inference network q_ω, which indicates that gradients are not being computed through it. In FOCAL, the complete information of z and s is combined as the full state. The parameter $\hat{\delta}$ is the adaptive regularization coefficient, with the term D_{KL} indicating the Kullback-Leibler divergence between the current policy π_θ and the behavior policy π_b. The KL divergence serves as a regularization term within the policy objective function, effectively mitigating overestimation issues present in the Q-values.