TESLA: Thermally Safe, Load-Aware, and Energy-Efficient Cooling Control System for Data Centers

Hanfei Geng
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China
Department of Electronic Engineering, Tsinghua University
Beijing, China

Yi Sun
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China
Department of Electronic Engineering, Tsinghua University
Beijing, China

Yuanzhe Li
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China

Jichao Leng
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China

Xiangyu Zhu
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China

Xianyuan Zhan
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China
Shanghai Artificial Intelligence Laboratory
Shanghai, China

Yuanchun Li
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China
Shanghai Artificial Intelligence Laboratory
Shanghai, China

Feng Zhao
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China

Yunxin Liu*
Institute for AI Industry Research (AIR), Tsinghua University
Beijing, China
Shanghai Artificial Intelligence Laboratory
Shanghai, China

ABSTRACT

The increasing demand for artificial intelligence and cloud computing has led to skyrocketing energy consumption of data centers (DCs). This paper focuses on tackling this energy challenge through cooling control system optimization, which aims to ensure thermal safety with minimal cooling energy consumption. Current industry practice involves human operators, while many data-driven methods have also been proposed. However, human intervention often results in unnecessary energy consumption, particularly in the face of fluctuating server loads, whereas existing data-driven methods struggle to maintain thermal safety in practice. To overcome these issues, we propose TESLA, a thermally safe, load-aware, and energy-efficient cooling control system for data centers. TESLA employs a novel data-driven framework that integrates domain knowledge to predict DC temperature and cooling energy under dynamic server load. Based on these predictions, a Bayesian optimizer (BO) finds the energy-optimal settings for the cooling system at every control step. Besides cooling energy, BO's optimization objective also includes minimizing cooling interruption that causes rapid temperature rise within the data center and leads to thermal safety violations. We *deploy* TESLA on a real data-center testbed and show that it achieves on average 10.1% cooling energy saving relative to a fixed cooling system parameter setting and no thermal safety violation relative to previous data-driven methods.

CCS CONCEPTS

• **Applied computing** → **Enterprise computing infrastructures**; • **Hardware** → **Enterprise level and data centers power issues**.

KEYWORDS

Data centers, Bayesian optimization, time-series modeling, cooling control optimization, thermal safety

ACM Reference Format:
Hanfei Geng, Yi Sun, Yuanzhe Li, Jichao Leng, Xiangyu Zhu, Xianyuan Zhan, Yuanchun Li, Feng Zhao, and Yunxin Liu. 2024. TESLA: Thermally Safe, Load-Aware, and Energy-Efficient Cooling Control System for Data Centers. In *The 53rd International Conference on Parallel Processing (ICPP '24), August 12–15, 2024, Gotland, Sweden.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3673038.3673144

*Corresponding author: Yunxin Liu (liuyunxin@air.tsinghua.edu.cn).

ICPP '24, August 12–15, 2024, Gotland, Sweden

ACM ISBN 979-8-4007-1793-2/24/08
https://doi.org/10.1145/3673038.3673144

1 INTRODUCTION

The exponential growth of artificial intelligence and cloud computing has led to a monumental expansion of the data-center (DC)

industry. However, this demand surge has led to a skyrocketing energy consumption. It is projected that DC electricity usage could triple or even quadruple by 2030 [12]. This substantial rise in energy demand not only poses challenges to environmental sustainability but also escalates operational costs for commercial DC providers.

Orthogonal to server-side optimizations, this paper considers improving DC's energy efficiency by optimizing its cooling system. It entails adjusting the control settings of air-cooling units (ACUs) in order to curb energy consumption under specified thermal safety constraints. For companies that host their proprietary data centers, cooling energy reduction that is thermally safe ensures servers' optimal efficiency and prevents equipment damage [20]. As for DC providers who offer their infrastructure to tenants who build their private clouds, it brings immediate monetary benefits.

Currently, the most common control setting of ACUs is its inlet temperature set-point [5, 42]. To reach a given set-point, ACUs utilize proportional-integral-derivative (PID) control [27]. To choose the set-point, current industry practice relies on human operators who select a value based on experience, whereas many data-driven methods [5, 8, 20, 23, 29, 35, 42, 43, 46] have also been proposed, which learn how to compute the set-point under dynamic load from historical data center traces.

Unfortunately, neither approach meets the goal of reducing DC's cooling energy within thermal safety. Human intervention cannot provide optimal cooling energy as the server load is unknown to the operator. A mismatch between cooling provisioning and server load results in unnecessary cooling energy consumption.

Existing data-driven methods, although load-aware, struggle to meet the thermal safety constraints due to overlooking the dynamics of ACU's PID controller. They adopt cooling energy minimization under thermal safety constraints as their optimization objective. This setup incentivizes ACU to operate at the boundary of the constraints. As a result, cooling interruption tends to occur. The set-point is higher than the actual inlet temperature such that the PID controller thinks it has over-achieved its objective and cold air is delivered at a reduced or zero rate. It is a phenomenon that does not conform to DC cooling standards dictated by *The American Society of Heating, Refrigerating and Air-Conditioning Engineers* (ASHRAE) [3]. When it happens, the temperature within the entire data center rises rapidly due to heat generated by the servers being removed at lower rates. Even if cold air starts to be delivered faster again, it takes much longer to undo the increase (Section 2.2), making previous data-driven methods unable to generate a set-point that curbs the temperature rise in time.

To overcome these issues, we propose TESLA, a thermally safe, load-aware, and energy-efficient cooling control system for data centers. TESLA employs a novel, DC-customized, and data-driven time-series modeling that predicts DC temperature and cooling energy under dynamic server load over a finite time window. Based on the predictions, TESLA uses a Bayesian optimizer (BO) that solves for the energy-optimal set-point under thermal safety constraints while accounting for its modeling error. Unlike previous data-driven methods which merely minimize cooling energy under thermal safety constraints, TESLA also adds minimizing cooling interruption in the optimization objective to prevent rapid temperature rise that violates thermal safety. We **deploy** TESLA in a real DC testbed (Section 4) with four racks and one ACU. We test TESLA's cooling strategy under different load settings to demonstrate TESLA's effectiveness. Our contributions are summarized as follows:

- We propose a novel data-driven time-series model to predict DC temperature and ACU's cooling energy given a specified set-point.
- We propose a modeling-error-aware Bayesian optimizer (BO) that solves for the energy-optimal set-point under thermal safety constraints.
- We incorporate minimizing cooling interruption into BO's optimization objective in order to prevent rapid temperature rise that violates thermal safety.
- We implement TESLA on a real DC testbed and achieve on average 10.1% cooling energy saving relative to adopting a fixed set-point and no thermal safety violation compared to previous data-driven control methods.

Paper roadmap. The rest of the paper is organized as follows. Section 2 discusses relevant background and challenges. Section 3 presents TESLA's system design, which includes its modeling, set-point optimization and execution. Section 4 discusses the deployment issue of TESLA. Section 5 presents our experimental results. Section 6 analyzes TESLA in depth. Section 7 reviews the relevant studies. Section 8 concludes this paper.

2 BACKGROUNDS AND CHALLENGES

We consider data centers with an air-cooling system as shown in Figure 1. The cooling system consists of ACUs that supply cold air to the servers for their temperature regulation. Air containment is installed to improve cooling efficiency, resulting in cold and hot aisle (labeled in the figure). We consider floor-level cooling [20] as our thermal safety constraint which requires cold aisle temperature to stay below a specified limit. In order to emulate real DCs, we constructed a testbed consisting of 21 servers over four racks and one ACU. The ACU is equipped with sensors that track its inlet temperature for the PID control. Temperature sensors are also installed on the racks to monitor the thermal state throughout the entire data center. We collect servers' resource utilization (CPU and memory), servers' and ACU's instantaneous power consumption, and temperature readings of both ACUs internal sensors and rack-installed sensors. Section 4 details our testbed configuration.

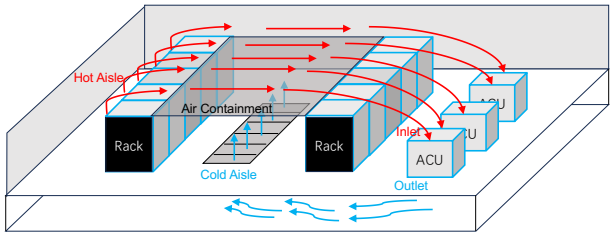


Figure 1: The type of data center considered in this work.

2.1 PID Control

An ACU utilizes proportional-integral-derivative (PID) control to reach a specified set-point. A PID controller generates a control signal to the ACU's responsible component that delivers cold air

(a compressor in the case of our testbed) based on the **residual error** (the difference between the set-point and the actual inlet temperature). The control signal consists of three terms: a term proportional to the current residual error, an integral term accounting for the accumulation of past residual errors, and a derivative term accounting for the rate of change of residual errors.

Linearly combining these three terms, a PID controller calculates the control signal sent to the compressor indicating how fast it should deliver cold air. Three terms' respective coefficients determine how fast the ACU can transition to a new set-point. When the residual error is a large negative value (set-point lower than the actual inlet temperature), the ACU's power consumption is high (as high as ~ 5kW on our testbed) as ACU is constantly sending cold air, and when the residual error is a large positive value (set-point higher than the actual inlet temperature), ACU power is low (as low as ~ 100W), during which cold air is delivered at a reduced or zero rate, i.e., **cooling interruption**.

2.2 Challenges

To achieve energy-efficient cooling control under thermal safety, TESLA must overcome the following challenges.

Modeling DC thermodynamics and black-box instantaneous ACU power. Selecting an energy-optimal set-point under dynamic load while providing thermal safety requires modeling the impact of a set-point on ACU's power and DC thermodynamics which refers to how DC temperature at any location changes over an **infinite** time window. However, it is not trivial to obtain such modeling. DC's thermodynamics are heavily influenced by server power, which is often hidden from DC providers [20]. As for power consumption, since ACUs typically come from third-party vendors, the inner mechanism that governs their power consumption under different set-points is unknown. Furthermore, ACU power can have high variance even under constant set-points due to server power fluctuations. Figure 2 shows that even though the set-point is constant, the ACU power can still vary between 2kW and 3kW, rendering it challenging to find a function between set-points and instantaneous ACU power.

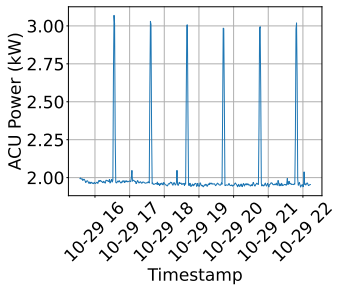
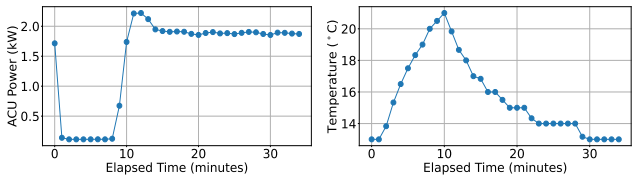


Figure 2: ACU power time series with set-point fixed at 27°C.

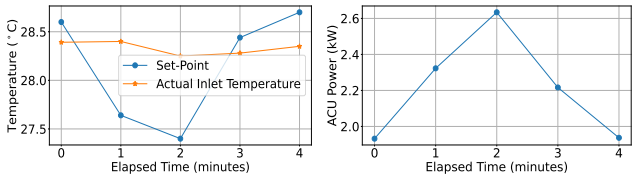
PID control dynamics. When cooling interruption occurs in Figure 3a during the first 10 minutes, it leads to a rapid temperature increase of the cold aisle as shown in Figure 3b (~ 1°C per minute) due to heat generated by the servers being removed at reduced rates. Although ACU eventually starts delivering cold air at a faster rate after $t = 10$ min as indicated by higher ACU power, it takes twice as long to undo this increase (~ 0.5°C per minute). Since cold aisle temperature is not directly regulated by ACU's PID controller, this rise could occur at any cold aisle temperature, including when it is close to the maximum limit, which does not give any buffer room to preserve thermal safety.



(a) (b)

Figure 3: a) ACU power and b) max cold aisle temperature under dynamic server load. Cooling interruption occurs during the first 10 minutes.

Control time granularity. It is also difficult [5, 34, 36] configuring an optimal control time granularity, the period at which a set-point is executed, in DC cooling control. If the granularity is too coarse, the controller does not respond to overheating events in time and a fine granularity creates frequent changes to the set-point, which need to be avoided for industrial equipment like ACUs [36]. Moreover, the PID control also brings energy implications. Figure 4 gives an example. Figure 4a shows that at $t = 0$ min, set-point is set at ~ 28.5°C, which decreases to ~ 27.5°C and comes back to ~ 28.6°C at $t = 2$ min and $t = 4$ min, respectively. Although 27.5°C is never achieved, additional power has already been consumed, as ACU's power increases by almost ~ 30% from 2kW to 2.6kW according to Figure 4b.



(a) (b)

Figure 4: a) Time series of ACU's actual inlet temperature and set-point and b) associated ACU power time series.

3 TESLA SYSTEM DESIGN

3.1 Overview

Design considerations. TESLA is a data-driven cooling control system for data centers, aiming to provide energy efficiency as well as thermal safety. Given the modeling challenge mentioned in Section 2.2, TESLA does not model how set-point changes ACU's instantaneous power or DC thermodynamics. Instead, TESLA's predictions are over a **finite** L-step horizon. It predicts how DC temperature changes as well as the associated cooling energy under a given set-point. Based on the predictions, it selects a set-point to be executed at the current time step, which minimizes the predicted cooling energy under the constraint that the predicted cold aisle temperature captured by any sensor does not exceed the specified limit. In addition to cooling energy minimization, the optimization objective also includes cooling interruption minimization in order

to curb the temperature rise risking thermal safety violation. Finally, due to the energy implication from set-point variations, the optimizer considers all time steps within the L-step horizon share the same set-point.

Overall workflow. These design considerations lead to the workflow of TESLA shown in Figure 5. At any time step t, a Bayesian optimizer (BO) computes a set-point for the interval $t+1$ to $t+L$ based on predictions of a DC time-series model. The predictions include DC temperature and cooling energy, which are made based on DC temperatures and server power from past L steps, specifically the interval from $t-L+1$ to t. The set-point computed by BO is not executed directly but passed through a smoothing buffer, which aims to limit the maximum set-point variations made at adjacent time steps. At time step $t+1$, the L-step window shifts and the DC time-series model makes predictions based on the time interval from $t-L+2$ to $t+1$.

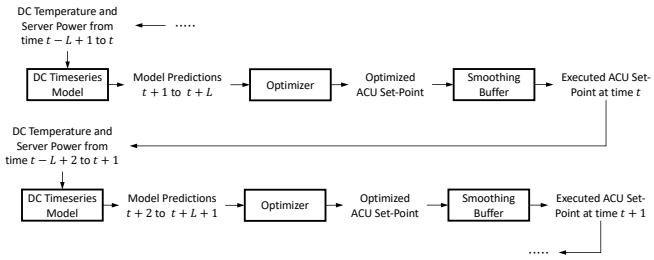


Figure 5: TESLA's overall workflow. At time t, set-point optimization is made for the next L steps using DC temperature from the past L steps, which is repeated at $t+1$, $t+2$, etc.

3.2 Data Center Time-Series Model

Design considerations. DC temperature is monitored by various sensors installed along the racks. Accurate prediction of future DC temperature relies on not only modeling the interdependence among sensors but also including influence due to exogenous inputs. In this case, there are two exogenous inputs, the heat-generation rate within the data center due to server power and the heat-removal rate indicated by the actual inlet temperature regulated by ACU's PID controller. However, including the impact of the exogenous inputs requires their values at time step $t+1, t+2, ..., t+L$, which are not accessible at time step t. Consequently, they must be predicted before computing future DC temperature, leading to the following model architecture shown in Figure 6.

Model architecture. In summary, TESLA's DC time-series model uses four sub-modules to predict the DC temperature and cooling energy. The first stage of TESLA's DC timerseries model builds an average server power (ASP) sub-module to predict the average server power of the next L steps. The second stage uses an ACU sub-module to predict ACU's actual inlet air temperature based on the given set-point and ASP's outputs. The ASP and ACU sub-modules give indications on how much heat is being generated and removed from the data center. Lastly, a DC sensor (DCS) model predicts the future DC temperature captured by sensors on different racks. As for the cooling energy, we use a cooling energy-sub-module that takes the predicted ACU inlet temperature and the ACU set-point as

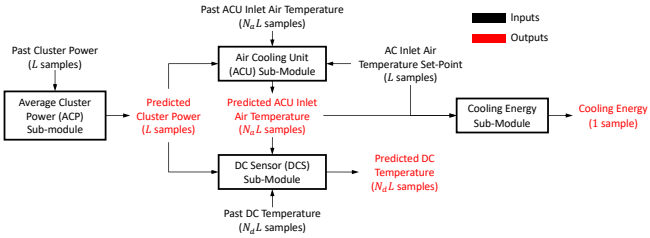


Figure 6: TESLA's DC time-series modeling architecture, where L is the prediction horizon, N_a is the number of ACU internal sensors, and N_d is the number of sensors installed along racks in the data center.

its inputs. The reason for this input-feature choice is that the cooling energy is directly influenced by the PID controller, which serves to reduce the residual error between ACU inlet temperature and the given set-point as mentioned in Section 2.1. Each sub-module uses linear regression with the direct strategy to map its inputs to outputs, which has shown effectiveness in modeling DC temperature [20] as well as general time-series forecasting tasks [45]. Specifically, the temperature at different steps within the L-step horizon uses different regression weights and biases.

Average server power sub-module. Server power represents the rate at which heat is generated within the data center. However, each server's power consumption could change abruptly, making it difficult to predict due to unknown upper-level workloads. To solve this problem, TESLA predicts the average power aggregated over servers adjacent to the ACU based on DC layouts, as it is less likely for multiple servers to change their power consumption simultaneously than a single server [19].

Specifically, the average server power p_{t+l} at the l^{th} step from the current time t, where $l \in \{1, 2, 3, ..., L\}$, is computed as follows:

$$p_{t+l} = \beta_l + \sum_{j=0}^{L-1} \beta_{l,j} p_{t-j} \tag{1}$$

where p_{t-j} is the average server power at time step $t-j$ and βs are the associated weights and bias.

Air-cooling unit sub-module. The actual inlet temperature represents the rate at which heat is removed from the data center. In addition, it also serves as the input for the following cooling energy sub-module, as the residual error of ACU's PID controller determines the cooling energy consumption. Since ACU typically uses multiple internal sensors to track its inlet temperature, we need to account for the interdependence among them.

Assuming N_a internal sensors, we compute the temperature $a^{n_a}_{t+l}$ captured by the n_a^{th} sensor at the next l^{th} step, where $n_a \in [N_a]$ and $l \in \{1, 2, 3, ..., L\}$, as follows:

$$a^{n_a}_{t+l} = \gamma^{n_a}_l + \gamma_1 s_{t+l} + \gamma_2 p_{t+l} + \left(\sum_{i=0}^{N_a-1} \sum_{j=0}^{L-1} \gamma^{n_a}_{l,i,j} a^i_{t-j} \right) \tag{2}$$

where the γs are the ACU sub-module's weights and bias, s_{t+l} is the set-point at the next l^{th} step from current time t, p_{t+l} is the predicted average server power at the next l^{th} step, and a^i_{t-j} is the i^{th} ACU sensor's reading at time step $t-j$, respectively. We use superscripts to represent sensor indices and subscripts

to represent time indices. The final double summation term uses weighted averaging to account for the contribution of past samples of all ACU sensors.

Data center sensor sub-module. The aforementioned ASP and ACU sub-modules provide the required exogenous inputs. Finally, TESLA uses a DCS model to predict how temperature evolves at different rack-installed sensors due to the exogenous inputs.

Given N_d sensors, temperature $d_{t+l}^{n_d}$ captured by the n_d^{th} sensor at the next l^{th} step, where $n_d \in [N_d]$ and $l \in \{1, 2, 3, ..., L\}$, is computed as follows:

$$d_{t+l}^{n_d} = \theta_l^{n_d} + \theta_1 p_{t+l} + \Big(\sum_{i=0}^{N_a-1} \theta_{l,i}^{n_d} a_{t+l}^{i} \Big) + \Big(\sum_{k=0}^{N_d-1} \sum_{j=0}^{L-1} \theta_{l,j,k}^{n_d} d_{t-j}^{k} \Big) \quad (3)$$

where the θs are the DCS sub-module's weights and bias, p_{t+l} is the predicted average server power at the next l^{th} step, a_{t+l}^{i}s are the predicted ACU's inlet temperature of the i^{th} ACU sensor at time step $t+l$, and d_{t-j}^{k} is the temperature of the k^{th} sensor at time step $t-j$. Same as Equation (2), superscripts are used to represent sensor indices, whereas subscripts represent time indices. The first single summation term accounts for the contribution from predicted AC inlet temperature, and the final double summation term accounts for the contribution of past samples of all DC rack-installed sensors.

Cooling energy sub-module. As mentioned in Section 2.2, directly modeling ACU's instantaneous power is ill-advised due to its high variance even under constant set-point and server power. As a result, we model the cooling energy over the prediction horizon. Specifically, we compute the total energy consumed by the ACU over L steps using the observed instantaneous power and build a cooling energy sub-module that maps the corresponding set-point to the computed energy value.

Given ACU's instantaneous power trace $\{p_{t+j}^{\text{ACU}}\}_{j=1}^{L}$ between time step $t+1$ and $t+L$, the energy E_{t+1}^{L} spent during this time interval ends at time $t+1$ is obtained using numerical integration in kilowatt-hour. Next, we find a mapping that connects the set-point to E_{t+1}^{L}. Since the PID controller operates based on the difference between the set-point and ACU's inlet temperature, we use all of their samples over the time interval as the input features. The sub-module computes E_{t+1}^{L} as follows:

$$E_{t+1}^{L} = \phi_E + \Big(\sum_{i=1}^{L} \phi_i^{s} s_{t+i} \Big) + \Big(\sum_{n_a=0}^{N_a-1} \sum_{i=1}^{L} \phi_{n_a,i}^{a} a_{t+i}^{n_a} \Big) \quad (4)$$

where the ϕs are the weights and bias, s_{t+i} is the set-point value at time step $t+i$ with $i \in \{1, 2, ..., L\}$, and $a_{t+i}^{n_a}$ is the ACU's inlet temperature captured by the n_a^{th} sensor in a total of N_a sensors.

Training methodology and loss function. Training TESLA's DC time-series model requires finding the weights and biases for each sub-module, i.e., the βs, γs, θs, and ϕs from historical data. Since three of the four sub-modules, ACU, DCS, and the cooling energy sub-modules, require predicted values as their inputs, training them iteratively in an end-to-end manner is challenging because the predicted values are random at the beginning of the training.

Instead, the sub-modules' weights and biases are retrieved separately. The ACU and DCS models use true ACU inlet temperature and average server power to predict their respective outputs $a_{t+l}^{n_a}$ and $d_{t+l}^{n_d}$. Each target value has its own dataset. All of them use mean squared loss (MSE) as the loss function. It results in $(1+N_a+N_d)L$ regression problems according to Figure 6, whose analytical solutions can be obtained directly [25].

Choosing regularization strength. We use the standard $L2$ regularization to penalize the norm of the weights of the sub-modules, where α_β, α_γ, α_θ, and α_ϕ control the regularization strength for each sub-module, respectively. $L2$ is adopted since $L1$ regularization would have promoted sparse weights, discounting contributions from most input features. Since ACU, DCS, and the cooling energy sub-modules see predicted values during inference instead of true ones, we account for this discrepancy by setting α_γ, α_θ, and α_ϕ to 1, turning the regression problems in these sub-modules into ridge regressions instead of adopting the ordinary least square (OLS) solution [9].

3.3 Set-Point Optimization

Design considerations. The time-series model establishes how a given set-point influences DC temperature at different locations in the data center over the finite time horizon. Based on the model's predictions, TESLA uses an optimizer to find the energy-optimal set-point without violating thermal safety constraints. However, the optimizer needs to be aware of the modeling error of the DC time-series model, even though we have made efforts to reduce its generalization error by incorporating the impact of exogenous inputs. Generalization error is only an averaged metric. Therefore, large deviations can still occur. When they happen, subsequent set-point optimization is conducted based on wrongful predictions, making the data center potentially vulnerable to thermal safety violations and sub-optimal energy savings.

Formulating the optimization problem. Consequently, TESLA forms the following optimization problem for interval $t+1$ to $t+L$.

$$\begin{aligned} &\text{argmin}_{s_{t+1},...,s_{t+L}} && TO_{t+1}^{L}(s_{t+1}, ..., s_{t+L}) \\ &\text{subject to} && TC_{t+1}^{L}(s_{t+1}, ..., s_{t+L}) \le 0 \\ & && TO_{t+1}^{L}(s_{t+1}, ..., s_{t+L}) = O_{t+1}^{L} + \epsilon \\ & && TC_{t+1}^{L}(s_{t+1}, ..., s_{t+L}) = C_{t+1}^{L} + \eta \\ & && s_{t+1} = ... = s_{t+L} = S_{t+1}^{L} \\ & && S_{\min} \le S_{t+1}^{L} \le S_{\max} \end{aligned} \quad (5)$$

where $s_{t+1}, ..., s_{t+L}$ is the set-point sequence. The optimizer maximizes an objective function TO_{t+1}^{L} that includes the true cooling energy subject to a constraint function TC_{t+1}^{L} that indicates true thermal safety violation if it is positive. However, only their predicted version O_{t+1}^{L} and C_{t+1}^{L} can be obtained directly through the DC time-series model, where ϵ and η are the modeling error. In order to avoid the energy implication from set-point variations as mentioned in Section 2.2, TESLA optimizer considers all time steps within the L-step horizon share the same set-point, i.e., $s_{t+1} = ... = s_{t+L} = S_{t+1}^{L}$. $S_{\min}$ and $S_{\max}$ are the minimum and maximum set-point values, respectively, achievable by the ACU's PID controller given in the ACU's specification.

Optimization objective. O_{t+1}^{L} consists of two terms. The first term is the cooling energy estimated by the cooling energy sub-module in the DC time-series model, while the second term reflects

cooling interruption. We include it in the optimization objective as it leads to rapid temperature rise in the data center due to heat being removed at reduced or zero rate. Additionally, according to DC cooling standards dictated by *The American Society of Heating, Refrigerating and Air-Conditioning Engineers* (ASHRAE) [3, 30], the duration of cooling interruption in data centers is limited based on their availability specifications. Although cooling interruption provides an energy-saving incentive, it conflicts with this requirement.

To compute cooling interruption D^L_{t+1}, we use the residual error of ACU's PID controller, i.e., the difference between the set-point and the actual inlet temperature over the time interval $t+1$ to $t+L$ as a proxy:

$$D^L_{t+1} = \sum_{j=1}^{L} U_{t+j} \tag{6}$$

where U_{t+j} is the residual error at time $t+j$ given by:

$$U_{t+j} = \begin{cases} s_{t+j} - \text{avg}_{N_a}(a^{n_a}_{t+j}) & \text{if } s_{t+j} - \text{avg}_{N_a}(a^{n_a}_{t+j}) \geq \kappa \\ 0 & \text{Otherwise} \end{cases} \tag{7}$$

where s_{t+j} is the set-point at time $t+j$, avg_{N_a} denotes the averaging operation across N_a ACU sensors, and $a^{n_a}_{t+j}$ is the inlet temperature of sensor n_a at time $t+j$. κ is a positive number that controls how much cooling interruption is penalized. Setting $\kappa = 0$ does not allow any interruption. The DC time-series model computes the objective function O^L_{t+1} as the summation of the aforementioned two components.

$$O^L_{t+1} = E^L_{t+1} + D^L_{t+1} \tag{8}$$

where E^L_{t+1} and D^L_{t+1} are given by Equation 4 and Equation 6, respectively.

Constraint. Given the floor-level cooling as our thermal safety constraint that the sensors monitoring the cold aisle must not exceed a given temperature, the constraint C^L_{t+1} computed by DC time-series model is computed as:

$$C^L_{t+1} = \max_{n_k, t+j}(d^{n_k}_{t+j}) - d_{\text{allowed}} \tag{9}$$

where $j \in \{1, 2, ..., L\}$ and n_k belongs to the index set I_{cold} which denotes the sensors installed in the cold aisle. $d^{n_k}_{t+j}$ is the corresponding sensor reading at time $t+j$. C^L_{t+1} computes how far the maximum temperature captured at the next L steps by the cold aisle sensors is from the allowed temperature d_{allowed}. If C^L_{t+1} is negative, it means that the constraint is not breached and is so otherwise.

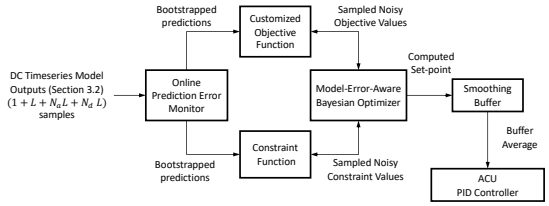


Figure 7: Overview of how TESLA computes and executes the optimal set-point based on the DC time-series model from Section 3.2.

Optimizer workflow. To solve the constrained optimization problem (5), we use the workflow shown in Figure 7 while accounting the DC timerseries modeling error. We choose Bayesian optimization [13] as our optimizer backbone, which has been effective in optimizing low-dimensional black-box functions such as the mappings from S^L_{t+1} to TO^L_{t+1} and TC^L_{t+1}, respectively. Based on the outputs of the DC time-series model described in Section 3.2, TESLA's optimizer samples values from distributions of the objective and constraint functions. It uses a modeling-error-aware Bayesian optimizer to find the optimal feasible set-point. To construct the distributions of the objective and constraint functions, uncertainty estimation of the objective and constraint is needed. TESLA uses an online prediction error monitor that keeps track of the prediction error made by the DC time-series model within the past day, which is a typical period where the data center load rises and falls [10]. The uncertainty estimates are obtained from the monitor using bootstrapping [16, 39]. After the Bayesian optimizer computes a set-point, it is sent to a smoothing buffer whose output is finally executed by ACU's PID controller.

Bayesian optimization primer. Conventional BO [13] relies on two components. The first one is a surrogate model that maps S^L_{t+1} to TO^L_{t+1}. Based on observed (S^L_{t+1}, TO^L_{t+1}) pairs, the model fits a maximal a-posterior estimate of the underlying distribution. Typically, a Gaussian process (GP) is used in this process. TC^L_{t+1} is usually added to TO^L_{t+1} during the fitting. BO proceeds iteratively to find set-point that yields better TO^L_{t+1} using an acquisition function. Conventional BO thus requires TO^L_{t+1} and TC^L_{t+1} to be accessible for any set-point. Unfortunately, in this case, O^L_{t+1} and C^L_{t+1} are available, which is why we resort to modeling-error-aware BO.

Surrogate model. We choose GP to create the distributions of objective and constraint functions. Unlike conventional BO where the objective and the constraint are shared the same GP, we have separate fixed-noise GPs that fit TO^L_{t+1} and TC^L_{t+1} using O^L_{t+1} and C^L_{t+1} for fast and accurate convergence [49]. Matern (5/2) [37] is used as the covariance kernel.

Uncertainty estimation using bootstrapping. The fixed-noise GP requires uncertainty estimation as its input. To achieve this goal, TESLA uses a prediction error monitor that keeps track of the past prediction errors made by TESLA's DC time-series modeling. For example, when deciding at time t for time interval $t+1$ to $t+L$, prediction errors up till time t have become available. Bootstrapping [16, 39] allows TESLA to repeatedly sample from the observed prediction errors N_b times. to create N_b versions of O^L_{t+1} and C^L_{t+1} under the same set-point from which the variance of the prediction error can be computed.

Acquisition function. Usually, the expected improvement (EI) acquisition function [40] is adopted to find the energy-optimal set-point under the thermal safety constraint. However, it assumes accessible TO^L_{t+1} and does not handle hard constraints in the constrained optimization problem 5. To address this issue, TESLA's optimizer adopts constrained noisy expected improvement (NEI) [21] with quasi-Monte Carlo integration (QMC) as its acquisition function, which assumes the observed objective and constraint values are not perfect and can process hard constraints. Finally, it is still possible that none of the set-points considered satisfies the

constraint, indicating that the constraint is close to being breached. Therefore, TESLA selects $S_{\min}$ and it will re-calibrate itself later.

3.4 Set-Point Execution

After the set-point is computed by the optimizer, it requires post-processing to be sent to the ACU's PID controller for execution. The reason is that set-point transitions take time, which varies based on server power and the previous set-point. As a result, executing set-points periodically is bound to have some of set-points be executed when the ACU has not stabilized around the previous ones yet. This type of execution risks additional energy consumption as demonstrated in Section 2.2. To avoid such cases, we add a smoothing buffer of length N_s that stores the set-points computed by TESLA's optimizer. The set-point that finally sent to the ACU's PID controller is the running average of values stored in the smoothing buffer. The buffer acts as a low-pass filter that removes the high-frequency variations in the computed set-points and thus alleviates unnecessary power peaks.

4 IMPLEMENTATION

Testbed. Our testbed consists of 21 servers installed on four racks. The servers communicate over one networking switch. $N_d = 35$ sensors are installed on racks to monitor DC temperature, 11 of which monitor the temperature of the cold aisle. Air containment is installed that separates the cold aisle from the hot aisle. The ACU is manufactured by [11], which has $N_a = 2$ sensors to track its inlet temperature for PID control. A Kubernetes [18] cluster is built using these servers for workload orchestration. We adopt Influxdb plus Telegraf [31] as our observability solution. Each server runs a Telegraf process that collects metrics including server power, CPU utilization, and memory utilization. Metrics related to ACU and DC sensors, specifically temperature data, are also collected using Telegraf with the Modbus protocol. All data is ported into a management server outside the cluster running Influxdb. The details of the software and hardware are listed in Table 1.

Table 1: Testbed details.

Testbed Hardware	
Servers	21
CPU	112-core Xeon® Gold 6330 (11 servers) 88-core Xeon® E5-2699 (10 servers)
ACU	Evicool [11] XR023A
Max allowable set-point $S_{\max}$	35°C
Min allowable set-point $S_{\min}$	20°C
Number of Cold Aisle Sensors	11
Number of Total Sensors N_d	35
Number of ACU Sensors N_a	2
Testbed Software	
Architecture	1 Master + 20 Slaves
Workload Orchestration	Kubernetes 1.25.12
Observability	InfluxDB 2.7.0 + Telegraf 1.29.1
Operating System	CentOS Stream 9

Workload Generation. We use the controller introduced by Gaetano [14] to create a steady CPU load at any given duration on a single server. The controller takes the target CPU cores, desired load level, and duration as its inputs. In order to create varying CPU load in the entire cluster, we deploy the controller to our testbed using the Job resource of Kubernetes [17].

TESLA. We trained the DC time-series model using Skforecast [2] with Scikit-learn [33]. The prediction error monitor and the bootstrapping sampling are PyTorch-based [28]. We implement the Bayesian optimizer in BoTorch [4], which uses the FixedNoiseGP class that calls GPyTorch [15] to implement the Gaussian processes. Our main function is implemented using two Python processes, a producer and a consumer that communicate over a message queue. One process periodically pulls testbed information, e.g., CPU utilization, server power, and temperature from InfluxDB and pushes it onto the message queue. The consumer process pulls the InfluxDB data from the queue and runs it through TESLA according to Figure 5. When the smoothing buffer outputs a set-point ready for execution, TESLA writes the value in the register of ACU's PID controller through the Modbus protocol.

5 EVALUATION

5.1 Setup

Server load settings. We adjust the desired load level of Gaetano [14]'s controller at 1-minute granularity to emulate a typical diurnal pattern seen in DCs, measured in average CPU utilization. To expedite each experiment, the testing period is set to 12 hours, i.e., the server load rises and falls over 12 hours. We consider three diurnal load settings in order to test TESLA's performance under dynamic server power. The load settings are selected based on cluster data collected from production clusters in Alibaba [10].

(1) Idle: the controller is not activated and no load is created in the testbed.
(2) Medium load: the average CPU utilization of the entire testbed over the 12-hour period is 20%.
(3) High load: the average CPU utilization of the entire testbed over the 12-hour period is 40%.

Datasets, preprocessing and metrics. For every 12 hours, we randomly pick a server load setting. During this period, the set-point is swept from 20°C to 35°C, which changes 0.5°C every 5 minutes. We repeat this operation for 1 month and collect the testbed traces in order to build our training dataset. The testing dataset is built using traces from another two weeks. All data from InfluxDB are normalized to the range of 0 and 1 using min-max normalization as the preprocessing step. We use mean absolute percentage error (MAPE) as the accuracy metric for modeling, which is the absolute difference between the predicted value and the ground truth divided by the ground truth. For the end-to-end performance, we compute the cooling energy over the testing period and the period during which the thermal safety constraint is breached.

Hyperparameters. We list the relevant hyperparameter values in Table 2 along with their symbols and meanings. α_γ, α_θ, and α_ϕ are chosen to be nonzero since these model inputs take true values during training and predicted versions during inference. The regularization choice is to prevent large output swings due to input errors from predictions. The limit of cold aisle temperature d_{allowed} of 22°C is chosen according to DC cooling standards of ASHRAE [3], which is also used to evaluate TESLA's and other methods' thermal safety violation.

Table 2: Relevant hyperparameters.

Symbol	Meaning	Value and Unit
α_β	Regularization for ASP sub-module	0
α_γ	Regularization for ACU sub-module	1
α_θ	Regularization for DCS sub-module	1
α_ϕ	Regularization for cooling energy sub-module	1
N_b	Number of bootstrapping samples	500
N_s	Size of the smoothing buffer	5
d_{allowed}	Limit of cold aisle temperature	22 °C
κ	Threshold for penalizing ACU's cooling interruption	0.5 °C
L	Prediction horizon	20
Δt	Sampling period	1 min

5.2 Modeling Results

DC temperature modeling. We compare TESLA's time-series model with models used by Lazic et al. [20] from Google and Wang et al. [42] to evaluate TESLA's performance in terms of DC temperature prediction. They are state-of-the-art works that model DC temperature for controlling DC's cooling system. Lazic et al. [20] uses linear regression with ordinary least square [9], whereas Wang et al. [42] uses multi-layer perceptron (MLP) [38]. Their MAPEs are listed in Table 3, which shows that TESLA outperforms these alternatives. The reason for this performance gain is that these baselines generate their outputs recursively, unlike TESLA that produces its outputs at each time step in parallel. Moreover, these baselines model all DC temperature in a data center collectively with the cooling demand (server power) and provisioning (ACU's inlet temperature), whereas TESLA predicts their values first in order to account for their dynamicity.

Cooling energy modeling. Table 4 benchmarks TESLA's DC timeseires model in terms of predicting the cooling energy. We consider the following non-linear models as baselines: MLP [42], XGBoost [7], and Random Forest [26]. The table shows that TESLA's cooling energy sub-module achieves the lowest MAPE relative to the other three alternatives.

Table 3: DC temperature MAPE.

Prediction Error	TESLA (Ours)	Lazic et al. [20]	Wang et al. [42]
MAPE(%)	**3.52**	5.52	10.73

Table 4: Cooling energy MAPE.

Prediction Error	TESLA (Ours)	MLP [38]	XGBoost [7]	Random Forest [26]
MAPE(%)	**7.90**	14.33	13.41	15.11

5.3 End-to-End Performance

We evaluate TESLA's end-to-end performance under the aforementioned three load settings using: the cooling energy (CE) spent during a 12-hour testing period and the amount of time of thermal safety violation (TSV). Three other alternatives are considered: a static policy where the set-point for the ACU's inlet temperature is fixed at 23°C, the DC cooling control method proposed by Lazic et al. [20] from Google based on model predictive control [1], and TSRL [8], a state-of-the-art offline reinforcement learning method. Lazic et al. [20] relies on an autoregressive linear modeling for DC temperature prediction, based on which a gradient-descent optimizer chooses the highest set-point such that the predicted maximum cold aisle temperature stays below the specified 22°C limit. TSRL [8], on the contrary, directly outputs the set-point decision without modeling DC temperature or cooling energy. It uses cooling energy saving as its reward and thermal safety violation as its cost. The learning process tries to maximize the reward while minimizing the cost based on historical DC traces. We trained TSRL [8] with the same training set used to train TESLA's DC time-series model.

The end-to-end results are shown in Table 5. We also include the duration of cooling interruption (CI) indicated by ACU power below 100W in our testbed. The table shows that TESLA outperforms the fixed 23°C policy with 5.24% – 15.3% (10.1% on average) of cooling energy saving under no thermal safety violation. The higher the load is, the more cooling energy TESLA saves (15.3% at the high load setting). The source of TESLA's energy savings is analyzed in Section 6.2. Although Lazic et al. [20] and TSRL [8] save more cooling energy compared to TESLA, the table shows that they can not provide thermal safety guarantee under all load settings due to extensive cooling interruption. Lazic et al. [20] and TSRL [8] cause at least 16.9% TSV, while TESLA does not have any violation. We analyze why Lazic et al. [20] and TSRL [8] cannot provide such guarantee in Section 6.3.

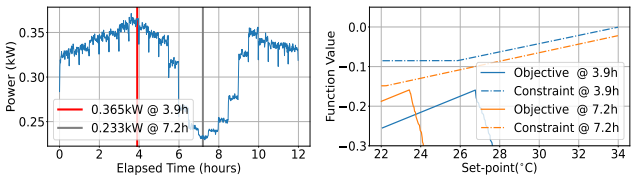
Table 5: End-to-end performance benchmark, including cooling energy (CE) along with percentage savings, thermal safety violation (TSV), and cooling interruption (CI) under idle, medium load, and high load settings. The relative CE saving is calculated with respect to the fixed 23°C policy.

Load-Setting	Metric	Fix 23°C	TESLA (Ours)	Lazic et al. [20]	TSRL [8]
Idle	CE (kWh)	24.8	23.5	11.9	12.6
	CE Saving (%)	0	5.24	52.0	49.2
	TSV (%)	0	0	24.0	40.4
	CI (%)	1.00	2.00	34.4	48.6
Medium	CE (kWh)	26.4	23.8	19.4	18.6
	CE Saving (%)	0	9.84	27.6	30.6
	TSV (%)	0	0	22.1	23.2
	CI (%)	1.00	2.00	32.4	21.0
High	CE (kWh)	28.7	24.3	22.7	20.2
	CE Saving (%)	0	15.3	26.4	42.1
	TSV (%)	0	0	25.0	16.9
	CI (%)	1.00	2.00	35.2	17.5

6 DISCUSSION

6.1 How TESLA Computes its Optimal Set-Point

We use two different time instants to illustrate how TESLA computes its optimal set-point at any time instant. Figure 8a shows the average server power over the 12-hour testing period under

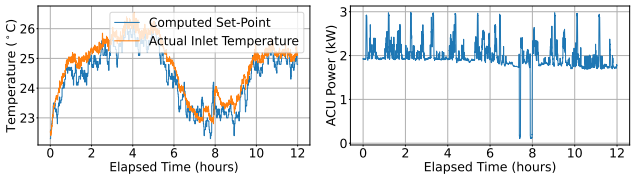


(a) Average server power over the testing period with medium load. The legend marks powers at 3.9h and 7.2h.

(b) Mean objective and constraint functions given by the Gaussian process at 3.9h and 7.2h, respectively.

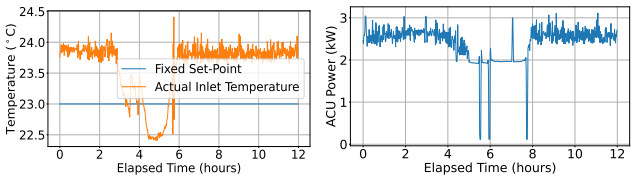
Figure 8: An overview of how TESLA computes its optimal set-point at different time instants.

medium load setting and Figure 8b shows the mean objective and constraint functions from TESLA's Gaussian processes. At $t = 3.9$h, the servers see a high average power with each machine working at 0.365kW shown by Figure 8a. At this time, TESLA finds the set-points which lead to negative constraint function values first, indicating thermal safety would not be breached for the next L steps, which is any set-point below 34°C. Next, it selects the one which maximizes the objective function, which is the peak at 26.8°C. The set-point is then passed to the smoothing buffer whose output gets executed and affects the DC temperature evolution, which leads to new objective and constraint functions at $t = 7.2$h. At this time instant, the aforementioned steps are repeated, which lead to an optimized set-point of 23.4°C that passed to the smoothing buffer.



(a) **(b)**

Figure 9: TESLA's a) set-point and actual inlet temperature trace in addition to b) its ACU power.



(a) **(b)**

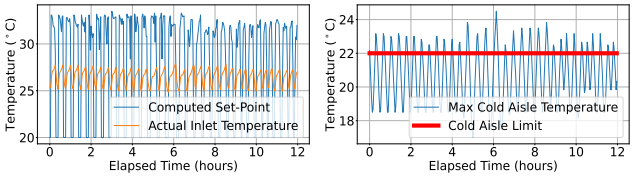
Figure 10: The a) set-point and actual inlet temperature trace in addition to b) the associated ACU power when the set-point is fixed at 23°C.

6.2 Energy Saving Analysis

We compare TESLA's computed set-point and associated ACU power with those of the fixed 23°C-policy to analyze how TESLA saves cooling energy. Figure 9 shows TESLA's computed set-point and actual inlet temperature trace as well as its ACU instantaneous power, while Figure 10 shows those of the fixed 23°C-policy. Although physical intuition indicates that a lower server power only needs a higher set-point, the figure shows that it is not the case for PID-enabled ACUs. When the server power is low, it becomes easier to achieve a lower inlet temperature. Having a set-point significantly higher than the actual inlet temperature risks cooling interruption. On the other hand, when the server power is high, the actual inlet temperature increases as well. Using a set-point significantly lower than the actual inlet temperature incurs more cooling energy. Figure 10 shows a large residual error between set-point and the actual inlet temperature, indicating it is difficult for the PID controller to achieve the set-point. As a result, the ACU constantly spends ~ 2.5kW during the first four hours, whereas the ACU's instantaneous power stays at ~ 2kW in the case of TESLA. Therefore, TESLA saves cooling energy by selecting **the highest set-point such that cooling interruption is minimized**. Since there is barely any time (~ 1%) where cooling is interrupted, it becomes less likely for TESLA to violate the thermal safety constraints.

6.3 Thermal Safety Analysis

We analyze the computed set-point trace and max cold aisle temperature of Lazic et al. [20] and those of TSRL [8] to show why they cannot provide thermal safety guarantee. Figure 11 shows the set-point, actual inlet temperature, and max cold aisle temperature for Lazic et al. [20]. In contrast to TESLA, which selects the highest set-point that minimizes cooling interruption, Lazic et al. [20] considers only cooling energy as its optimization objective. This setup makes it select **the highest set-point as long as the constraint function is negative**, promoting the ACU to operate at the boundary of the cold aisle limit. As a result, cooling interruption happens which leads to rapid temperature increase in the cold aisle that crosses the cold aisle limit. When Lazic et al. [20] tries to compute the set-point near these time instants, no feasible set-point can be found, which triggers a backup strategy of selecting $S_{\min} = 20$°C as shown in Figure 11a. However, since the ACU operates ≤ 4°C from the 22°C limit, the rapid rise of cold aisle temperature (~ 1°C per minute shown by Figure 3b does not give Lazic et al. [20] enough margin to curb the rise in time, leading to frequent overshoot of the cold aisle temperature limit as shown in Figure 11b.

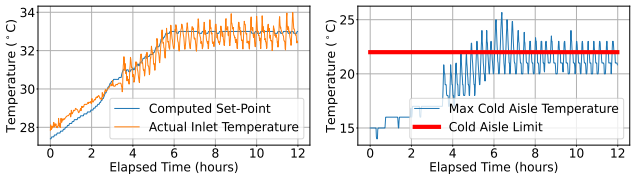


(a) **(b)**

Figure 11: Lazic et al. [20]'s a) set-point and actual inlet temperature trace in addition to b) its max cold aisle temperature.

Figure 12 shows the set-point, actual inlet temperature, and max cold aisle temperature for TSRL [8]. Like Lazic et al. [20], TSRL [8] also promotes ACU to operate at the boundary of the cold aisle limit, as it considers only cooling energy as its optimization objective. As indicated in Figure 12b, the set-point is selected such that the max cold aisle temperature gradually approaches 22°C cold aisle limit. However, since it also does not include cooling interruption into its

consideration, Figure 12b shows that it cannot curb the resultant temperature rise in time, leading to frequent overshoot of the cold aisle temperature limit similar to Lazic et al. [20].



(a) (b)

Figure 12: TSRL [8]'s a) set-point and actual inlet temperature trace in addition to b) its max cold aisle temperature.

7 RELATED WORK

Cooling control for data centers. The first requirement of DC cooling control is the collection of DC temperature data. RACNet [24] pioneers the use of wireless sensors to monitor DC temperatures among other things to enable fine-grained modeling of DC dynamics for effective cooling control and optimization. The maturity of DC temperature monitoring motivates recent works to adopt a data-driven approach, which consists of two directions: deep reinforcement learning (DRL) [5, 8, 23, 29, 35, 42, 43, 46] and model predictive control (MPC) [20]. DRL relies on agents to learn a control policy through interactions with the DC that maps currently observed DC temperature to a set-point. Initially, online model-free DRL methods [23, 29, 35] adopt direct agent-environment interactions. Due to drawbacks of learning inefficiency and thermal safety concerns, subsequent methods [5, 8, 42, 43, 46] consider the model-based alternative, in which a model tracks DC temperature and guides the learning process of the control policy. They use energy saving as rewards and thermal safety violations as costs in order to provide energy optimality while preserving thermal safety. MPC [20], on the other hand, chooses the energy-optimal set-point under thermal safety through explicit modeling of DC temperature and cooling energy. However, existing DRL methods do not provide interpretability for their decisions, as the mapping between DC temperature and the optimized set-point is encapsulated by deep neural networks. On the contrary, TESLA's decision process can be visualized at every control step through figures like Figure 8b. In addition, existing DRL methods are evaluated using simulators, whereas TESLA is **deployed** in a real DC environment. Finally, neither existing DRL nor MPC methods for DC cooling control account for the dynamics of ACU's PID controller.

Modeling DC temperature and cooling energy. The end-to-end performance of DC cooling control depends on the prediction accuracy of DC temperature and cooling energy. To achieve this goal, it is necessary to model not only the temporal relation of each sensor but also the sensors' interdependence. Both transformers and linear models have shown potentials due to their accuracy for a variety of multivariate time series [32, 44, 45, 48]. For the DC scenario specifically, Lazic et al. [20] relies on a single autoregressive linear model. In contrast, TESLA adopts a direct strategy, using a group of linear models that explicitly captures the impact of exogenous inputs, i.e., server power and ACU inlet temperature, in order to provide more accurate predictions.

Server power control in data centers. Besides energy-efficient cooling control, many works also focus on server power control since servers constitute the primary contributors of energy consumption in data centers. Chen et al. [6] develops the first energy-aware dynamic server provisioning for stateful services, demonstrating significant energy saving for a major Microsoft cloud service. At the infrastructure level, Li et al. [22] adopts a power-oversubscription approach where more servers are subscribed to a single rack than its allowed capacity. Power capping is employed to prevent overload in a throughput-optimized and quality-of-service-aware fashion. Zhang et al. [47] and Stojkovic et al. [41] tackle the power control problem from the perspective of serverless computing, a cloud computing paradigm that starts to gain popularity among different businesses. These methods complement TESLA in order to provide a comprehensive solution for energy-efficient data centers.

8 CONCLUSION

This paper proposes TESLA, a thermally safe, load-aware, and energy-efficient cooling control system for data centers. TESLA uses data-driven modeling to predict DC temperature and cooling energy based on which the set-point is optimized for PID-enabled ACUs in existing DCs. Relative to a fixed policy, TESLA provides energy efficiency by selecting set-points that match provisioned cooling resources with server loads. Compared to prior data-driven methods, TESLA provides thermal safety by explicitly considering rapid temperature rise due to the control dynamics of ACU's PID controller. Since the decision-making process is decoupled into two separate stages: modeling and optimization, DC operators can gain insights of why a particular set-point is selected. Moreover, since the set-point optimization takes place at every control step, TESLA can adjust the thermal safety constraints during deployment without retraining, while existing DRL methods have to retrain their agents. TESLA improves DC's energy efficiency by reducing the energy of the cooling system relative to that of servers. One future direction is to optimize DC's total energy consumption by integrating TESLA with server-side optimizations such as energy-aware workload scheduling. We leave it as our next step.

ACKNOWLEDGMENTS

This work is supported by NSFC No.62302262 and the joint research project with Intel Corporation (No. 20233000111).

REFERENCES

[1] Abdul Afram and Farrokh Janabi-Sharifi. 2014. Theory and applications of HVAC control systems–A review of model predictive control (MPC). *Building and Environment* 72 (2014), 343–355.
[2] Joaquin Amat Rodrigo and Javier Escobar Ortiz. 2023. skforecast. (11 2023). https://doi.org/10.5281/zenodo.8382788
[3] ASHARE. 2024. The American Society of Heating, Refrigerating and Air-Conditioning Engineers. (Jan 2024). https://www.ashrae.org
[4] Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. 2020. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*.
[5] 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.
[6] Gong Chen, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao, and Feng Zhao. 2008. Energy-Aware Server Provisioning and Load Dispatching for

Connection-Intensive Internet Services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design & Implementation (NSDI'08)*. San Francisco, CA, USA.

[7] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM. https://doi.org/10.1145/2939672.2939785

[8] Peng Cheng, Xianyuan Zhan, zhihao wu, Wenjia Zhang, Youfang Lin, Shou cheng Song, Han Wang, and Li Jiang. 2023. Look Beneath the Surface: Exploiting Fundamental Symmetry for Sample-Efficient Offline RL. In *Advances in Neural Information Processing Systems*, A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 7612–7631. https://proceedings.neurips.cc/paper_files/paper/2023/file/181a027913d36bc0a8857c0da661d621-Paper-Conference.pdf

[9] Paramveer S Dhillon, Dean P Foster, Sham M Kakade, and Lyle H Ungar. 2013. A risk comparison of ordinary least squares vs ridge regression. *The Journal of Machine Learning Research* 14, 1 (2013), 1505–1511.

[10] Haiyan Ding. 2024. Cluster data collected from production clusters in Alibaba for cluster management research. (Jan 2024). https://github.com/alibaba/clusterdata

[11] Envicool. 2024. (Jan 2024). https://www.envicool.net/

[12] Tom Bawden et al. 2016. Global Warming: Data Centres to Consume Three Times as Much Energy in Next Decade, Experts Warn. *The Independent* 23 (2016).

[13] Peter I. Frazier. 2018. A Tutorial on Bayesian Optimization. (2018). arXiv:stat.ML/1807.02811

[14] Carlucci Gaetano. 2024. CPU Load Generator. (Jan 2024). https://github.com/GaetanoCarlucci/CPULoadGenerator

[15] Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. 2018. Gpytorch: Blackbox matrix-matrix gaussian process inference with GPU acceleration. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 7587–7597.

[16] Rob J Hyndman and George Athanasopoulos. 2021. *Forecasting: Principles and Practice* (3rd ed.). OTexts, Melbourne, Australia. https://otexts.com/fpp3

[17] Hemant Kumar. 2024. Jobs. (Jan 2024). https://kubernetes.io/docs/concepts/workloads/controllers/job/

[18] Hemant Kumar. 2024. kubernetes/kubernetes: Production-Grade Container Scheduling and Management. (Jan 2024). https://github.com/kubernetes/kubernetes

[19] Alok Gautam Kumbhare, Reza Azimi, Ioannis Manousakis, Anand Bonde, Felipe Frujeri, Nithish Mahalingam, Pulkit A. Misra, Seyyed Ahmad Javadi, Bianca Schroeder, Marcus Fontoura, and Ricardo Bianchini. 2021. Prediction-Based Power Oversubscription in Cloud Platforms. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. USENIX Association, 473–487. https://www.usenix.org/conference/atc21/presentation/kumbhare

[20] Nevena Lazic, Craig Boutilier, Tyler Lu, Eehern Wong, Binz Roy, MK Ryu, and Greg Imwalle. 2018. Data Center Cooling Using Model-Predictive Control. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*.

[21] Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. 2019. Constrained Bayesian optimization with noisy experiments. (2019).

[22] Shaohong Li, Xi Wang, Xiao Zhang, Vasileios Kontorinis, Sreekumar Kodakara, David Lo, and Parthasarathy Ranganathan. 2020. Thunderbolt: Throughput-Optimized, Quality-of-Service-Aware Power Capping at Scale. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 1241–1255. https://www.usenix.org/conference/osdi20/presentation/li-shaohong

[23] Yuanlong Li, Yonggang Wen, Dacheng Tao, and Kyle Guan. 2019. Transforming Cooling Optimization for Green Data Center via Deep Reinforcement Learning. *IEEE Transactions on Cybernetics* 50, 5 (2019), 2002–2013.

[24] Chieh-Jan Liang, Jie Liu, Liqian Luo, Andreas Terzis, and Feng Zhao. 2009. RACNet: A High-Fidelity Data Center Sensing Network. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys 2009)*.

[25] Sifan Liu and Edgar Dobriban. 2020. Ridge Regression: Structure, Cross-Validation, and Sketching. (2020). arXiv:math.ST/1910.02373

[26] Gilles Louppe. 2015. Understanding Random Forests: From Theory to Practice. (2015). arXiv:stat.ML/1407.7502

[27] Qi Mao, Yong Xu, Jianqi Chen, Jie Chen, and Tryphon Georgiou. 2023. Classical Stability Margins by PID Control. (2023). arXiv:math.OC/2311.11460

[28] Meta. 2024. PyTorch. (Jan 2024). https://pytorch.org

[29] Takao Moriyama, Giovanni De Magistris, Michiaki Tatsubori, Tu-Hoa Pham, Asim Munawar, and Ryuki Tachibana. 2018. Reinforcement Learning Testbed for Power-Consumption Optimization. In *Methods and Applications for Modeling and Simulation of Complex Systems: 18th Asia Simulation Conference, AsiaSim 2018*. Springer, 45–59.

[30] Theodosis Moumiadis. 2024. Tier 4 Data Center Cooling System Design. (Jan 2024). http://moumiadis.blogspot.com/2019/03/tier-4-data-center-cooling-system-design.html

[31] Daniel Nelson. 2024. Get InfluxDB: #1 Ranked Time Series Database. (Jan 2024). https://www.influxdata.com/get-influxdb/

[32] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations*.

[33] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Martin Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, 85 (2011), 2825–2830.

[34] Yongyi Ran, Han Hu, Yonggang Wen, and Xin Zhou. 2022. Optimizing energy efficiency for data center via parameterized deep reinforcement learning. *IEEE Transactions on Services Computing* 16, 2 (2022), 1310–1323.

[35] Yongyi Ran, Han Hu, Xin Zhou, and Yonggang Wen. 2019. Deepee: Joint optimization of job scheduling and cooling control for data center energy efficiency using deep reinforcement learning. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 645–655.

[36] Yongyi Ran, Xin Zhou, Han Hu, and Yonggang Wen. 2022. Optimizing Data Center Energy Efficiency via Event-Driven Deep Reinforcement Learning. *IEEE Transactions on Services Computing* 16, 2 (2022), 1296–1309.

[37] Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning*. The MIT Press.

[38] Md. Shiblee, P. K. Kalra, and B. Chandra. 2009. Time Series Prediction with Multilayer Perceptron (MLP): A New Generalized Error Based Approach. In *Advances in Neuro-Information Processing*, Mario Köppen, Nikola Kasabov, and George Coghill (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 37–44.

[39] Skforecast. 2024. Skforecast: Probabilistic Forecasting. (Jan 2024). https://skforecast.org/0.11.0/user_guides/probabilistic-forecasting

[40] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2 (NIPS '12)*. Curran Associates Inc., 2951–2959.

[41] J. Stojkovic, N. Iliakopoulou, T. Xu, H. Franke, and J. Torrellas. 2024. EcoFaaS: Rethinking the Design of Serverless Environments for Energy Efficiency. In *Proceedings of the 51st International Symposium on Computer Architecture (ISCA)*. To Appear.

[42] Ruihang Wang, Zhiwei Cao, Xin Zhou, Yonggang Wen, and Rui Tan. 2023. Phyllis: Physics-Informed Lifelong Reinforcement Learning for Data Center Cooling Control. In *Proceedings of the 14th ACM International Conference on Future Energy Systems*. 114–126.

[43] Ruihang Wang, Xinyi Zhang, Xin Zhou, Yonggang Wen, and Rui Tan. 2022. Toward physics-guided safe deep reinforcement learning for green data center cooling control. In *2022 ACM/IEEE 13th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 159–169.

[44] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021), 22419–22430.

[45] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*.

[46] Chi Zhang, Sanmukh R. Kuppannagari, Rajgopal Kannan, and Viktor K. Prasanna. 2019. Building HVAC Scheduling Using Reinforcement Learning via Neural Network Based Model Approximation. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. ACM, 287–296.

[47] Lu Zhang, Chao Li, Xinkai Wang, Weiqi Feng, Zheng Yu, Quan Chen, Jingwen Leng, Minyi Guo, Pu Yang, and Shang Yue. 2023. First: Exploiting the multi-dimensional attributes of functions for power-aware serverless computing. In *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 864–874.

[48] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International Conference on Machine Learning*. PMLR, 27268–27286.

[49] Zhuangzhuang Zhou, Yanqi Zhang, and Christina Delimitrou. 2023. AQUATOPE: QoS-and-Uncertainty-Aware Resource Management for Multi-stage Serverless Workflows. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '23)*, Vol. 1. ACM, New York, NY, USA, 14. https://doi.org/10.1145/3567955.3567960