Energy-efficient LLM Training in GPU datacenters with Immersion Cooling Systems

Shuntao Zhu
The Hong Kong Polytechnic Univeristy
Kowloon, Hong Kong
shun-tao.zhu@connect.polyu.hk

Dan Wang
The Hong Kong Polytechnic University
Kowloon, Hong Kong
dan.wang@polyu.edu.hk

Abstract

With the increase in AI applications, the energy consumption of datacenters that run AI jobs is greatly increasing. The overall energy consumption of a datacenter is closely linked with that of its cooling system. Recently, there has been a revolution in immersion cooling technologies, in which servers can be directly immersed in dielectric cooling liquid (coolant). However, there is a lack of understanding of how the performance of AI jobs is affected by immersion cooling systems. While the physics behind immersion cooling is understood, in this paper we observe key restricting factors: (1) the boiling state of the coolant and (2) the heat removal rate of the coolant may not match the heat generation rate of the GPUs, triggering the thermal-throttle mechanisms of the GPUs.

In this paper, we study the energy-efficient and delay-ensured computing of large language model (LLM) training jobs over a cluster of GPUs in immersion cooling systems. We model the thermal characteristics of the system (e.g., heat generation, heat removal, and temperature) and develop an algorithm with workload assignment and frequency scaling to avoid the delay incurred by the thermal-throttle mechanisms and to execute the workloads in energy-efficient frequencies. In our evaluation, we simulate the computational fluid dynamics (CFD) of the immersion cooling systems through the Ansys Fluent software. We show that we outperform baseline algorithms by up to 53.2% in energy and 22.5% in delays.

CCS Concepts

• **Hardware** → **Power and energy**.

Keywords

Immersion Cooling, LLM Training, Thermal Control

ACM Reference Format:
Shuntao Zhu and Dan Wang. 2025. Energy-efficient LLM Training in GPU datacenters with Immersion Cooling Systems. In *The 16th ACM International Conference on Future and Sustainable Energy Systems (E-ENERGY '25), June 17–20, 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3679240.3734609

1 Introduction

With the surge in AI applications, the energy consumption of the GPU datacenters that support AI jobs is greatly increasing [18]. The overall energy consumption of a datacenter is closely linked with that of the cooling system. Typically, the servers consume 52.9% of the energy of a datacenter, while the cooling system consumes 40%, with a power usage effectiveness (PUE) of 1.89 [1, 24].

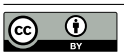
Recently, there has been a revolution in immersion cooling technologies, where servers can be directly immersed in the dielectric cooling liquid (coolant) (see Fig. 1). Immersion cooling systems have a much greater coefficient of performance (COP), i.e., by injecting a certain amount of electricity, the overall heat removal capacity is much greater. Intrinsically, traditional cooling systems need extra electricity to cool the circulating air, yet this is no longer needed in immersion cooling systems. The circulating coolant can absorb the heat of the servers and dissipate heat in an ambient environment through natural heat dissipation. Continuing reductions in the cost of coolant have recently caused immersion cooling systems to become economically viable. In Appendix A, we present background on immersion cooling systems. Studies have shown that the energy consumed by immersion cooling systems in the overall energy consumption of a datacenter can be as low as 3.6%-13.0%, achieving a PUE of 1.037-1.15 [12, 20].

Nevertheless, how immersion cooling systems may affect the performance of AI jobs is unknown. Specific questions include: For an AI job with certain workloads running on a cluster of GPUs, what restrictions will the immersion cooling system introduce? How can AI jobs be executed given such restrictions? This paper presents the very first work to answer these questions. The focus of this study is on LLM training jobs.

For the first question, we study the physics behind the immersion cooling systems and observe two basic restricting factors given a certain type of coolant: (1) the boiling state: a coolant has two states, a normal-working state, and a boiling state. In the boiling state, a vapor film forms over portions of the GPU surface and the heat removal rate substantially decreases and (2) the heat removal rate of the coolant may not catch up with the heat generation rate of a GPU. Temperatures will increase and this will trigger the thermal-throttle mechanisms of the GPUs when the GPUs are overheated.

For the second question, we study an *E*nergy-efficient *L*LM training in *I*mmersion *C*ooling (ELIC) problem, where we compute an LLM training job with a delay requirement on a GPU cluster with diverse types of GPUs in an immersion cooling system with a certain type of coolant. In computing this LLM job, we minimize the overall energy of the GPU cluster.[1] We capture the thermal characteristics of the datacenter system using a *coolant state model* and a *system temperature model* with a *heat generation model* and a *heat removal model*. We develop an ELIC algorithm to output the *workload assignment* of the LLM job workloads and the *frequency scaling* of each

[1]We omit the energy consumed by the immersion cooling system since it is a small amount and doing so simplifies the study. Additional discussions are in §5.



This work is licensed under a Creative Commons Attribution 4.0 International License.
E-ENERGY '25, Rotterdam, Netherlands
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1125-1/25/06
https://doi.org/10.1145/3679240.3734609

GPU, which are two common techniques to save datacenter energy [10, 25]. ELIC runs the LLM job at energy-efficient frequency levels and avoids triggering the boiling state of the coolant and the thermal-throttle mechanisms of the GPUs, which introduce delays.

We evaluate our algorithm using simulations. We adopt computational fluid dynamics (CFD) simulations using the Ansys Fluent software to simulate the thermal environment. We evaluate typical LLM training jobs (e.g., LLaMA 3-8B [3] and Phi-2 [11]) on regular clusters of 8 GPUs of NVIDIA H100, A100, and L40S, with immersion cooling systems with standard coolants HFE-7100 (Novec 7100), FC-72, and Ethanol. We observe that without the thermal characteristics models on immersion cooling systems, a coolant agnostic energy-first and a coolant agnostic delay-first algorithm can both end up with long delays that cannot meet the delay requirements. Even with an appropriately designed cooling-aware baseline algorithm, ELIC shows an energy reduction of 53.2% and a delay reduction of 22.5%. ELIC can effectively avoid the delay incurred by the thermal-throttle mechanisms and execute the workloads in energy-efficient frequencies. We also observe that for certain types of coolant, high-end GPUs have to run at a low performance irrespective of the algorithm. This is due to the great heat generation rates of the GPUs. We present solutions in our discussions §5.

2 Modeling the Thermal Characteristics of the System

We present the general background of the immersion cooling system in Appendix A. The coolant has two states: a *normal-working state* and a *boiling state*. In §2.1, we present the cooling state model that specifies the conditions for the GPUs to trigger the coolant stage change given certain types of GPUs and certain coolants.

In LLM job computing, we need to manage GPU workloads and frequencies, and thus temperatures, to avoid triggering GPU overheating. The GPU temperature depends on the heat generation rate and heat removal rate. In §2.2, we present a GPU temperature model with a heat generation model and a heat removal model.

2.1 The Coolant State Model

The thermal property of the coolant in the boiling state differs from that in the normal-working state. In the boiling state, the heat removal rate decreases significantly. For example, the heat removal rate in the normal-working state can reach 330.4W for a GPU A100 with Die Size 826mm^2; but drops to 41.3W in the boiling state [9].

The intrinsic trigger of the change in the coolant state is the power per unit area that impacts the coolant. It is called the heat flux of a coolant (W/m^2). The maximum heat flux that a coolant allows is called the critical heat flux (CHF). Each coolant has a specific CHF. Note that this is independent of the temperature. It is the unit power that triggers the change in state of the coolant. As an example, for coolant HEF7100, the CHF is 45.1W/cm^2 [6]. CHF, q_{max}, can be estimated using Zuber's CHF model [17], $q_{max} = 0.131\rho_v h_{fg}[\frac{\sigma(\rho_l-\rho_v)g}{\rho_v^2}]^{1/4}$. We explain q_{max} in detail in Appendix B. Given a coolant, q_{max} can be seen as a constant.

Given a datacenter with an immersion cooling system, the GPUs and the coolant are determined. Thus, the die sizes of the GPUs A_{GPU} and the CHF q_{max} of the coolant are constants. We can calculate the power of the GPU to trigger the boiling state. This is the *boiling limit*, $\dot{Q}_{max}$ of the GPU:

$$\dot{Q}_{max}(\mathrm{W}) = q_{max}(\mathrm{W/m^2}) \times A_{GPU}(\mathrm{m^2}) \tag{1}$$

If a GPU runs at a power that is greater than the boiling limit, the coolant enters the boiling state.

2.2 The GPU Temperature Model

The GPU Temperature Model: We study the GPU temperature in the normal-working state. The temperature of a GPU i at time t, $T_i(t)$, is dynamic. It can be captured using Eq. 2.

$$T_i(t) = T_i(t-\Delta t) + \frac{(\dot{Q}_G - \dot{Q}_R) \times \Delta t}{mC_i} \tag{2}$$

$T_i(t)$ depends on the prior temperature $T_i(t-\Delta t)$, the heat generation rate $\dot{Q}_G$ (W), and the heat removal rate $\dot{Q}_R$ (W) in this period (i.e., the total amount of heat removed from the substance), subject to the heat capacity of this substance, i.e., the increase in the temperature of this substance given the injection of a certain amount of heat. Here, m is the mass of the heat source (i.e., a GPU) and C_i is the heat capacity of the heat source. We next derive the heat removal rate $\dot{Q}_R$ and the heat generation rate $\dot{Q}_G$.

The Heat Removal Model of Immersion Cooling Systems: $\dot{Q}_R$ changes dynamically. Specifically, $\dot{Q}_R$ depends on the intrinsic property of the coolant, the heat flux q (W/m^2), and the die size of the GPU A_{GPU}. Intuitively, q reflects the fact that the higher the temperature difference between the heat source and the coolant, the higher the heat removal rate. This is reflected in Eq. 3.

$$q = \begin{cases} \mu_l \cdot h_{fg}\left[\frac{C_L(T_i-T_L)}{C_{sf}h_{fg}\mathrm{Pr}_L^{1.7}}\right]^3\left[\frac{g(\rho_L-\rho_v)}{\sigma}\right]^{1/2} & \text{if } T_i \geq T_{ONB} \\ h_{cv}(T_i - T_L) & \text{if } T_i < T_{ONB} \end{cases} \tag{3}$$

Here T_i and T_L are the temperature of the heat source (GPU i) and the liquid coolant; and (T_i - T_L) is the difference. There is a temperature of Onset of Nucleate Boiling T_{ONB}, and q is described by a piecewise function where if $T_i \geq T_{ONB}$, it is described by the Rohsenow's Correlation [29]; otherwise, it is described by the natural convection [2]. Other terms are constants. A description of the terms is given in Appendix B. Finally, $\dot{Q_R}$ is

$$\dot{Q_R} = q \times A_{GPU} \tag{4}$$

The Heat Generation Model of GPUs: The heat generation rate ($\dot{Q_G}(i)$) of a GPU i is equal to the power consumption (P_i) of the GPU according to the 1st-law of thermodynamics [12], $\dot{Q_G}(i) = P_i(f)$. According to the AccelWattch power model [13], the power consumption $P_i(f)$ of a GPU is a function of its core frequency f, where $P_i(f)$ can be estimated as

$$\dot{Q_G}(i) = P_i(f) = \beta_i G_i f^3 + \tau_i f + P_i^c \tag{5}$$

Here G_i is the gate capacitance decided by the GPU. P_i^c is the constant power caused by peripheral components. β and τ depend on the specific LLM training job and the GPU type.

An LLM training job contains a series of foundational operations (GPU kernels) ($\{K\}$). This operation set can be analyzed from the computational graph of the foundation model [21]. The modern deep-learning framework provides efficient tools (i.e., Pytorch Profiler and Tensorflow Profiler) to derive this operation set.

Given the GPU(i) and the GPU kernel ($k \in \{K\}$), the power of this kernel can be described by β_i^k and τ_i^k. For example, *Softmax* is a kernel and *General Matrix Multiplications* (GEMM) is also a kernel. In practice, β_i^k and τ_i^k are constants and can be profiled. For example, the GEMM kernel by a GPU A100 SXM4 with frequency 1410Mhz will consume 385W, and the profiles of $\beta_{A100}^{GEMM} = 4.38$ and $\tau_{A100}^{GEMM} = 2.18$. Given the set of kernels $\{K\}$ and the number of each kernel N_k, $k \in \{K\}$, the β_i and τ_i can be calculated by Eqs.:

$$\beta_i = \sum_{k \in \{K\}} N_k \times \beta_i^k, \quad \tau_i = \sum_{k \in \{K\}} N_k \times \tau_i^k. \tag{6}$$

3 The Problem and Algorithm

The Energy-efficient LLM Training in Immersion Cooling (ELIC) Problem: Given a datacenter with $\mathcal{N}$ GPUs, where each has a set of frequencies $\mathcal{F}$; the thermal characteristics of the datacenter system follow Eq. 1 and Eq. 2; and an LLM training job specified by an LLM model $\mathcal{M}$ with $\mathcal{W}$ layers and a delay requirement $\mathcal{D}$, determine a workload assignment scheme to assign each GPU i a set of model layers W_i, and a frequency scaling scheme to run each GPU i at a certain frequency f_i; so as to minimize the energy consumption of the LLM training job.

In practice, the LLM training job has a scheduler for workload assignments to parallel GPUs, and the operating system can perform frequency scaling. We leave it to a future work to develop a system to control workload assignments and frequency scaling, while focusing on the algorithmic aspects in this paper.

This problem has an intrinsic knapsack structure and is NP-hard. We omit the proof for the sake of conciseness. We develop a heuristic algorithm to solve this problem. In addition to the model of the thermal characteristics of the system, we need to model the performance (delay) of a certain amount of workloads of an LLM job given a specific GPU.

The performance of an LLM job on a GPU: We follow the model from *LLMCarbon* [7]. Given the number of floating point operations that the LLM model layer w requires ($FLOP_w$); the set of layers assigned to GPU i (W_i); and the frequency f_i of GPU i to execute W_i, the computing time t_i is determined using Eq. 7 and 8

$$t_i(f_i) = \sum_{w \in W_i} \frac{FLOP_w}{(FLOP(f_i) \cdot eff_{i,w})} \tag{7}$$

$$FLOP(f_i) = N_{core}^i \cdot f_i \cdot 2 \tag{8}$$

Here $eff_{i,w}$ is the hardware efficiency for a given GPU i and an LLM model layer w; and N_{core}^i is the core number for the given GPU i. Both of those parameters can be profiled.

The ELIC Algorithm: We propose a greedy-based algorithm. The boiling state and the thermal throttle mechanism can substantially reduce the performance of the computing of the LLM job. We call these the coolant constraints (i.e., Eq. 1 and Eq. 2). We search for the workloads W_i assigned to each GPU i and the frequency f_i of each GPU i. ELIC has three steps. The rationale is to first assign the workloads in a balanced manner (Step 1 and Step 2) and then iteratively adjust the workload across different GPUs based on their energy efficiency (Step 3). The pseudo-code is given in Appendix C.

Step 1 (lines 2-8): We start with the maximum frequency of each GPU and assign workloads to allow GPUs to finish their workloads simultaneously (using Eq. 7), subject to the coolant constraints.

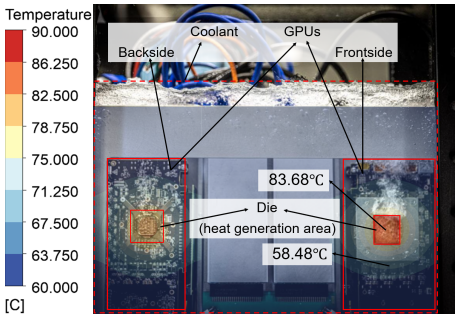


Figure 1: Visualization of the thermal environment

Step 2 (lines 10-12): For each GPU, we decrease its frequency to a scale that ensures its workloads are completed by the delay constraint $\mathcal{D}$. The energy is minimized for each GPU.

Step 3 (lines 14-25): We iteratively adjust the workloads among the GPUs, since the energy efficiency across GPUs is not balanced. In each iteration, we find the most energy-efficient GPU i and the least energy-efficient GPU j. The energy efficiency is the FLOPs processed per unit of energy. We adjust the workloads from GPU j to GPU i, subject to coolant constraints. The iteration is repeated until there is no further reduction in energy consumption.

4 Simulation and Evaluation

4.1 Simulation Setup

LLM Training Job: The LLM training jobs include 1) pre-training *LLaMA3-8B* [3] on dataset *RedPajama* [28] for 5 epochs; 2) fine-tuning 10% of parameters in *Gemma-7B* [26] on *OpenMathInstruct-1* [27] for 10 epochs; 3) fine-tuning 20% of parameters in *Phi-2* [11] on *no_robots* [22] for 25 epochs. The GPU types are L40S, H100, and A100. The other details are shown in Appendix D

Computational fluid dynamics (CFD) simulation: We need to simulate the process of heat exchange, a CFD process. In other words, when our system generates heat, this simulated environment will facilitate heat exchanges until the heat is removed and the temperature dynamics are maintained. As such, we use the software Ansys Fluent to conduct the simulation. Note that the CFD simulation utilized the built-in models in the Ansys Fluent. The models listed in this paper are used to guide the algorithm. For example, with the power of the GPU, the software can simulate the core temperature of the GPU to be 83.68°C, and the periphery with a certain distance of 3.47 cm to be 58.48°C (see Fig. 1).

4.2 Evaluation

In this section, we evaluate the overall performance of ELIC under different cases, coolants, and training jobs on three baselines. We list the performance on different quantities of GPUs, and task delay requirements in Appendix G.

4.2.1 Evaluation Methodology. We evaluate the ELIC under three commonly used coolants in the immersion cooling system: HEF-7100 [5], FC-72 [6], and Ethanol [8], the thermodynamic characteristics of which are listed in Table 2 of Appendix E. The delay requirement is set to 40% longer than the ideal minimum delay (i.e., each GPU works at Boost Frequency and the cooling system can efficiently remove heat without limitations.)

Baseline. We compare ELIC with three baselines: 1.) Delay First (***DF***): the frequency of each GPU is scaled first where the heat generation rate equals the boiling limit; the workloads are then assigned

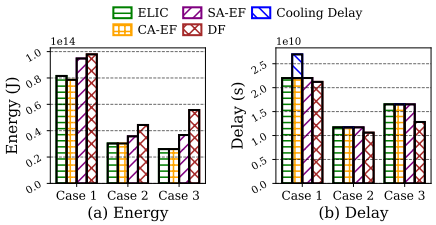


Figure 2: Performance of various cases with coolant FC-72 during pretraining.

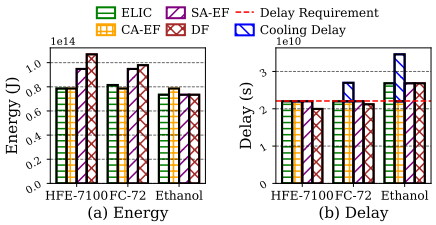


Figure 3: Performance of various coolants on Case 1 during pretraining.

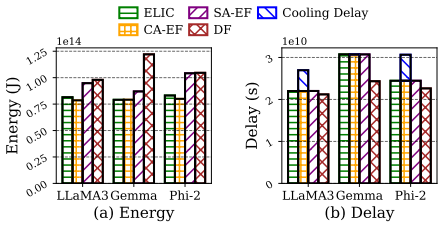


Figure 4: Performance of various training jobs on Case 1 and FC-72.

based on the throughput at such a frequency. 2.) Scheduler Agnostic Energy First (***SA-EF***): the workloads are assigned first to achieve the maximum throughput; the frequency is then scaled based on the cooling constraint and delay constraint. 3.) Coolant Agnostic Energy First (***CA-EF***) aggressively migrates layers to more energy-efficient GPUs without considering cooling constraints, resulting in overheating and triggering the thermal throttle mechanism.

Metrics. i.) Energy consumption of the computing system (J); ii.) Delay in the training job (s).

4.2.2 *Evaluation Results.*

ELIC under different Cases: Fig. 2 presents the energy consumption and delay in different cases on the FC-72 coolant. We first take Case 3 as an example. We observe that SA-EF and DF spend $3.67e13$ J and $5.57e13$ J of energy when ELIC is $2.60e13$ J, saving 53.2% and 29.1%, respectively. This is because ELIC has a lower frequency than DF and more proper workload assignments based on GPU energy efficiency compared to SA-EF. Moreover, ELIC and CA-EF have similar energy consumption levels, as they did not exceed the cooling constraint. From the aspect of delay, all baselines maintain a similar delay, approximating the delay requirement of about $1.65e10$ s, except for DF, the one with the lowest delay, since it runs at the highest frequency, resulting in a $1.28e10$ s delay. In another case, Case 1, we observe that CA-EF can save 3.62% of energy compared to ELIC; however, to save this 3.62% of energy, CA-EF aggressively assigns more workloads to a single GPU but increases the executing delay. To maintain the delay requirement, CA-EF has to run so that it exceeds the cooling constraint, i.e., overheating, requiring 22.5% more time to cool down. In contrast, ELIC detects the cooling constraint and abandons this assignment. The results show that ELIC is the most energy-efficient in different server configurations under an immersion cooling system.

ELIC under different Coolants: Fig. 3 presents the energy consumption and delay under different coolants on case 1. When using Ethanol, we observe that the ELIC, SA-EF, and DF have similar levels of energy and delay, that is $7.34e13$ J energy and $2.68e10$ s delay. This is because even the DF, the fastest approach, already exceeds the delay requirement, meaning that there is no space for energy optimization. Therefore, among the three coolants, Ethanol is the least suitable for GPU servers since all approaches exceed the delay requirement. This is because Ethanol's CHF q_{max} is only 23.48 W/cm^2 [16], resulting in the boiling constraint of GPU H100 being 191.13W, 27.3% of the maximum power of 700W. In contrast, HFE-7100 and FC-72 can better support the GPU servers. This is because the CHF of HFE-7100 and FC-72 are 45.1 W/cm^2 [5] and 39 W/cm^2 [6], respectively. The higher the CHF of the coolants, the greater the energy savings of the ELIC. The results show that under a suitable coolant, ELIC can save more energy when using a coolant with a higher CHF, also outperforming when compared with other baselines, and guaranteeing the delay requirements.

ELIC under different training jobs: Fig 4 presents the energy consumption and delay under different training jobs on Case 1 using the FC-72 coolant. The results are normalized since there is a huge gap in workload between the fine-tuning job and the pertaining job. From Phi-2, we observe that the ELIC spends $8.32e9$ J of energy, while SA-EF spends $1.04e10$ J of energy. The ELIC can save 20.2% more energy than SA-EF in fine-tuning the Phi-2 model. The ELIC saves more energy in fine-tuning than in pre-training (14% for LLaMA 3 pre-training). This is because, during fine-tuning, the computation of each epoch is dynamic for each LLM layer, making the workload hard to schedule. ELIC calculates the FLOPs of each layer and assigns the layer with the maximum workloads to an energy-efficient GPU to save energy, which can adapt to the dynamic workload of each layer. The results show that ELIC can handle dynamic workloads and outperform other baselines in energy reduction under different training jobs.

5 Conclusion and Discussions

Immersion cooling systems represent a revolution in cooling systems for datacenters, as they can substantially reduce the energy of cooling systems. Existing studies on immersion cooling systems have focused on how to improve immersion cooling technologies, e.g., materials, the phase changes of the coolant, costs, and performance measurements. This paper presented the first work on how to perform energy-efficiency LLM training *given* immersion cooling systems. We developed models to capture the thermal characteristics of immersion cooling systems. We showed that such models are critical in providing key information to develop algorithms to perform energy-efficient delay-ensured AI training.

This is a pioneer study, and many gaps remain to be filled. In this paper, we studied both the AI jobs and immersion cooling systems in their standard forms. AI workloads have unique patterns that can be leveraged, and inference jobs differ from training jobs. An immersion cooling system can increase the pressure on the coolant or speed up the circulation of coolant to prevent localized boiling near the heat source. These operations can change the boiling limit and the characteristics of the heat removal rate. Yet, they raise the energy consumption of the immersion cooling systems. New thermal models and new (joint)-optimization processes need to be developed to achieve high-performance and energy-efficient computing for AI jobs in immersion cooling systems.

Acknowledgments

Dan Wang's work is supported in part by RGC GRF 15200321, 15201322, 15230624, ITC ITF-ITS/056/22MX, ITS/052/23MX, and PolyU 1-CDKK, G-SAC8.

References

[1] Zhiwei Cao, Ruihang Wang, Xin Zhou, and Yonggang Wen. 2023. Toward Model-Assisted Safe Reinforcement Learning for Data Center Cooling Control: A Lyapunov-based Approach. In *Proceedings of the 14th ACM International Conference on Future Energy Systems*. 333–346.

[2] Pin Chen, Souad Harmand, and Safouene Ouenzerfi. 2020. Immersion cooling effect of dielectric liquid and self-rewetting fluid on smooth and porous surface. *Applied Thermal Engineering* 180 (2020), 115862.

[3] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783* (2024).

[4] Eaton. 2024. AC Unit for Server Racks - Rack Mount, 7,000 BTU (2.0 kW), 120V, 8U. https://assets.tripplite.com/product-pdfs/en/srcool7krm.pdf. Accessed: 12/17/2024.

[5] Mohamed S El-Genk and Jack L Parker. 2005. Enhanced boiling of HFE-7100 dielectric liquid on porous graphite. *Energy Conversion and Management* 46, 15-16 (2005), 2455–2481.

[6] Mohamed S El-Genk and Jack L Parker. 2008. Nucleate boiling of FC-72 and HFE-7100 on porous graphite at different orientations and liquid subcooling. *Energy conversion and management* 49, 4 (2008), 733–750.

[7] Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Chukwunyere Osi, Prateek Sharma, Fan Chen, and Lei Jiang. [n. d.]. LLMCarbon: Modeling the End-to-End Carbon Footprint of Large Language Models. In *The Twelfth International Conference on Learning Representations*.

[8] BR Fu, MS Tsou, and Chin Pan. 2012. Boiling heat transfer and critical heat flux of ethanol–water mixtures flowing through a diverging microchannel with artificial cavities. *International Journal of Heat and Mass Transfer* 55, 5-6 (2012), 1807–1814.

[9] S. Mostafa Ghiaasiaan. 2018. *Transition and Film Boiling*. Springer International Publishing, Cham, 1695–1746. https://doi.org/10.1007/978-3-319-26695-4_42

[10] Rohit Gupta, Douglas G Down, and Ishwar K Puri. 2021. Energy and Exergy-Aware Workload Assignment for Air-Cooled Data Centers. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*. 437–442.

[11] Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog* 1, 3 (2023), 3.

[12] Baris Burak Kanbur, Chenlong Wu, Simiao Fan, Wei Tong, and Fei Duan. 2020. Two-phase liquid-immersion data center cooling system: Experimental performance and thermoeconomic analysis. *International Journal of Refrigeration* 118 (2020), 290–301.

[13] Vijay Kandiah, Scott Peverelle, Mahmoud Khairy, Junrui Pan, Amogh Manjunath, Timothy G Rogers, Tor M Aamodt, and Nikos Hardavellas. 2021. AccelWattch: A power modeling framework for modern GPUs. In *Proc. of the IEEE/ACM International Symposium on Microarchitecture (MICRO'21)*. Virtual Event.

[14] Brian Edward Launder and Dudley Brian Spalding. 1983. The numerical computation of turbulent flows. In *Numerical prediction of flow, heat transfer, turbulence and combustion*. Elsevier, 96–116.

[15] Wen Ho Lee. 1980. A pressure iteration scheme for two-phase flow modeling. *Multiphase transport fundamentals, reactor safety, applications* 1 (1980), 407–431.

[16] Xingping Li, Lucang Lv, Xinyue Wang, and Ji Li. 2021. Transient thermodynamic response and boiling heat transfer limit of dielectric liquids in a two-phase closed direct immersion cooling system. *Thermal Science and Engineering Progress* 25 (2021), 100986.

[17] Gangtao Liang and Issam Mudawar. 2018. Pool boiling critical heat flux (CHF)–Part 1: Review of mechanisms, models, and correlations. *International Journal of Heat and Mass Transfer* 117 (2018), 1352–1367.

[18] Liuzixuan Lin, Rajini Wijayawardana, Varsha Rao, Hai Nguyen, Emmanuel Wedan GNIBGA, and Andrew A Chien. 2024. Exploding AI Power Use: an Opportunity to Rethink Grid Planning and Management. In *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*. 434–441.

[19] LiquidStack. 2022. DataTank 4U, Up to 6kW. https://liquidstack.com/content/uploads/2022/06/DataTank4U-Data-Sheet.pdf. Accessed: 2025-01-11.

[20] Cheng Liu and Hang Yu. 2021. Evaluation and optimization of a two-phase liquid-immersion cooling system for data centers. *Energies* 14, 5 (2021), 1395.

[21] Nvidia. 2020. Kernel Profiling Guide. https://docs.nvidia.com/nsight-compute/2020.1/pdf/ProfilingGuide.pdf. Accessed: 2025-01-11.

[22] Nazneen Rajani, Lewis Tunstall, Edward Beeching, Nathan Lambert, Alexander M. Rush, and Thomas Wolf. 2023. No Robots. https://huggingface.co/datasets/HuggingFaceH4/no_robots.

[23] Amazon Web Services. 2024. Instance Types-Amazon EC2. https://docs.aws.amazon.com/pdfs/ec2/latest/instancetypes/ec2-types.pdf#ec2-instance-type-specifications Accessed: 2025-01-11.

[24] Arman Shehabi, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner. 2016. United states data center energy usage report. (2016).

[25] Zhenheng Tang, Yuxin Wang, Qiang Wang, and Xiaowen Chu. 2019. The impact of GPU DVFS on the energy and performance of deep learning: An empirical study. In *Proceedings of the Tenth ACM International Conference on Future Energy Systems*. 315–325.

[26] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295* (2024).

[27] Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024. OpenMathInstruct-1: A 1.8 Million Math Instruction Tuning Dataset. *arXiv preprint arXiv: Arxiv-2402.10176* (2024).

[28] Maurice Weber, Daniel Y Fu, Quentin Gregory Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, et al. 2024. RedPajama: an Open Dataset for Training Large Language Models. In *38th. Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

[29] James Welty, Gregory L Rorrer, and David G Foster. 2014. *Fundamentals of momentum, heat, and mass transfer*. John Wiley & Sons.

A Background on Immersion Cooling Systems

A two-phase immersion cooling system is shown in Fig. A.1. There are two cooling cycles. The first cooling cycle removes the heat from the immersed servers to the coolant; and from the coolant to the water in the condenser pipe. The second cooling cycle removes the heat from the water in the condenser pipe to the open atmosphere.

The physics for why an immersion cooling system is more energy-efficient than a traditional cooling system is that a traditional cooling system uses air to absorb heat, whereas an immersion cooling system uses coolant to absorb heat. Air has a lower capacity than coolant to absorb heat. Thus, in a traditional cooling system the air first needs to be chilled to a low temperature (e.g., from an ambient temperature of 23°C to 8°C) by a compressor, and this process requires a significant amount of energy. Coolant has a much greater capacity to absorb heat through phase change. Circulation of the coolant (relayed by water) to ambient air can remove the heat to the environment. A comprehensive experiment was conducted in [20] where 140 T2T CPUs were used with an immersion cooling system of coolant Novec 7100. The collective power of the CPUs ranged from 1127 W to 1577 W, yet the power of the immersion cooling system remained at 59 W. When a traditional air cooling system was used in a similar setting with the servers having a heat generation capacity of 2000 W, the power consumption of the traditional cooling system was 1020 W [4]. Continuing reductions in the cost of coolants has finally made immersion cooling systems economically viable.

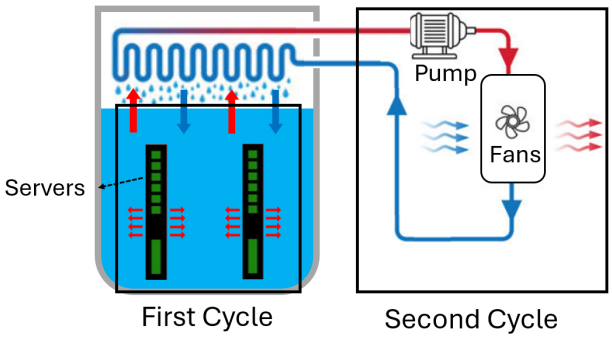


Figure A.1: A two-phase immersion cooling system

B Description of Coolant Models

Zuber's CHF Model: The critical heat flux (CHF) is the maximum heat flux that can be applied to the surface of the heat source. Beyond this point, a vapor film forms over portions of the surface due to the rapid generation of vapor. This prevents direct contact between the liquid coolant and the heated surface, resulting in a significant decrease in the heat removal rate. The CHF can be estimated using Zuber's CHF model [17], $q_{max} = 0.131\rho_v h_{fg}[\frac{\sigma(\rho_l-\rho_v)g}{\rho_v^2}]^{1/4}$, where $\rho_{\{l,v\}}$ is the density of the liquid and vapor coolant; h_{fg} is the latent heat of vaporization; and σ is the surface tension coefficient.
Heat Removal Model: When the heat flux is lower than the CHF, the coolant works at a normal-working state. Rohsenow's Correlation [29] and natural convection [2] can describe the relationship between the heat flux and the temperature, divided by the temperature of Onset of Nucleate Boiling T_{ONB}, as shown below:

$$q = \begin{cases} \mu_l \cdot h_{fg}\left[\frac{C_L(T_i-T_L)}{C_{sf}h_{fg}\mathrm{Pr}_L^{1.7}}\right]^3\left[\frac{g(\rho_L-\rho_v)}{\sigma}\right]^{1/2} & \text{if } T_i \geq T_{ONB} \\ h_{cv}(T_i - T_L) & \text{if } T_i < T_{ONB} \end{cases}$$

Here, μ_l is the chemical potential of the liquid coolant; h_{fg} is the latent heat of vaporization of the coolant; C_L is the heat capacity of the liquid coolant; C_{sf} is an empirical constant representing surface properties; Pr_L is the Prandtl number for a liquid, defined as the ratio of momentum diffusivity to thermal diffusivity; and σ is the surface tension coefficient. Those parameters in the CHF and Heat Removal models are thermodynamic constants that describe the physical properties of materials and these values are well-documented and stable. Table 2 lists some examples of those thermodynamic parameters under standard atmosphere.

C Algorithm Pseudo-codes

In this section, the pseudo-codes of the ELIC algorithm are presented as shown in Algorithm 1. Specifically, the input of the ELIC algorithm is the set of GPUs $\{i\}$, Coolant, Delay constraint $\mathcal{D}$, and LLM model $\mathcal{M}$. The output is the Workload assignment $\{W_i\}$ and frequency scaling $\{f_i\}$ of each GPU i. The algorithm works in three steps. In lines 2-8, we start with the maximum frequency of each GPU and assign workloads to allow all GPUs to finish simultaneously (using Eq. 7), subject to the constraints of coolant. In lines 10-12, for each GPU, we decrease its frequency to the scale that its workloads can be completed by the delay constraint $\mathcal{D}$. The energy is minimized for each GPU. In lines 14-25, we iteratively adjust the workloads among the GPUs since the energy efficiency across the GPUs is not balanced. In each iteration, we find the most energy-efficient GPU i and the worst energy-efficient GPU j. The energy efficiency is the FLOPs processed per unit of energy. We adjust the workloads from GPU j to GPU i subject to coolant constraints. The iteration is repeated until no further reductions occur in energy consumption.

Algorithm 1: ELIC Algorithm

Input: GPUs $\{i\}$, Coolant, Delay constraint $\mathcal{D}$, LLM model $\mathcal{M}$
Output: workloads $\{W_i\}$ and frequencies $\{f_i\}$

```
// Step 1: Initialize Frequencies and Workloads
foreach GPU i do
    Calculate Q_i^max by Eq. 1 and 2;
    Calculate f_i^max by Eq. 5 with Q_i^max;
    set f_i = f_i^max;
    Calculate t_i by Eq. 7;
for layer in M do
    Assign layer to GPU i with minimum t_i;
// Step 2: Minimize Energy for Each GPU
foreach GPU i do
    Calculate f_i^min by Eq. 7 with D;
    set f_i = f_i^min;
// Step 3: Balance Energy Efficiency Across GPUs
repeat
    foreach GPU i do
        Calculate power P_i by Eq. 5 with f_i;
        energy E_i = P_i · t_i;
        energy_eff_i = W_i / E_i;
    Identify GPU i with maximum energy_eff_i;
    Identify GPU j with minimum energy_eff_j;
    Move layer with maximum FLOPs from GPU j to GPU i;
    Scale f_i and f_j to minimize energy following line 11-12;
    if f_i > f_i^max OR f_j > f_j^max then
        Discard this assignment;
until no further reduction in energy consumption;
return {W_i} and {f_i}
```

D Datacenter Configurations

To support the LLM training jobs, we simulate three typical datacenter configurations. Each configuration contains two instances from AWS EC2 [23] and each instance is equipped with 8 GPUs. Those datacenter configurations use three types of GPUs: L40S, H100, and A100. They all have a large GPU memory (> 40GB) suitable for LLM training. Details of the specifications are listed in Table 1.

Table 1: Server Configurations

Case	Specifications (GPUs)
Case 1	p5.48xlarge (8×NVIDIA H100) + p4d.24xlarge (8×NVIDIA A100)
Case 2	p5.48xlarge (8×NVIDIA H100) + g6e.48xlarge (8×NVIDIA L40S)
Case 3	p4d.24xlarge (8×NVIDIA A100) + g6e.48xlarge (8×NVIDIA L40S)

To support such datacenters, a typical immersion cooling system is created from liquidstack [19], with a tank of 880mm × 1584mm × 733mm (W × H × D) that can contain 800mm height coolant.

E Details of the Computational Fluid Dynamics Simulation

Computational Fluid Dynamics (CFD) simulations involve simulating and analyzing the immersion cooling system's heat transfer and boiling phenomenon. Ansys Fluent is a widely used and validated professional CFD simulation software that is used to simulate fluid flow, heat transfer, and related physical phenomena in complex systems.

In the CFD, Reynolds-Averaged Navier-Stokes (RANS) equations describe the mass and energy transfer between both phases (liquid

Table 2: Thermodynamic parameters of various coolants.

Properties	HFE-7100	FC-72	Ethanol
Saturation Temperature (°C)	61	56.4	78.24
Liquid density (kg/m^3)	1372	1602	736.44
Latent heat of vaporization (kJ/kg)	111.6	94.9	849.95
Liquid thermal conductivity (W/(m·K))	0.062	0.054	0.153
Liquid viscosity (Pa·s)	$3.61e-4$	$4.25e-4$	$4.41e-4$
Surface tension (N/m)	0.010	0.008	0.015

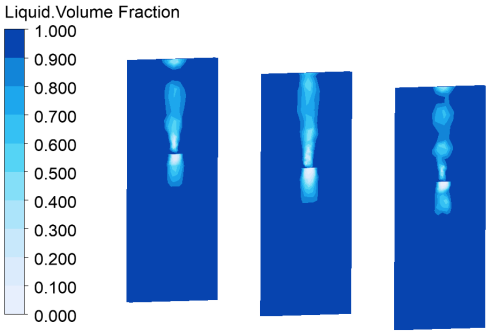


Figure E.1: Visualization of Boiling

and vapor). The RANS equations are shown in Eq. 9-11.

$$\frac{\partial}{\partial t}(\rho) + \sum_{j=1}^{3} \frac{\partial}{\partial x_i}(\rho \overline{u}_i) = S_M \tag{9}$$

$$\frac{\partial}{\partial t}(\rho \overline{u}_i) + \sum_{j=1}^{3} \frac{\partial}{\partial x_j} \cdot (\rho \overline{u}_i \overline{u}_j) = -\frac{\partial p}{\partial x_i} + \sum_{j=1}^{3} \frac{\partial}{\partial x_j}(-\rho \overline{u_i' u_j'}) + S_{F,i} + \sum_{j=1}^{3} \frac{\partial}{\partial x_j}[\mu(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i} - \frac{2}{3}\delta_{ij} \sum_{l=1}^{3} \frac{\partial \overline{u}_l}{\partial x_l}] \tag{10}$$

$$\frac{\partial}{\partial t}(\rho E) + \sum_{j=1}^{3} \frac{\partial}{\partial x_j}(\rho E \overline{u}_j) = \sum_{i=1}^{3} \sum_{j=1}^{3} (\frac{\partial}{\partial x_j}(\tau_{ij} - \rho \overline{u_i' u_j'}) \overline{u}_i) - \sum_{j=1}^{3} \frac{\partial}{\partial x_j} q_j + S_E \tag{11}$$

Eq. 9 describes the conservation of mass within the two-phase immersion cooling system, which is used to simulate the volume of fluids, where S_M is the source term in the mass conservation equation(kg/m^{3}s). Eq. 10 describes the conservation of the momentum to simulate the movement of the fluids, where S_F is the source term in the momentum conservation equation(kg/m^{2}s^2). Eq. 11 describes the conservation of energy to simulate the heat transfer of the fluids, where S_E is the source term in the energy conservation equation (J/m^{3}s).

Table 3: Settings of the CFD model

Setting Parameters	Settings/Options
Multiphase Model	Volumn of The Fluid (VOF)
Evaporation & Condensation Model	Lee's Two-Phase Flow Model
Viscous Model	Standard k-epsilon Model
Pressure-Velocity Coupling Scheme	SIMPLE
Spatial Gradient	Least Squares Cell Based
Spatial Pressure	Body Force Weighted
Spatial Momentum	QUICK
Spatial Volume Fraction	Geo-Reconstruct
Spatial Turbulent Kinetic Energy	QUICK
Spatial Turbulent Dissipation Rate	QUICK
Spatial Energy	QUICK

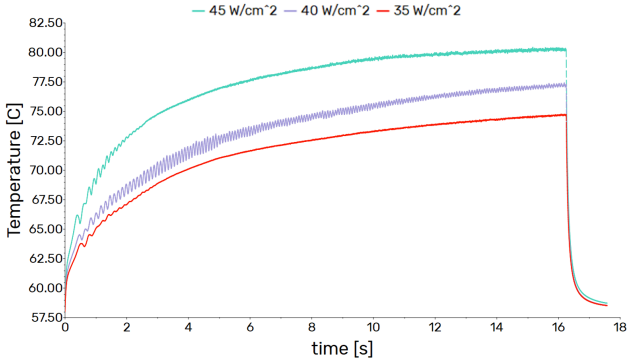


Figure E.2: Temperature with Threshold

In the Ansys Fluent, the volume of the fluid (VOF) model is used to simulate the boiling heat transfer in the tank. Lee's two-phase flow model [15] is used to simulate the evaporation and condensation of the coolant. The standard k-epsilon model [14] is used to simulate the viscosity. The method setting in the Ansys Fluent software is summarized in Table 3. The thermodynamic characteristics of coolants are listed in Table 2; these were used to define the liquid in the simulation.

We set the heat flux of 3 H100 GPUs to $\{45, 40, 35\}$W/cm^2 respectively. This setting for heat flux was chosen since the CHF of the HFE-7100 is 45.1 W/cm^2. Fig. E.1 demonstrates the simulated heat removal process of 3 H100. The vapor and liquid are distinguished by volume fraction. In addition, we add a temperature threshold to ensure that the temperature of the GPU is under the operation range to simulate the thermal-throttle mechanism of the GPUs. The temperature results are shown in Fig. E.2. The GPUs are shut down at 16.25 s. Hence the last curve of the temperature results shows the effect of cooling.

F Details of Profiling

We need to profile three sets of parameters: 1.) The GPU kernels $\{K\}$ executed by the LLM training job. 2.) The $(\beta_i^k, \tau_i^k, P_i^c)$ for each GPU kernel *oper*. 3.) The $\alpha_{i,w}$ for each GPU.

Methodology: We employ a uniform sampling method with a defined delay cutoff to filter out infeasible frequency configurations $\{f_i\}$ for each GPU i. Given the monotonic relationship between delay and frequency, we discard invalid frequencies and redistribute

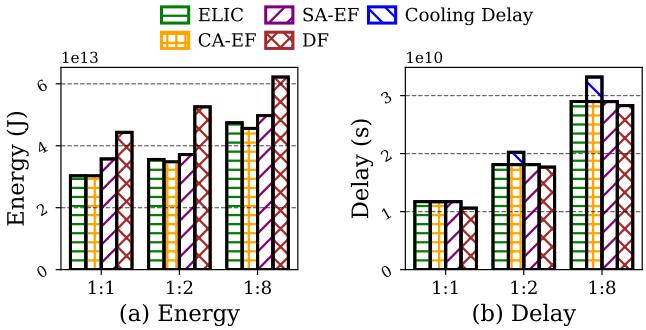


Figure G.1: Performance of GPU quantities with coolant FC-72 during pretraining

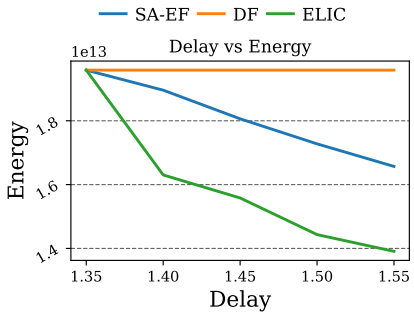


Figure G.2: Performance of Delay Requirements with coolant FC-72 during pretraining

saved samples, ensuring uniform exploration of valid configurations within the search space. Given the LLM training model $\mathcal{M}$, a server configuration with a set of GPU $\{i\}$, we execute the LLM job on the given GPU server with several iterations of each frequency in the sampled set $\{f_i\}$. The frequency can be scaled and the *Power Consumption* of the GPU can be measured using the NVIDIA Management Library (NVML), which is a C-based API for managing and monitoring various aspects of NVIDIA GPUs.

During execution, we monitor the sequence $\{K_{\mathcal{M}}\}$ and power consumption $\{P_i^k(f_i)\}$ of GPU kernels and the execution time $t_{i,w}(f_i)$ for layer w using NVIDIA Nsight Compute [21]. The FLOPs of the model $\mathcal{M}$ can be approximated by $FLOPs \approx 6N_P N_D$ [7], where N_P is the number of parameters of $\mathcal{M}$ and N_D is the number of tokens in the dataset. Then we can fit the parameters (β_i^k, τ_i^k) through the collected sample data $\{P_i^k(f_i)\}$ and $\{f_i\}$ using Eq. 5, and fit the parameters $\alpha_{i,w}$ through the collected sample data $t_{i,w}(f_i)$, and calculate $FLOPs$ using Eq. 7.

G Additional Evaluation

G.1 Impact of GPU quantities

Fig. G.1 presents the energy consumption and delay in different GPU quantities. We used three cases: (8xL40S & 8xH100), (4xL40S & 8xH100), and (1xL40S & 8xH100), according to the instance specifications of Amazon EC2 [23]. The x-tick name refers to the ratio L40S to H100. We observe that as the proportion of H100 increases, the overall energy consumption also increases. This is because L40S is more energy-efficient than H100. We also observe that as the proportion of H100 increases, the ELIC saved energy decreases compared with SA-EF (from 15.2% of 1:1 to 4.7% of 1:8). This is because as the proportion of H100 increases, the heterogeneity of the server decreases. Hence, the SA-EF can come close to the optimal energy consumption even if it is scheduler agnostic.

G.2 Impact of delay requirements

Fig. G.2 presents the energy consumption of SA-EF, DF, and ELIC with different delay requirements. The delay requirements are expressed as a multiple of the ideal minimum delay. We observed that the delay requirement must be more than 1.35 times that of the ideal minimum delay for FC-72, where SA-EF, DF, and ELIC consume similar amounts of energy. Otherwise, the LLM pretraining job cannot be accomplished. We also observed that ELIC saves 16.8% of energy as the delay requirement extends from 1.35 times to 1.4 times, compared with SA-EF, which only saves 3.2%. But when the delay requirement extends from 1.4 times to 1.45 times, ELIC only saves 4.4% of energy, compared with SA-EF which saves 4.7%. This is because, during the first period, ELIC saves energy from both workload assignment and frequency scaling. However, during the second period, the workload assignment is the same as that in the first period, which means that ELIC saves energy only from frequency scaling.

H Discussion and Future Work

This paper studies the restrictions of the immersion cooling system and transfers them to the power constraint of GPUs. Meanwhile, since we only changed the GPU *workload assignment* and *frequency scaling*, we ignored the changes in the non-GPU server energy. For the LLM training task, real systems may not always have explicit deadlines; in this paper, the deadline serves as an abstraction for performance constraints.

This work utilized simulation to evaluate the performance of the proposed models and algorithm. Although simulation provides a flexible and controlled environment for evaluating thermal behavior and energy dynamics, real systems can exhibit additional factors such as fluctuations in ambient temperature. But both frequency scaling and workload assignment are feasible in practice, and we plan to explore a system-level implementation in a future work.

Several potential techniques can be utilized in the immersion cooling system. Pressure management affects the density of the vapor coolant, while dynamic coolant circulation influences the liquidity of the liquid coolant. Both factors directly impact the critical heat flux (CHF) of the coolant and may increase the permissible GPU power constraint but also increase the energy consumption of the cooling system. We plan to investigate those techniques and the corresponding consequences in the future.

This paper is the first work on LLM training given an immersion cooling system. It focuses on average power consumption patterns and ignores the specific patterns of the LLM training task. In addition, this paper discusses only GPU clusters on a small scale. In this paper only the operational energy costs were considered, while the total costs of ownership (TCO), including the construction and maintenance costs, were not discussed. Those are the limitations of this paper and we will investigate those problems in a future work.