Joint Cooling and Computing Optimization for Language Model Serving

Nardos Belay Abera **Yize Chen**
Department of Electrical and Computer Engineering
University of Alberta
{nbelay, yize.chen}@ualberta.ca

Abstract

AI data centers are now deployed at a massive scale, supporting the deployment and serving of power-hungry large language models (LLMs). The sheer volume of both computing and cooling inside data centers raises growing concerns about LLM's energy and emission impacts. While the energy efficiency of LLM inference have been studied recently, most prior work focuses on compute-side scheduling and optimizations without explicitly accounting for thermal objectives or constraints. While intensive computing on GPUs can emit a lot of heat, which in turn affects data center performance, such an oversight can inadvertently increase the overall energy consumption or reduce the efficiency of LLM servers. To address this gap, we propose a joint modeling process for cooling and computing inside AI data centers and a novel hierarchical control framework that co-optimizes computing and thermal management by jointly tuning GPU parallelism, frequency (DVFS), and cooling control knobs. Using real Azure inference traces and detailed GPU profiling, our model balances serving latency with both energy efficiency and thermal requirements, achieving more than 10% in daily data center energy consumption.

1 Introduction

Recent surging use of LLMs has resulted in the deployment of massive inference clusters that handle millions of requests every day (1). These clusters often run on high-end GPUs with thermal design powers of several kilowatts (2). Although these GPUs provide the computational power required to support increasingly sophisticated models, they also consume large amount of electricity and place significant stress on datacenter cooling systems (3; 4). Consequently, growing energy footprint are significantly affecting both the environmental impact and operational costs of LLM services.

LLM inference engines do not consume power uniformly. Their power profiles are highly stochastic and are influenced by both computing and cooling loads. The hardware's operating conditions, such as the frequency of the GPU, the query service-level objectives (SLOs), and the degree of parallelism, determine the effectiveness of the energy used (5). Although the research community has made progress in improving the efficiency of LLM service through scheduling and optimization (6; 7; 8; 9), and some studies have examined LLM' computing power profiles (10), the role of temperature and cooling constraints remains underexplored (11). This is particularly important given that cooling can account for more than 40%

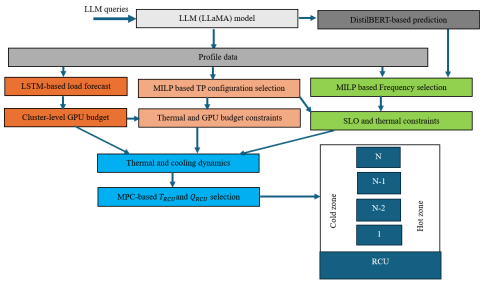

Figure 1: Block diagram of the proposed LLM inference computing and cooling control system.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

of total energy use in data centers (12), making it a critical component in improving AI data center efficiency.

To address this gap, this work develops a dynamic cooling-and-computing-aware management system for LLM inference clusters, which jointly optimizes cooling load, GPU parallelism, and frequency tuning in response to both service-level objectives (SLOs) and thermal constraints. A previous case study on LLM serving indicated that running `gpt2-large` can still meet the SLO (1000 ms) even when the GPU frequency is reduced from 1300 MHz to 800 MHz (13). By doing so, the power consumption was reduced by a factor of 1.8×, from 214 Watts to 120 Watts. We further extend such frequency scaling by proactively controlling the cooling load. By developing a data center cooling model (14), we formulate a joint optimal control problem that integrates both the thermal limits of GPU clusters and the thermal dynamics into the energy minimization framework for LLM serving sensitive to SLO (15). In contrast to techniques that rely on fixed computing and cooling resource allocations (16), our approach constantly reconfigures the system in response to changes in workload. Using a learning-based forecasting model for LLM traffic, the framework employs a hierarchical control scheme that includes slow-timescale optimization of tensor parallelism (TP) and fast-timescale model predictive control (MPC) of GPU frequency scaling and cooling supply.

2 Methodology

This section outlines the core components of our hierarchical cooling–computing control framework. Detailed mathematical derivations, complete thermal models, and optimization formulations are provided in Appendix A due to space constraints.

2.1 LLM Load Modeling

Workload Properties of LLM Inference: LLM serving workloads show dynamic bursty behavior, where user requests have different lengths. This variability leads to fluctuating GPU utilization, nonlinear latency patterns, and a wide range of thermal signatures. Context tokens dominated requests are computationally intensive but short-latency; whereas requests dominated by decoding lead to sustained load over a longer duration. These differences generate different latency profiles, so we classify requests based on their token lengths into three classes — short, medium, and long. This classification increases the accuracy of capacity estimation and control, since each request type has a different impact on throughput, power consumption, and temperature parameters under GPU frequency scaling.

Generating LLM profiles data: We start by profiling the LLM serving system under different hardware setups, measuring how throughput, power, and temperature change across different GPU TP configurations. By setting the GPU frequencies at different levels (1000–1800 MHz), we evaluate latency, computing energy use, and thermal behavior, grouped by request length (short, medium, long). Together, these measurements provide per-pool capacity, power, and performance curves that serve as the foundation for our control framework.

LSTM Workload Forecasting: We use an LSTM model trained on the Azure LLM Inference Trace to forecast aggregate token demand 30 minutes ahead, enabling proactive GPU allocation. The required number of GPU is given by $G_{\text{req}} = \left\lceil \frac{\hat{L}}{C_{\max}} \right\rceil$, where $\hat{L}$ is the predicted peak load and $C_{\max}$ is the maximum per-GPU capacity obtained from profile data.

2.2 Hierarchical Cooling and Computing Control

Building on the profile and LSTM forecast-derived GPU budgets, we design a three-level hierarchical controller: (i) window-level selection and allocation of TP pools, (ii) MPC-based cooling control, and (iii) GPU frequency tuning.

In the 5-minute control time-frame, we formulate a Mixed Integer linear program (MILP) to optimize the provision of GPUs and configurations of TP instances. This is achieved by utilizing the LSTM's forecasted load and profiled energy consumption data based on incoming queries' length and serving characteristics. In this optimization, we minimize overall energy consumption while ensuring that the predicted traffic $\hat{L}$ can be served by the provisioned TP instances. We solve this at a slow timescale,

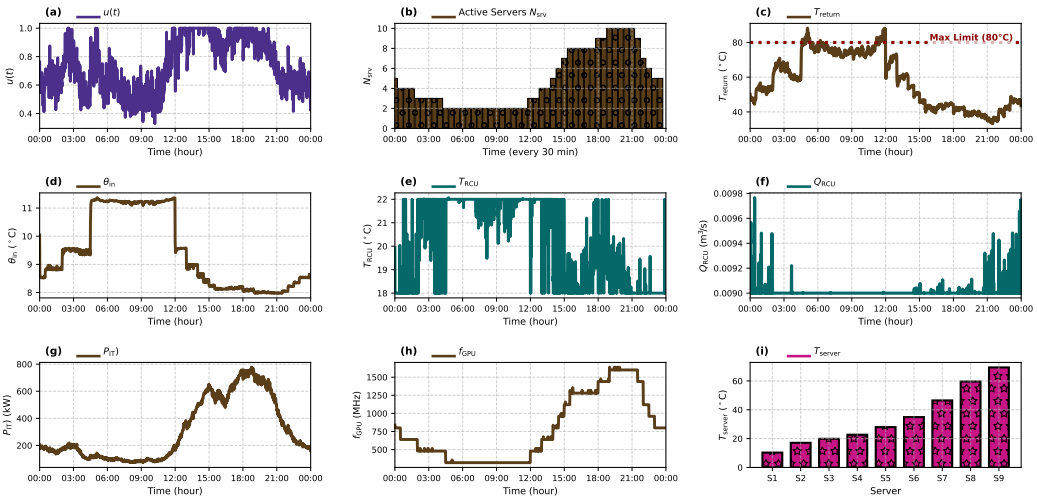

Figure 2: Joint MPC-controlled computing and cooling dynamics over a full day. (a) Normalized workload utilization $u(t)$; (b) number of active servers; (c) return-air temperature; (d) maximum inlet temperature across all servers; (e) RCU supply-air temperature θ_{HPCU}; (f) rack cooling airflow rate Q_{HPCU}; (g) total IT power P_{IT}; (h) mean GPU frequency across all GPUs; and (i) server temperature.

since GPU re-sharding usually takes longer and inference instances are not reconfigured frequently. Moreover, in upper-level optimization, we only give an upper bound for the cooling energy, which reduces the computational burden of deriving exact cooling actions. See Appendix A.3 in particular Equation (8) for more details.

In order to optimize TP instances, we further optimize the frequency of GPU with a MILP formulation that minimizes the total energy consumption while satisfying both the latency SLOs for each query type and the GPU temperature constraints. The MILP takes the characteristics of the LLM workload directly into account by applying an input of the token length of each LLM request to the DistilBERT-based classifier which predicts its class of job-length (short, medium, or long) and applies class-dependent latency limits to constraints. A detailed mathematical representation of this MILP, which gives us the objective function and the latency and thermal constraints, is presented in Appendix A.3; see in particular Equation (9). At the cooling level, we model fine-grained thermal dynamics for the data center and embed these into an MPC scheme with a prediction horizon NP = 2. The MPC objective is to minimize the total cooling energy subject to constraints on inlet temperature, airflow rate, supply temperature, GPU temperature, and return temperature. Finally, the controller also includes the predicted LLM workload, selected GPU frequency, and TP provisioning, explicitly allowing cooling setpoints to be proactively adjusted based on the computing workload treated as a disturbance. The detailed thermal model and MPC formulation are presented in Appendix A.3; specifically in Equation (10).

3 Evaluation

Setup: We evaluate our framework on a 1-day simulation with 30 min cluster-level planning and 5 min TP scheduling, capturing real LLM serving patterns from the *Azure LLM Inference Trace* (17). Optimization is solved using PuLP (18)(for upper-level server configuration) and SciPy SLSQP (for lower-level cooling control)(19). To validate our approach, we performed experiments on 8×Tesla V100 GPUs (16 GB) with the Llama-2 7B model inference workload, using peak 1-hour and trace 1-day. The controller is built on vLLM (7) for the TP configuration, and the results are compared with a baseline TP8 inference operating at the maximum GPU frequency.

Results: Fig. 2 shows the joint compute–cooling control on an entire day. Panel (a) shows the utilization $u(t)$ that follows the workload of the time-varying state. In panel (b), it is shown that the controller assigns additional active servers at busy times and reduces them back down during off peak ones. Consequently, the IT power in panel (g) increases with increasing workload (intensity) and increasing number of active servers, emphasizing the cost of the idle GPUs and thus the advantage of dynamic allocation. The controller uses MPC at the cooling layer to jointly optimize the flow rate

Q_{RCU} and the supply temperature T_{RCU} under thermal conditions. From panel (c), we see that the return temperature T_{return} continues to remain within the limit (80°C). Inlet temperatures θ_{in} (Panel (d)) remain within the limits. During light-load periods, MPC lowers Q_{RCU} and provides slightly higher T_{RCU} (Panel (e)–(f)) inlet temperatures, allowing them to increase by a small enough amount to save cooling energy. At high loads, Q_{RCU} rises and T_{RCU} falls, to improve heat removal, acting as the strongest actuator. Panel (h) illustrates how the mean GPU frequency varies with workload and SLOs: For loose latency requirements lower frequencies will be used, and for long jobs or tight workloads higher frequencies will be chosen. These daily decisions are aggregated to form a diurnal frequency profile that cuts power, but not at the expense of service quality. Panel (i) illustrates the temperature across servers, revealing a spatial gradient from rack-based cooling, where upstream servers experience warmer inlet air during cooling and more recirculation than those in proximity to the cooling source. In general, proposed hierarchical MPC adjusts server activation, GPU frequency, and cooling setpoints to track workload while maintaining thermal safety constraints, and MPC controller provision resources and energy more efficiently than static provisioning with performance.

The experimental results in Table 1 and Figure 3 show the relationship between power consumption, GPU temperature, and latency under the hierarchical control framework. Compared to the baseline, the proposed approach consistently reduces power consumption, demonstrating the effectiveness of workload-aware GPU frequency tuning and dynamic scheduling.

The hierarchical configuration also keeps temperatures stable and within safe operating limits, even with reduced GPU power draw. This indicates that the control algorithm lowers power while avoiding thermal violations. Latency performance remains close to the baseline, with only minor variations, and no significant overhead is observed. Therefore, Service-level objectives are maintained.

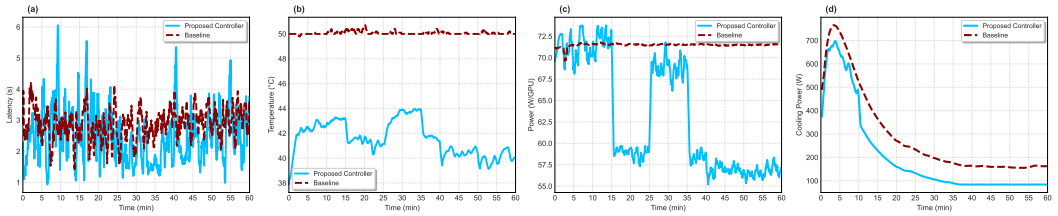

Figure 3: Overall evaluation of the proposed dynamic hierarchical control (blue) compared with a static baseline (red): (a) end-to-end latency and compute-side metrics, (b) temperature, (c) IT power and (d) cooling power profile.

Table 1 and Figure3 show that the hierarchical controller reduces the GPU power by nearly 12% and no additional latencies have been introduced. MILP-based TP selection and per-job frequency tuning jointly balance efficiency with reliability, sustaining safe operation at a lower cost. On the cooling side, Figure 3(a) shows that MPC achieves a 17.4% reduction in daily cooling power while respecting thermal limits. Combined, computing and cooling optimization produces about 29% total power savings per day, highlighting the effectiveness of workload-aware hierarchical control.

Table 1: Comparison of power consumption, temperature, and latency between Baseline and Controlled setup.

Metric	**Baseline**	**Proposed**	**Improvement**
Power (W/GPU)	71.50	63.00	**-11.9%**
Temperature (°C)	47.8	45.2	-2.6
Latency (s)	2.31	2.28	≈ 0

4 Conclusion and Future Work

In this work, we propose a hierarchical framework that jointly optimizes cooling and computing for LLM inference. Our approach explicitly considers thermal objectives and constraints by modeling the interactions between computing and cooling processes inside data centers. Simulation results on real LLM inference load and GPU clusters demonstrate a reduction of total energy by nearly 29% without SLO violations. The results highlight the importance of integrating thermal dynamics with compute scheduling to achieve sustainable AI data centers. Future work will explore proactive forecasting, larger-scale validation, and distributed optimization for real deployment.

References

[1] Y. Yao, H. Jin, A. D. Shah, S. Han, Z. Hu, Y. Ran, D. Stripelis, Z. Xu, S. Avestimehr, and C. He, "Scalellm: A resource-frugal llm serving framework by optimizing end-to-end efficiency," *arXiv preprint arXiv:2408.00008*, 2024.

[2] Y. Li, M. Mughees, Y. Chen, and Y. R. Li, "The unseen ai disruptions for power grids: Llm-induced transients," *arXiv preprint arXiv:2409.11416*, 2024.

[3] D. Min, I. Byun, G.-h. Lee, and J. Kim, "Cooldc: A cost-effective immersion-cooled datacenter with workload-aware temperature scaling," *ACM Transactions on Architecture and Code Optimization*, vol. 21, no. 3, pp. 1–27, 2024.

[4] Y. Li, Z. Hu, E. Choukse, R. Fonseca, G. E. Suh, and U. Gupta, "Ecoserve: Designing carbon-aware ai inference systems," *arXiv preprint arXiv:2502.05043*, 2025.

[5] J. Fernandez, C. Na, V. Tiwari, Y. Bisk, S. Luccioni, and E. Strubell, "Energy considerations of large language model inference and efficiency optimizations," *arXiv preprint arXiv:2504.17674*, 2025.

[6] T. Dao, D. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," *Advances in neural information processing systems*, vol. 35, pp. 16 344–16 359, 2022.

[7] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th symposium on operating systems principles*, 2023, pp. 611–626.

[8] X. Miao, C. Shi, J. Duan, X. Xi, D. Lin, B. Cui, and Z. Jia, "Spotserve: Serving generative large language models on preemptible instances," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2024, pp. 1112–1127.

[9] Y. Fu, S. Zhu, R. Su, A. Qiao, I. Stoica, and H. Zhang, "Efficient llm scheduling by learning to rank," *Advances in Neural Information Processing Systems*, vol. 37, pp. 59 006–59 029, 2024.

[10] J. Stojkovic, C. Zhang, Í. Goiri, E. Choukse, H. Qiu, R. Fonseca, J. Torrellas, and R. Bianchini, "Tapas: Thermal-and power-aware scheduling for llm inference in cloud platforms," in *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2025, pp. 1266–1281.

[11] J. Stojkovic, C. Zhang, Í. Goiri, J. Torrellas, and E. Choukse, "Dynamollm: Designing llm inference clusters for performance and energy efficiency," in *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2025, pp. 1348–1362.

[12] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 732–794, 2015.

[13] H. Qiu, W. Mao, A. Patke, S. Cui, S. Jha, C. Wang, H. Franke, Z. Kalbarczyk, T. Başar, and R. K. Iyer, "Power-aware deep learning model serving with $\{\mu$-Serve$\}$," in *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, 2024, pp. 75–93.

[14] C. Nadjahi, H. Louahlia, and S. Lemasson, "A review of thermal management and innovative cooling strategies for data center," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 14–28, 2018.

[15] X. Han, W. Tian, J. VanGilder, W. Zuo, and C. Faulkner, "An open source fast fluid dynamics model for data center thermal management," *Energy and Buildings*, vol. 230, p. 110599, 2021.

[16] A. H. Khalaj and S. K. Halgamuge, "A review on efficient thermal management of air-and liquid-cooled data centers: From chip to the cooling system," *Applied energy*, vol. 205, pp. 1165–1188, 2017.

[17] E. Choukse, P. Patel, C. Zhang, A. Shah, I. Goiri, S. Maleki, R. Fonseca, and R. Bianchini, "Splitwise: Efficient generative llm inference using phase splitting," *IEEE Micro*, 2025.

[18] Python PuLP Community, "Optimization with pulp," https://coin-or.github.io/pulp/, 2024, accessed: 2024.

[19] SciPy Community, "Scipy library," https://scipy.org/, 2024, online; accessed 2024.

A Appendix

A.1 LLM Serving

Parallelism. In practice, two parallelism strategies are commonly employed in LLM serving: pipeline parallelism (PP) and tensor parallelism (TP). With PP, different layers of the model are placed on separate GPUs, which reduces memory pressure but requires additional communication between stages. TP, on the other hand, splits the computations of a single layer across several GPUs. This method relies on high-bandwidth connections, but can achieve greater efficiency within a single server. The level of parallelism selected therefore has a direct bearing on system performance, power consumption, and thermal behavior.

LLM Serving Quality. When evaluating LLMs performance, several common metrics are typically used, such as throughput, time to first token (TTFT) and time between tokens (TBT). TTFT reflects the delay before the model produces its very first token, whereas TBT captures the average time required to generate each additional token. These latency measures are especially important for applications with strict SLOs, as they directly shape how responsive the system feels to users. However, in our work, we take a broader perspective. Instead of focusing only on TTFT or TBT, we measure the end-to-end latency of a request—starting from its arrival and continuing until the final output token is produced. This more comprehensive view provides a clearer sense of the quality of service experienced by users and allows us to better capture the trade-offs among energy consumption, performance, cost, and thermal constraints within LLM inference clusters.

A.2 Data Center Thermal and cooling dynamics.

We consider a rack-based air-cooled rack with N total servers. For each server $i = 1, \dots, N$, we monitor three temperatures: the cold-aisle inlet temperature $\theta_{c,i}(t)$, the server exhaust (core) temperature $\theta_{s,i}(t)$, and the hot-aisle temperature $\theta_{h,i}(t)$. The overall state vector is

$$\mathbf{z}(t) = \left[\theta_{c,1}, \theta_{s,1}, \theta_{h,1}, \dots, \theta_{c,N}, \theta_{s,N}, \theta_{h,N}\right]^\top.$$

The manipulated inputs from the rack cooling unit (RCU) are the setpoint of the air temperature supply $T_{\mathrm{RCU}}(t)$ and the airflow rate delivered $Q_{\mathrm{RCU}}(t)$:

$$\mathbf{u}(t) = \begin{bmatrix} T_{\mathrm{RCU}}(t) \\ Q_{\mathrm{RCU}}(t) \end{bmatrix}.$$

The temperature dynamics for cooling, exhaust, and hot-aisle are given by :

$$\dot{\theta}_{c,i}(t) = \frac{1}{V_c}\Big(\Phi_{h,i} T_{\mathrm{RCU}}(t) + \Phi_{i-1}^{\mathrm{OC}}\, \theta_{c,i-1}(t) + \Phi_L\, \theta_{h,i}(t) - \big(\Phi_i^{\mathrm{OC}} + \Phi_{s,i}\big)\theta_{c,i}(t)\Big), \tag{1}$$

$$\dot{\theta}_{s,i}(t) = \frac{\Phi_{s,i}}{C_{\mathrm{th}}}\big(\theta_{c,i}(t) - \theta_{s,i}(t)\big) + P_i(t), \tag{2}$$

$$\dot{\theta}_{h,i}(t) = \frac{1}{V_h}\Big(\Phi_{s,i}\, \theta_{s,i}(t) - \Phi_L\, \theta_{h,i}(t) + \Phi_{i-1}^{\mathrm{OH}}\, \theta_{h,i-1}(t) - \Phi_i^{\mathrm{OH}}\, \theta_{h,i}(t)\Big). \tag{3}$$

Here ρ is the density of the air, c_p is the specific heat of the air, V_c and V_h are the volumes of cold and hot zones, and C_{th} is the thermal capacitance of the server. The term $\Phi_{s,i}$ is the airflow from the internal server fan and $\Phi_{h,i}$ is the share of the airflow supplied by RCU passing through the server i. The airflow coupling parameters Φ_i^{OC} and Φ_{i-1}^{OC} describe mixing along the cold aisle; Φ_i^{OH} and Φ_{i-1}^{OH} capture hot air recirculation; and Φ_L represents leakage from the hot aisle back to the cold aisle.

The IT power of server i is modeled as

$$P_i(t) = \kappa_3 f_i(t) u_i(t) + \kappa_2 f_i(t) + \kappa_1 u_i(t) + \kappa_0,$$

where $f_i(t)$ is the GPU frequency, $u_i(t) \in [0, 1]$ is the utilization fraction and $\kappa_0, \kappa_1, \kappa_2, \kappa_3$ are the fit power-model coefficients. The average rack return temperature is

$$T_{\text{return}}(t) = \frac{1}{N} \sum_{i=1}^{N} \theta_{h,i}(t).$$

A simple affine surrogate for the server GPU temperature i is

$$T_{\text{GPU},i}(t) = \alpha\, \theta_{c,i}(t) + \beta\, P_i(t) + \gamma,$$

where α, β, γ are the identified coefficients, $\theta_{c,i}(t)$ is the temperature of the input (cold-aisle) and $P_i(t)$ is the power of the computer.

The cooling load extracted by the RCU is

$$Q_{\text{load}}(t) = \rho\, c_p\, Q_{\text{RCU}}(t) \left(T_{\text{return}}(t) - T_{\text{RCU}}(t)\right), \tag{4}$$

and the corresponding chiller power consumption is

$$P_{\text{cool}}(t) = \frac{Q_{\text{load}}(t)}{\eta(T_{\text{RCU}}(t))}, \tag{5}$$

where $\eta(\cdot)$ is the performance coefficient (COP) of the cooling plant, modeled as a quadratic function of the supply temperature:

$$\eta(T_{\text{RCU}}(t)) = a_2 T_{\text{RCU}}^2(t) + a_1 T_{\text{RCU}}(t) + a_0. \tag{6}$$

Finally, the power of the fan required to deliver airflow $Q_{\text{RCU}}(t)$ is approximated by a cubic:

$$P_{\text{fan}}(t) = \delta_0 Q_{\text{RCU}}(t) + \delta_1 Q_{\text{RCU}}^2(t) + \delta_2 Q_{\text{RCU}}^3(t), \tag{7}$$

where $\delta_0, \delta_1, \delta_2$ are the coefficients of the fan-model.

A.3 Hierarchical Control Formulation

TP selection at the Window-level. Given the token demand L_w in the window w and the GPU budget $C_{g,w}^{\max}$ decided by the cluster-level plan, we solve the following MILP to decide how many TP pools of each type to activate:

$$\begin{aligned}
\min_{\{y_{m,w} \in \mathbb{Z}_{\geq 0}\}} \quad & \sum_m \sum_t P_{m,w}\, y_{m,w} \\
\text{s.t.} \quad & \sum_m C_{m,w}\, y_{m,w} \;\geq\; L_w && \text{(coverage)} \\
& \sum_m m y_{m,w} \;\leq\; C_{g,w}^{\max} && \text{(GPU budget; multi-pool selection)} \\
& T_{m,w}^{\text{GPU}} \;\leq\; T_{\text{GPU}}^{\max}, \quad \forall m \in \mathcal{TP} && \text{(temperature cap).}
\end{aligned} \tag{8}$$

Here, $y_{m,w}$ denotes the number of TP type pools $m \in \mathcal{TP}$ activated in the window w. $C_{m,w}$ is the effective token-processing capacity per-pool and $P_{m,w}$ is the corresponding power consumption. The first constraint guaranties that the total capacity $\sum_m C_{m,w} y_{m,w}$ is sufficient to serve the token demand L_w. The second constraint enforces the GPU budget $C_{g,w}^{\max}$ coming from the cluster-level controller. The last constraint keeps the average GPU temperature $T_{m,w}^{\text{GPU}}$ of each pool below the safe limit $T_{\text{GPU}}^{\max}$. Thus, (8) selects a mix of TP pools that covers demand while minimizing power under GPU and thermal limits.

Per-job Frequency Selection. In the lower layer, for each job i in the window w, the GPU frequency is chosen from the set $\mathcal{F} = \{1000, \ldots, 1800\}$ MHz. The latency profile for the job class $c(i, w) \in \{\text{short, med, long}\}$ at frequency f is $L_{c(i,w),f}$.

$$
\begin{aligned}
\min_{\{x_{f,i,w}\}} \quad & \sum_{f \in \mathcal{F}} x_{f,i,w}\, p_{f,i,w} \\
\text{s.t.} \quad & \sum_{f \in \mathcal{F}} x_{f,i,w} = 1 && \text{(select one frequency)} \\
& \sum_{f \in \mathcal{F}} x_{f,i,w}\, L_{c(i,w),f} \;\le\; L^{\max}_{c(i,w)} && \text{(latency cap)} \\
& \alpha\theta^{\text{in}}_{i,w} \;+\; \beta \sum_{f \in \mathcal{F}} x_{f,i,w}\, p_{f,i,w} \;+\; \gamma \;\le\; T^{\max}_{GPU} && \text{(temperature cap)} \\
& x_{f,i,w} \in \{0, 1\}, \quad \forall f \in \mathcal{F}.
\end{aligned}
\tag{9}
$$

Here, $x_{f,i,w} \in 0, 1$ is a binary decision variable that equals 1 if the job i in window w is executed with frequency $f \in \mathcal{F}$ and 0 otherwise. The term $p_{f,i,w}$ denotes the power consumption of the GPU of job i at frequency f, and $c(i, w) \in$ short, med, long is the latency class of the job. For each pair of class frequencies, $L_{c(i,w),f}$ is latency and $L^{\max} c(i, w)$ is the corresponding latency deadline (SLO). The constant $T^{\max}_{GPU}$ is the safe temperature limit of the GPU. The objective of (9) minimizes the total power $\sum_{f \in \mathcal{F}} x_{f,i,w}, p_{f,i,w}$ for the job i. The first constraint requires that exactly one frequency be selected. The second constraint ensures that the latency of the chosen frequency does not exceed the class-specific bound $L^{\max}_{c(i,w)}$. The third constraint guaranties that the temperature of the GPU remains below $T^{\max}$.

MPC-based T_{RCU} and Q_{RCU} Control: presents an MPC formulation for cooling control in an AI data center. The controller jointly adjusts T_{RCU} and Q_{RCU} of the RCU on a prediction horizon to minimize energy use while ensuring thermal safety.

$$
\begin{aligned}
\min_{T_{\text{RCU}}(k),\, Q_{\text{RCU}}(k)} \quad & J = \sum_{k=0}^{N_p-1} \Big(P_{\text{cool}}(k) + P_{\text{fan}}(k) \Big) \\
\text{s.t.} \quad & \theta_{c,i}(k) \le \theta^{\max}_{c,i}, \quad \forall i,\ k = 0, \ldots, N_p - 1, \\
& T_{\text{return}}(k) \le T^{\max}_{\text{return}}, \quad k = 0, \ldots, N_p - 1, \\
& T_{\text{GPU},j}(k) \le T^{\max}_{\text{GPU}}, \quad j = 1, \ldots, n,\ k = 0, \ldots, N_p - 1, \\
& T^{\min}_{\text{RCU}} \le T_{\text{RCU}}(k) \le T^{\max}_{\text{RCU}}, \quad k = 0, \ldots, N_p - 1, \\
& Q^{\min}_{\text{RCU}} \le Q_{\text{RCU}}(k) \le Q^{\max}_{\text{RCU}}, \quad k = 0, \ldots, N_p - 1.
\end{aligned}
\tag{10}
$$

Here, the decision variables are the temperature of the RCU supply $T_{\text{RCU}}(k)$ and the airflow rate $Q_{\text{RCU}}(k)$. The objective J accumulates the total cooling power $P_{\text{cool}}(k)$ (chiller power) and the fan power $P_{\text{fan}}(k)$ on the horizon, so minimizing J corresponds to minimizing the total cooling energy. The term $\theta_{c,i}(k)$ denotes the temperature of the cold-zone i (e.g. the temperature of the inlet or the cold air temperature for the server i), and $\theta^{\max}_{c,i}$ is its upper limit allowed. $T_{\text{return}}(k)$ is the return-air temperature in the RCU, with an upper limit $T^{\max}_{\text{return}}$. The last two constraints enforce actuator limits on the RCU: the supply temperature must remain within $[T^{\min}_{\text{RCU}}, T^{\max}_{\text{RCU}}]$ and airflow within $[Q^{\min}_{\text{RCU}}, Q^{\max}_{\text{RCU}}]$.