Kalibre: Knowledge-based Neural Surrogate Model Calibration for Data Center Digital Twins

Ruihang Wang
Nanyang Technological University, Singapore

Xin Zhou
Nanyang Technological University, Singapore

Linsen Dong
Nanyang Technological University, Singapore

Yonggang Wen
Nanyang Technological University, Singapore

Rui Tan
Nanyang Technological University, Singapore

Li Chen
Alibaba Inc.
China

Guan Wang
Alibaba Inc.
China

Feng Zeng
Alibaba Inc.
China

ABSTRACT

Computational fluid dynamics (CFD) model has been widely used for prototyping data centers. Evolving it to high-fidelity *digital twin* is desirable for the management and operations of large-scale data centers. Manually calibrating CFD model parameters to achieve twin-class fidelity by specially trained domain expert is tedious and labor-intensive. To reduce manual efforts, existing automatic calibration approaches developed for various computational models apply heuristics to search model configurations within an empirically defined parameter bound. However, in the context of CFD, each search step requires long-lasting CFD model's iterated solving, rendering these approaches impractical with increased model complexity. This paper presents Kalibre, a knowledge-based neural surrogate approach that performs CFD model calibration by iterating four key steps of i) training a neural surrogate model based on CFD-generated data, ii) finding the optimal parameters at the moment through neural surrogate retraining based on sensor-measured data, iii) configuring the found parameters back to the CFD model, and iv) validating the CFD model using sensor-measured data as the ground truth. Thus, the parameter search is offloaded to the neural surrogate which is ultra-faster than CFD model's iterated solving. To speed up the convergence of Kalibre, we integrate prior knowledge of the twinned data center's thermophysics into the neural surrogate design to improve its learning efficiency. With about five hours computation on a 32-core processor, Kalibre achieves mean absolute errors (MAEs) of 0.81°C and 0.75°C in calibrating two CFD models for two production data halls hosting thousands of servers each while requires fewer CFD solving processes than existing baseline approaches.

CCS CONCEPTS

• **Applied computing** → **Data centers**; • **Computing methodologies** → *Modeling methodologies*; *Neural networks*.

BuildSys '20, November 18–20, 2020, Virtual Event, Japan

ACM ISBN 978-1-4503-8061-4/20/11…$15.00
https://doi.org/10.1145/3408308.3427982

KEYWORDS

CFD, Data center, Digital twin, Knowledge-based neural net, Surrogate model

ACM Reference Format:
Ruihang Wang, Xin Zhou, Linsen Dong, Yonggang Wen, Rui Tan, Li Chen, Guan Wang, and Feng Zeng. 2020. Kalibre: Knowledge-based Neural Surrogate Model Calibration for Data Center Digital Twins. In *The 7th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '20), November 18–20, 2020, Virtual Event, Japan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3408308.3427982

1 INTRODUCTION

To meet the ever increasing cloud computing and storage demands, the scales of modern data centers have been continuously growing. According to a white paper from Cisco [2], the number of hyperscale data centers will double from 338 at the end of 2016 to 628 by 2021. The data centers' increases in size and complexity bring substantial challenges to the effective and efficient management of their supporting infrastructures for avoiding operational risks and reducing energy costs. Currently, data center infrastructure management (DCIM) system is a common tool that visualizes and monitors the infrastructure status based on the measurements collected from deployed sensors [32]. DCIM provides the operator with useful and important information for proper responses in case of abnormalities and failures. However, with the increases in system scale and complexity, it is important to extend DCIM to have accurate prediction capabilities. With such, the operator can perform various what-if analyses, such as whether the increase of certain temperature setpoints can improve the energy efficiency without causing server overheating.

We consider *digital twin* for the desired capability extension. Digital twin is a collection of integrated multi-physics, multi-scale, and probabilistic modeling and simulation techniques for as-built systems [25]. It aims to pursue high modeling accuracy for complex systems based on data from various sources, including sensors, prior models, and domain knowledge. The concept was early applied in the aerospace industry and is now attracting interest in smart manufacturing [21], cyber-physical systems [4], and smart city creation [17]. To build digital twins for data centers, various elementary techniques from multiple disciplines have existed to model the cyber-physical processes from the building level to the chip level. In particular, the computational fluid dynamics (CFD) modeling is a primary technique to characterize the thermodynamics in data centers [22]. It has been adopted in the offline analysis

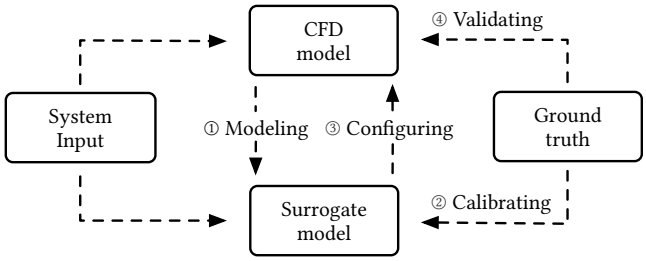

Figure 1: Kalibre operates by iterating ① training a CFD surrogate model, ② searching optimal parameters through surrogate retraining, ③ configuring the found parameters back to the CFD model, ④ validating the new configuration.

for energy cost reduction and risk management [23]. However, the accuracy of the CFD models in general does not reach the digital twin class for online operations. This is because the assumptions or simplifications made in the prototyping phase may lead to result distortions.

To evolve a CFD model into its digital twin form, it is important to instrument the model with sufficiently complete configuration of the physical infrastructure. A model with incomplete configuration may diverge from the ground truth. For example, as reported in [26, 31], a manually constructed CFD model can yield temperature prediction errors up to 5°C. Unfortunately, obtaining the complete system configuration often faces substantial challenges due to 1) the large number of parameters in the configuration and 2) the labor-intensive and error-prone manual calibration process for these parameters. For instance, each server in a data center may have its own characteristics of the passing-through air flow rate due to its internal fan control logic. However, such information is often not available in the server hardware's specification and can only be empirically estimated or manually collected via *in situ* measurement.

To achieve twin-class accuracy, automatic calibration of the difficult-to-obtain system configuration parameters will be necessary. However, this turns out to be a challenging task, due primarily to the ultra-high computation overhead of executing the CFD model to assess any candidate parameter configuration. The existing heuristic approaches (e.g., evolution strategies, genetic algorithms, simulated annealing, etc.) can be applied for automatic CFD calibration [12, 24]. These approaches in general require many search iterations, e.g., hundreds (cf. §5), to find good settings for the system configuration parameters. In each iteration, a CFD model solving is performed with the candidate configuration. When the CFD is built for a large-scale data hall with fine meshing granularity, the iterative search process incurs unacceptable computation times, since a single CFD model solving is already an hours-long or even days-long computing process of solving the Navier-Stokes formulation [5] using the finite element method. As such, the existing search-based approaches scale poorly with the size and granularity of the CFD model.

To advance automatic calibration, we propose Kalibre, a neural surrogate-assisted approach to calibrate data center CFD models with increasing scales and complexities. Kalibre avoids directly solving the CFD model for configuration search with the help of a trainable neural net. Fig. 1 illustrates Kalibre's workflow, in which the surrogate model iteratively updates the system configuration to minimize the CFD model's prediction errors by four key steps. ① The "coarse" surrogate is trained to align with the "fine" CFD model in the current system state locality by updating its internal weights based on CFD-generated data. ② The trained surrogate is re-optimized by updating the system configuration, which is also a part of trainable variables of the neural net, to maximize the consistency between the surrogate's predictions and the ground-truth sensor measurements. ③ The updated system configuration is set back to the CFD model for refining. ④ The ground-truth sensor measurements are used to validate the refined CFD model. Therefore, Kalibre offloads the fine-grained parameter configuration search to the surrogate. Vis-à-vis the existing heuristic approaches that solve the CFD model every configuration search step, Kalibre solves the CFD model much less frequently for merely providing feedback to the surrogate.

The implementation of Kalibre faces two challenges. First, the design of the surrogate to capture the complex thermophysics encompassed in the CFD model is challenging. Second, the training data for the neural surrogate is limited since generating such data using the CFD model is compute-intensive. Piecemeal solutions to address the above two challenges separately are contradictory, i.e., a deeper neural surrogate to well capture the complex thermodynamics requires more CFD-generated training data. To address the challenges, we design a neural surrogate architecture that integrates the prior knowledge of the thermal relations among a number of key variables in the twinned data hall. Compared with the vanilla neural net that approximates the CFD model as a black box, the introduction of the prior knowledge regulates the number of trainable variables and significantly improves the learning efficiency on small data.

We implement Kalibre and apply it to calibrate the CFD models of two production data halls sized hundreds of square meters that host thousands of servers, respectively. The CFD models calibrated by Kalibre achieve mean absolute errors (MAEs) of 0.81°C and 0.75°C in predicting the temperatures at tens of cold/hot aisle positions in each hall, respectively. The calibration process takes about 5 hours on a 32-core virtual machine in the cloud. In contrast, the heuristic configuration search and the vanilla neural net-based surrogate approach achieve MAEs of around 1.5∼4°C with the same computation time for calibration as Kalibre. We also invite a domain expert to manually fine-calibrate the two CFD models, yielding MAEs of 1.32°C and 1.1°C, which are higher than Kalibre's MAEs by 63% and 46%, respectively. The absolute MAE reductions of 0.51°C and 0.35°C achieved by Kalibre in comparison with the expert's manual calibration are significant in CFD modeling, due to the sharply increased difficulty in improving accuracy when the errors are already low (i.e., at around 1°C). The evaluation shows the effectiveness of Kalibre in automatically calibrating CFD models toward their digital twin forms.

Roadmap: The rest of this paper is structured as follows. §2 reviews related work. §3 formulates problem and overviews our approach. §4 and §5 design and evaluate Kalibre, respectively. §6 discusses several issues and concludes this paper.

2 RELATED WORK

This section reviews the relevant studies in data center modeling, surrogate-assisted optimization and knowledge-based neural nets.

Data center modeling: A variety of modeling techniques have been proposed for thermal management in data centers. They can be broadly categorized into law-based, data-driven, and hybrid models. The CFD models are representative law-based models, in that they capture the thermodynamic laws followed by the physical processes [22, 23, 26]. However, the CFD models are computationally expensive due to their internal recursive execution. As the mesh complexity increased for large-scale data centers, the CFD models solving times may increase from hours to days, introducing significant challenges for model calibration. An alternative is to use black-box data-driven models to learn a thermal map in the data center. For example, the Weatherman system [18] predicts the steady-state temperatures of certain server blocks using a neural net consisting of two hidden layers. In [33], a long short-term memory (LSTM) network is designed for predicting server CPU temperature. Although these data-driven models are fast and suitable for real-time use, they often perform poorly in the cases that are not covered by the training data. For instance, these models cannot well capture the thermal processes in case of cooling system failures, because the training data for such failure scenarios is generally lacking. Hybrid methods of combining CFD models and data-driven models have also been proposed. For instance, in [29], a psychrometric model is jointly used with three multilayer perceptrons to predict steady system state. In [10], the actual dataset is augmented with CFD-generated data for rare scenarios; the augmented dataset is used to train a linear regression model for temperature prediction. To ensure fidelity, the CFD model used in [10] is manually fine-calibrated by a human expert. As such, the approach is only evaluated on a small-size testbed and cannot scale well with the data center size.

Surrogate-assisted optimization: Surrogate-assisted optimization [15] speeds up the parametric optimization of those compute-intensive and non-differentiable models. It builds a lightweight surrogate of the original model and then uses the surrogate to guide the parameter search. This technique has been applied to building energy [19], hydrological [6], and aerodynamic [11] model optimization. The surrogate design is application-specific. For example, low-fidelity law-based surrogate model is built for full-fledged models in microwave engineering by aggressive space mapping [8]. Data-driven surrogate based on neural net is designed for high-dimensional and nonlinear coplanar waveguide model [31]. Response surface methodology based on radial basis function is studied for CFD model [20]. Among these studies, data-driven surrogates exhibit advantage in fast forwarding. However, the design of surrogate-assisted optimization faces a general challenge in well balancing the surrogate fidelity and the computation overhead of generating training data for surrogate via executing the original compute-intensive models. An effective approach to addressing the challenge is to improve the learning efficiency of the data-driven surrogate via its architectural design. Unfortunately, few studies are dedicated to pursuing surrogate's learning efficiency in the context of data center CFD.

Knowledge-based neural nets: Knowledge-based modeling incorporates empirical methods or first principles to improve model generalization. For neural nets, the knowledge can be any extra information about the modeled function beyond the function's inputs/outputs used as training samples [7]. Several studies have shown that the knowledge-based neural nets exhibit better extrapolation capabilities while require fewer training data, compared with vanilla neural nets. In [27], the neural net is trained by learning a loss function capturing a physical constraint expressed in closed form. This method is also applied in neural surrogate modeling for fluid flows without using any simulator-generated data [28].

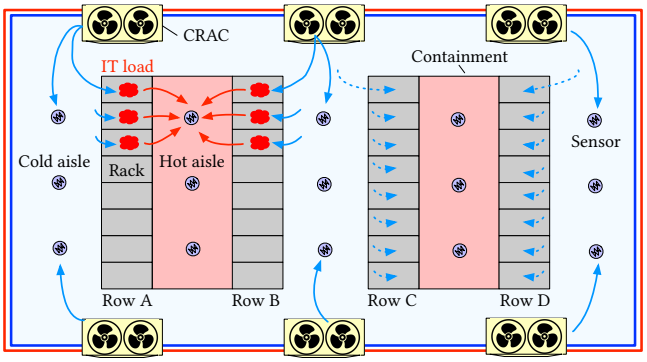

Figure 2: The layout of a typical data hall. Sensors are installed at the cold and hot aisles for cooling evaluation. Sensor measurements are mostly affected by the nearby CRACs.

This paper aims to develop a surrogate-assisted calibration approach for data center CFD models with high computational cost. We will advance the methodology of designing surrogate to improve its learning efficiency by incorporating first principles and prior knowledge of the modeled data hall. Therefore, we can achieve high calibration performance with much less CFD-generated training data for the surrogate. In addition, to the best of our knowledge, we are the first demonstrating the use of neural surrogate to calibrate industry-grade CFD models for large-scale data halls.

3 PROBLEM FORMULATION AND APPROACH OVERVIEW

In this section, we introduce the related background. Then, we formulate the problem and present the overview of our approach.

3.1 Background

CFD model can estimate the temperature and air velocity distributions in a given space by solving a simplified form of the Navier-Stokes equations [5]. For air-cooled data centers, CFD has been widely used during the prototyping phase for thermal and air flow analysis to avoid operational risks. To pursue higher efficiency of the cooling systems while not compromise the thermal safety of the computing and network equipment, it is desirable to improve the accuracy of the CFD model toward the paradigm of digital twin in the operational phase of data centers.

Fig. 2 illustrates the layout of a typical data hall, where *racks* hosting *servers* are assigned into multiple *rows* that separate *aisles*. These aisles alternate between cold and hot aisles. The *computer room air conditioning* units (CRACs) supply cold air to the servers

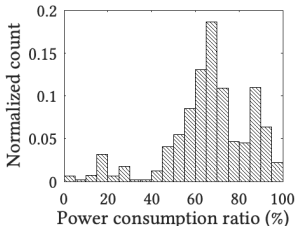

Figure 3: Server power consumption distribution

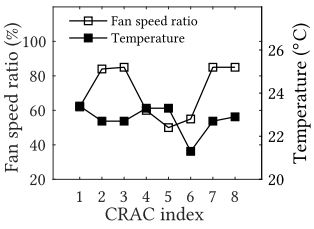

Figure 4: CRAC setpoints and fan speeds

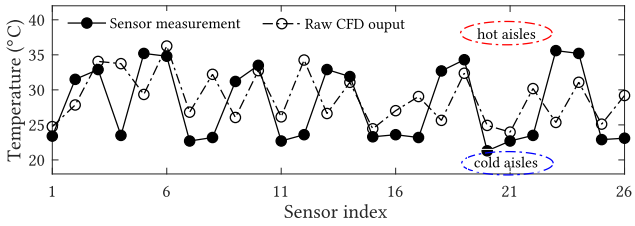

Figure 5: Sensor measurements and raw CFD temperature outputs at corresponding sensor locations.

Table 1: Summary of Notations.

Sym.	Definition	Sym.	Definition
$\|\| \cdot \|\|_2$	ℓ_2-norm	$\mathbf{e}$	sensor one-hot vector
$\otimes$	Element-wise product	$\mathbf{T}_\mathrm{c}$	setpoints vector
l	CRAC count	$\mathbf{V}$	fan speeds vector
m	server count	$\mathbf{P}$	powers vector
n	sensor count	$\boldsymbol{\alpha}$	flow rates vector
α_l, α_u	α lower and upper bound	$\mathbf{T}_\mathrm{s}$	measurements vector
$\mathcal{L}_1$	first loss function	$\widetilde{\mathbf{T}}_\mathrm{s}$	CFD results vector
$\mathcal{L}_2$	second loss function	$\widehat{\mathbf{T}}_\mathrm{s}$	surrogate results vector
$\mathbf{W}^{\mathrm{cs}}$	CRAC to sensor matrix	$\mathbf{W}^{\mathrm{ss}}$	server to sensor matrix

through the cold aisles and draw hot air from the hot aisles. To avoid air recirculation, containments are often implemented for the hot aisles. To evaluate the thermal condition in a data hall, the inlet and outlet temperatures of servers are often used as the key thermal variables. Therefore, temperature sensors are deployed in the cold and hot aisles to monitor such thermal variables. The inlet temperatures are often required to be in the range of 15°C to 27°C [13]. The outlet temperatures characterize the heat generated by the servers. Although the CFD model can predict the temperature at any location, we focus on the locations that are deployed with temperature sensors and thus have ground-truth temperature measurements for accuracy evaluation.

The servers in general have different characteristics in passing the cooling air through them. The characteristic highly depends on the server form factor and the control logics of the server's internal fans. Owing to the distinct characteristics, the servers will have different passing-through air flow rates in cubic feet per minute watt (cfm/W), where the cubic feet is for air volume, the minute is for time, and the watt is for the server power. The collection of the server air flow rates is part of the system configuration that greatly affects the thermodynamics of the data hall. Therefore, to achieve high CFD accuracy, the server air flow rates should be configured in the CFD model. Unfortunately, they are often unknown and hard to obtain. The manual *in situ* measurement using an air volume flow rate meter for each server is extremely labor intensive, especially for a large-scale data hall that hosts many types of servers. As a result, the server air flow rates are often empirically estimated by human expert. For a CFD model with many (e.g., thousands) servers, the rough settings of the server air flow rates could significantly downgrade the temperature prediction capability of the CFD model. The low accuracy will impede the use of CFD model for the desired fine-grained operational adjustment to pursue energy efficiency without causing thermal risk.

In this paper, we focus on devising an automatic approach to calibrate the server air flow rates configuration for data center CFD models on a steady system state. The approach can be also extended to include other parameters (e.g., by-pass air flow rates and recirculated air flow rates) into calibration. The system state consists of the following measurements: the setpoints and fan speeds of CRAC units, server powers and the temperatures measured in the hot and cold aisles. With the calibrated server air flow rates, the CFD model will yield more accurate temperature distribution prediction.

3.2 Problem Formulation

To formulate the calibration problem, we first define the relevant parameters in a data hall. Unless particularly specified, the notations used in this paper are summarized in Table 1. We consider a data hall hosting l CRACs, m servers, and n temperature sensors deployed in the cold and hot aisles, respectively.

Definition 1 (Input)**.** The input data for solving a CFD model is a vector consisting of all modeling parameters. Formally, the input $\mathbf{x} = (\mathbf{T}_\mathrm{c}, \mathbf{V}, \mathbf{P}, \boldsymbol{\alpha})$, where $\mathbf{T}_\mathrm{c} = (T_{\mathrm{c}1}, T_{\mathrm{c}2}, \ldots, T_{\mathrm{c}l})$, $\mathbf{V} = (V_1, V_2, \ldots, V_l)$, $\mathbf{P} = (P_1, P_2, \ldots, P_m)$, and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ are the vectors of CRAC setpoints, CRAC fan speeds, server powers, and server air flow rates, respectively.

Definition 2 (Output)**.** The output of CFD is a steady-state temperature and air velocity distribution map. For CFD model calibration, we focus on a set of results within the map at the locations installed with temperature sensors, which is denoted by $\widetilde{\mathbf{T}}_\mathrm{s} = (\tilde{T}_{\mathrm{s}1}, \tilde{T}_{\mathrm{s}2}, \ldots, \tilde{T}_{\mathrm{s}n})$.

Definition 3 (Measurement)**.** The measurement is a vector of real temperature values recorded by the physical sensors, which is denoted by $\mathbf{T}_\mathrm{s} = (T_{\mathrm{s}1}, T_{\mathrm{s}2}, \ldots, T_{\mathrm{s}n})$.

Let $|| \cdot ||_2$ denote the ℓ_2-norm of a vector. With the above definitions, the CFD model calibration aims to find the server air flow rate configuration that minimizes the ℓ_2-norm of the error vector between the model output and the measurement:

$$\boldsymbol{\alpha}^* \triangleq \arg\min_{\boldsymbol{\alpha}} ||\widetilde{\mathbf{T}}_\mathrm{s}(\mathbf{x}) - \mathbf{T}_\mathrm{s}||_2^2, \quad \text{s.t. } \alpha_\mathrm{l} \le \alpha_i \le \alpha_\mathrm{u}, i = 1, \ldots, m, \quad (1)$$

where $\boldsymbol{\alpha}^*$ is the vector of calibrated air flow rates. Each element in $\boldsymbol{\alpha}^*$ should be within an empirically estimated range $[\alpha_\mathrm{l}, \alpha_\mathrm{u}]$. The servers of the same type in general have the same air flow rate.

We now use an example in a real production data hall to illustrate the discrepancy of an uncalibrated CFD model and the actual sensor measurements. We first show a summary of the working conditions of the data hall. Fig. 3 shows a sample distribution of the servers' power consumption ratios at a time instant. We can see that most servers are working at approximately 60% of its maximum power. Fig. 4 is the CRAC setpoints and the corresponding fan speed ratios. Fig. 5 shows the temperature values measured by a number of sensors and uncalibrated CFD predictions on the locations of these sensors. For the sensor measurements, the cold aisle temperatures range from 20°C to 24°C, which are related to the CRAC setpoints and fan speed ratios. The hot aisle temperatures range from 30°C to 36°C, which are affected by the generated heat from the servers. The air flow rate of each server is empirically determined for the raw CFD model. With these initial configurations, the CFD model has temperature prediction errors from 2°C to 10°C. Such large errors disqualify the raw CFD model as a data center digital twin.

3.3 Approach Overview

Due to the high computational cost of CFD model solving, directly solving the optimization problem in Eq. (1) using search algorithms will incur unacceptable computation overhead. To address this issue, we design a surrogate model of the CFD model. Let $\widehat{\mathbf{T}}_\mathrm{s} \in \mathbb{R}^{1\times n}$ denote the temperature output vector of the surrogate model. Then, the problem in Eq. (1) is converted to a surrogate-assisted optimization that can be solved by iterating four consecutive steps. First, the surrogate model is trained to be locally aligned with the CFD model by minimizing the discrepancy between the surrogate's and the CFD's outputs:

$$\mathbf{W}^* \triangleq \arg\min_{\mathbf{W}} ||\widetilde{\mathbf{T}}_\mathrm{s}(\mathbf{x}) - \widehat{\mathbf{T}}_\mathrm{s}(\mathbf{W}, \mathbf{x})||_2^2, \tag{2}$$

where $\mathbf{W}$ is a set of trainable weights of the surrogate and $\mathbf{W}^*$ is the result of the surrogate training. Second, with $\mathbf{W}^*$, the surrogate is re-optimized through re-training such that the discrepancy between the surrogate's output and the measurement is minimized:

$$\boldsymbol{\alpha}^* \triangleq \arg\min_{\boldsymbol{\alpha}} ||\widehat{\mathbf{T}}_\mathrm{s}(\mathbf{W}^*, \mathbf{x}) - \mathbf{T}_\mathrm{s}||_2^2. \tag{3}$$

Third, the $\boldsymbol{\alpha}^*$ is configured into the CFD model. Finally, the CFD is validated based on sensor measurements. If the surrogate approaches to the CFD model, the $\boldsymbol{\alpha}^*$ after the convergence of the four-step iterations will approach to the one given by Eq. (1).

As discussed in §1, to address the challenges of the surrogate's complexity versus the needed volume of CFD-generated training data, we build a knowledge-based neural surrogate that can capture the physical layout and thermal relations among a number of key variables of the considered data hall. Specifically, we model a set of facilites (i.e., CRACs, servers, and sensors) in the considered hall as nodes and their connections as edges into a directed graph. The direction of an edge characterizes the thermal causality between the two end nodes of the edge. For example, an edge points from a CRAC node to a sensor node, because the supply air temperature of the CRAC affects the measured temperature of the sensor. The normalized reciprocal spatial distance between any two facilities will be used as the weight of the edge connecting the corresponding two nodes in the graph. This modeling approach follows the fact that the temperature measured by a sensor is mostly affected by the facilities in its neighborhood [16].

4 CFD CALIBRATION VIA SURROGATE

In this section, we present the design of the knowledge-based neural surrogate and Kalibre's iterative four-step model calibration.

4.1 Knowledge-based Neural Surrogate

The neural surrogate aims to approximate the complex thermophysics encompassed in the CFD model. In particular, its efficient training with a small amount of data generated from the CFD model is desirable, since the data generation requires intensive computation. Fig. 6 shows the proposed neural surrogate architecture. It consists of a *cooling block* and a *heating block*. The cooling block models the impact of the CRACs on the temperatures at all sensor locations; the heating block models the impact of the servers on the temperatures at the hot aisle sensor locations. Thus, the sum of the two blocks captures the effects from both CRACs and servers. The input of the model consists of the free variables of the data hall's steady state at a time instant, including CRAC temperature setpoints and fan speeds, and server powers. The server air flow rates are designated as trainable variables of the neural surrogate and initialized with rough estimates. Note that, as the neural surrogate is differentiable, the server air flow rates can be updated efficiently by backpropagation-based neural net training algorithms for the purpose of calibration. The output of the neural surrogate is a vector of n predicted temperatures at the sensor locations. In what follows, we present the designs of the cooling and heating blocks of the neural surrogate to capture prior knowledge of the thermal relations among the key variables. Lastly, we present the settings of the constants used by the neural surrogates, which are also based on the prior knowledge on the layout of the modeled data hall.

4.1.1 Cooling block. This block models the impact of the CRAC temperature setpoints $\mathbf{T}_\mathrm{c}$ and fan speeds $\mathbf{V}$ on the temperatures at all sensor locations. First, we encode the two free variables (i.e., $\mathbf{T}_\mathrm{c}$ and $\mathbf{V}$) into a hidden-layer variable for the k^{th} sensor as $X_k^{\text{cold}} = \sum_{i=1}^{l} T_{\mathrm{c}i} \cdot c_{ik}$, where T_{ci} is the setpoint of the i^{th} CRAC and c_{ik} is a cooling coefficient characterizing the impact of the i^{th} CRAC on the k^{th} sensor. We design c_{ik} to be positively related to the CRAC fan speed. Specifically, we use softmax activation to compute the *cooling coefficient matrix* as $c_{ik} = \frac{e^{z_{ik}}}{\sum_{a=1}^{l} e^{z_{ak}}}$, where z_{ik} is an intermediate variable defined by $z_{ik} = V_i \cdot W_{ik}^{\mathrm{cs}}$, V_i is the fan speed of the i^{th} CRAC, and W_{ik}^{cs} is a weight characterizing the thermal impact of the i^{th} CRAC on the k^{th} sensor. The CRAC-to-sensor matrix $\mathbf{W}^{\mathrm{cs}} \in \mathbb{R}^{n\times l}$ consisting of W_{ik}^{cs} for $i = 1, \ldots, l$ and $k = 1, \ldots, n$ is an adjacency matrix. The weights in this matrix can be fixed or trainable. If they are fixed, their settings are important and will be discussed in §4.1.3. §5 will compare the performance of the neural surrogates with $\mathbf{W}^{\mathrm{cs}}$ fixed or trainable. Lastly, we use a linear layer to project the hidden-layer variable to temperature as $T_k^{\text{cold}} = a_k X_k^{\text{cold}} + b_k$, where a_k and b_k are two trainable weights.

4.1.2 Heating block. This block models the impact of the servers on the temperatures at the hot aisle sensor locations. We assume

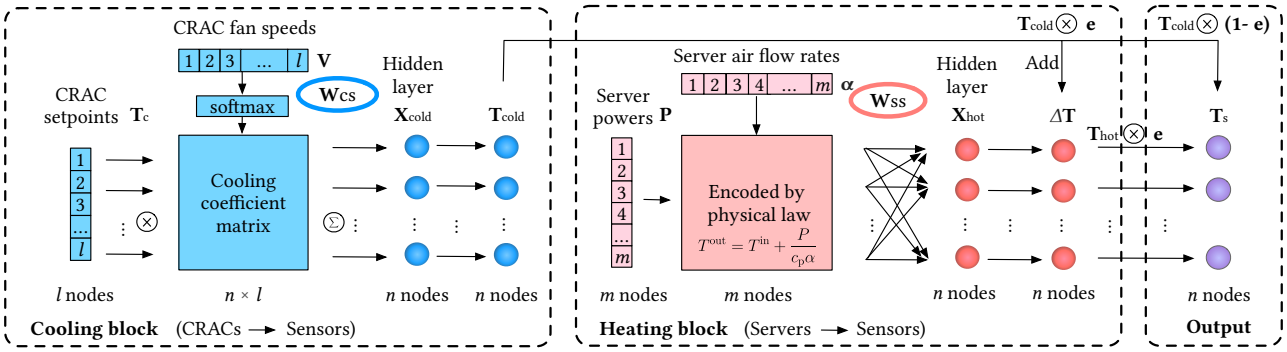

Figure 6: The architecture of the knowledge-based neural surrogate for temperature prediction. The structure consists of a *cooling block* and a *heating block*. The weight between any two facilities is initialized using their normalized reciprocal spatial distance. Two linear hidden layers are used to predict the cold aisle temperatures and temperature rises induced by servers.

that the energy dissipated from the servers in the forms of electromagnetic radiation and mechanical movements is negligible compared with that dissipated in the form of heat. Thus, from [13], the temperature increase caused by a server consuming P watts at its outlet can be modeled by $\frac{P}{c_p \alpha}$, where c_p is a heating constant representing the heat capacity of air and α is the server air flow rate. Based on this first principle, we use server powers and air flow rates to predict the server-induced temperature increase ΔT_k at the k^{th} hot aisle sensor location. Specifically, $\Delta T_k = c_k X_k^{\text{hot}} + d_k$, where c_k and d_k are two trainable weights, and X_k^{hot} is a hidden-layer variable. The X_k^{hot} is defined by $X_k^{\text{hot}} = \sum_{j=1}^{m} \frac{P_j}{\alpha_j} \cdot W_{jk}^{\text{ss}}$, where P_j is the j^{th} server power, α_j is the j^{th} server air flow rate, and W_{ij}^{ss} is the weight characterizing the thermal impact of the j^{th} server on the k^{th} sensor. We define the server-to-sensor adjacency matrix $\mathbf{W}^{\text{ss}}$ consisting of W_{jk}^{ss} for $j = 1, \ldots, m$ and $k = 1, \ldots, n$. Similar to $\mathbf{W}^{\text{cs}}$, $\mathbf{W}^{\text{ss}}$ can be fixed or trainable. For the former case, its setting is discussed in §4.1.3. Note that the heating block outputs ΔT_k for all sensor locations (denoted by $\Delta \mathbf{T}$); but only the outputs at hot aisle sensor locations will be used when combing the results of the cooling and heating blocks. This design simplifies the vectorized implementation of the neural surrogate using TensorFlow (cf. §4.3).

4.1.3 Joining two blocks and adjacency matrices settings. With hot-aisle containment and blanket, heat recirculation is negligible. Thus, the temperatures at cold aisle sensor locations are mainly affected by the CRACs; the temperatures at hot aisle sensor locations are jointly affected by the CRACs and servers. To combine the outputs of the cooling and heating blocks, we define a one-hot vector $\mathbf{e} \in \{0, 1\}^n$, where its element $e_k = 1$ or 0 represents that the k^{th} sensor location is in hot or cold aisle, respectively. Therefore, the final output of the neural surrogate, i.e., the temperatures at all sensor locations, can be expressed by $\widehat{\mathbf{T}}_s = \mathbf{T}_{\text{cold}} \otimes (\mathbf{1} - \mathbf{e}) + \mathbf{T}_{\text{cold}} \otimes \mathbf{e} + \Delta \mathbf{T} \otimes (\mathbf{1} - \mathbf{e})$, where $\otimes$ represents element-wise product, $\mathbf{T}_{\text{cold}} \otimes (\mathbf{1} - \mathbf{e})$ gives the temperatures at the cold aisle sensor locations, and $\mathbf{T}_{\text{cold}} \otimes \mathbf{e} + \Delta \mathbf{T} \otimes (\mathbf{1} - \mathbf{e})$ gives the temperatures at the hot aisle sensor locations.

If $\mathbf{W}^{\text{cs}}$ and $\mathbf{W}^{\text{ss}}$ are fixed, the weights of the neural surrogate are $\mathbf{W} = \{a_k, b_k, c_k, d_k | k = 1, \ldots, n\}$; otherwise, $\mathbf{W}$ additionally include $\mathbf{W}^{\text{cs}}$ and $\mathbf{W}^{\text{ss}}$. We now discuss the settings of $\mathbf{W}^{\text{cs}}$ and $\mathbf{W}^{\text{ss}}$ if they are not trainable. Each of their elements represents the thermal impact of a facility (CRAC or server) on a sensor location. Since the thermal impact decreases with spatial distance, in this paper, we set it to be a normalized reciprocal of the spatial distance between the facility and the sensor location. When it is lower than a threshold, it is forced to be zero, indicating that the corresponding thermal impact is negligible. Thus, to set these two two matrices, the layout of the data hall and the sensor locations will be needed, which are available to the data center operator in general.

4.2 Four-step Iterations for CFD Calibration

Let $\widehat{T}_{sk}$, $\widetilde{T}_{sk}$, T_{sk} denote the surrogate-predicted temperature, CFD-predicted temperature, and the measured temperature at the location of the k^{th} sensor, respectively. Algorithm 1 shows the pseudocode of the four-step iterations. We now explain it in detail.

① **Neural surrogate training** (Line 4-6): The training data is generated by solving the CFD model with collected system input (including CRAC temperature setpoints and fan speeds, server powers) and the initial $\boldsymbol{\alpha}$ or calibrated $\boldsymbol{\alpha}$ by step ③ of the previous iterations to yield the predicted temperatures at sensor locations. The detailed training data generation is described in §5.2.2. Note that each element of $\boldsymbol{\alpha}$ should be within $[\alpha_l, \alpha_u]$. The system input, the $\boldsymbol{\alpha}$, and the predicted temperatures form a new training data sample that is added to the training dataset accumulated from the first iteration. With the training dataset, the neural surrogate is updated to minimize the errors between its predicted temperatures and the CFD-predicted temperatures of the training samples. Thus, the weights of the neural surrogate are updated using the gradient of the least squares loss function of $\mathcal{L}_1 = \frac{1}{n} \sum_{k=1}^{n} (\widehat{T}_{sk}(\mathbf{W}, \mathbf{x}) - \widetilde{T}_{sk}(\mathbf{x}))^2$. As a result, the surrogate is trained to align with the CFD model. At the end of this step, $\mathbf{W}$ is frozen.

② **Surrogate-assisted calibration** (Line 7-8): The surrogate is re-optimized to minimize the errors between its predicted temperatures and the measured temperatures by updating $\boldsymbol{\alpha}$. In this step, $\boldsymbol{\alpha}$ is set trainable. An empirical regularization term is added to penalize the loss function if the temperature difference between the hot and cold aisle, i.e., ΔT, is out of the empirical range $[\Delta T_l, \Delta T_u]$. The penalty term is expressed using the rectified linear units (ReLU) as $h(T) = \sum_{j=1}^{m} (\text{ReLU}(\Delta T_l - \Delta T) + \text{ReLU}(\Delta T - \Delta T_u)) \times P_j$, where P_j is

Algorithm 1 Kalibre's CFD model calibration procedure.

Input: Measurements collected from a data hall at a time instant, including CRAC setpoints $\mathbf{T}_c$, CRAC fan speed ratios $\mathbf{V}$, server powers $\mathbf{P}$, and sensor measurements $\mathbf{T}_s$. Initial server air flow rates $\boldsymbol{\alpha}$. Cooling and heating coefficient matrix $\mathbf{W}^{cs}$ and $\mathbf{W}^{ss}$.
Output: Calibrated $\boldsymbol{\alpha}$.

1: Initialize each α within $[\alpha_l, \alpha_u]$ and CFD error ϵ;
2: Assign initial configurations to the surrogate graph $\mathcal{G}$;
3: **for** $i = 1$: Max iteration **do**
4: Solve CFD model to obtain $\widetilde{\mathbf{T}}_s$;
5: Aggregate CFD solving results as training data;
6: Train surrogate by performing gradient descent on $\mathcal{L}_1$;
7: Search $\boldsymbol{\alpha}$ by performing differential evolution;
8: Search $\boldsymbol{\alpha}$ by performing gradient descent on $\mathcal{L}_2$;
9: Configure $\boldsymbol{\alpha}$ to the CFD model;
10: **if** $\frac{1}{n}\sum_i^n |\widetilde{T}_{si} - T_{si}| < \epsilon$ **then**
11: $\epsilon \leftarrow \frac{1}{n}\sum_i^n |\widetilde{T}_{si} - T_{si}|$; $\quad \boldsymbol{\alpha}^* \leftarrow \boldsymbol{\alpha}$;
12: **end if**
13: **end for**
14: **return** Calibrated server air flow rate configurations $\boldsymbol{\alpha}^*$;

the j^{th} server power. The term means that, if the server power is higher, the penalty should be more significant. Thus, the second loss function with regularization is $\mathcal{L}_2 = \frac{1}{n}\sum_{k=1}^n (\widehat{T}_{sk}(\boldsymbol{\alpha}) - T_{sk})^2 + \frac{\lambda}{n}\sum_{k=1}^n h(T)$, where λ is a regularization coefficient. In our experiments, we set ΔT_l and ΔT_u to be 5°C and 15°C, based on the data center operator's experience. To accelerate the re-optimization, we implement a hybrid approach of combining differential evolution algorithm with gradient backpropagation to minimize the loss function $\mathcal{L}_2$. This hybrid approach has been shown effective in accelerating neural net training [30]. We will also evaluate its effectiveness for our specific problem in §5.2.2.

③ **CFD configuration** (Line 9): The updated $\boldsymbol{\alpha}$ is configured back to the CFD model. The refined CFD model is then used for step ① of the next iteration.

④ **CFD validation** (Line 10-12): The CFD model's accuracy is validated against the ground-truth sensor measurements. Only better $\boldsymbol{\alpha}$ is recorded for final output candidate.

Through iterative optimization of the two loss functions $\mathcal{L}_1$ and $\mathcal{L}_2$, the $\boldsymbol{\alpha}$ will be calibrated to improve the CFD model's accuracy.

4.3 Implementation of Kalibre

We implement Kalibre with Python 3.5 and Google TensorFlow 1.15.0, where the latter is a library widely used for building machine learning applications. When we use TensorFlow to build the neural surrogate's computational graph, the server air flow rates $\boldsymbol{\alpha}$ are set as a vector of trainable variables instead of a TensorFlow placeholder. This allows us to control their updating by choosing to freeze the gradients or not. We choose Adam [14] as the optimizer, which is a method for efficient stochastic optimization that only requires first-order gradients and little memory space. The CFD model solving is performed by 6SigmaDCX [1], a commercial CFD software package. The 6SigmaDCX can load $\boldsymbol{\alpha}$ from a configuration file. During the four-step iterations, our Python program writes the candidate $\boldsymbol{\alpha}$ to the file, invokes a 6SigmaDCX session to solve the CFD model, and collects results by parsing 6SigmaDCX's output.

Table 2: CFD model solving time.

CPU cores	1	2	4	8	16	32
Solving time (h)	5.95	3.72	2.54	0.99	0.6	0.44

Table 3: Hyperparameter settings of Kalibre.

Hyperparameter	Setting	Hyperparameter	Setting
$[\alpha_l, \alpha_u]$ (cfm/W)	[0.01, 3]	$[\Delta T_l, \Delta T_u]$ (°C)	[5, 15]
Augment batch size	16	Training epoch	150
Initial learning rate	0.1	Regularization term	1
Decay coefficient	0.8	Max search iteration	100
Population size	10	Crossover rate	0.6

5 PERFORMANCE EVALUATION

In this section, we apply Kalibre to calibrate the CFD models built for two production data halls and present the evaluation results against other baseline approaches.

5.1 Experiment Methodology and Settings

5.1.1 Data halls and CFD models. Our targets are two production data halls (referred to as Hall A and Hall B) in operation for e-commerce applications. Both of them are sized hundreds of square meters that host thousands of servers, respectively (the details of the two data halls are omitted here due to confidentiality requirement). Their CFD models were built and meshed with 10 million grid cells by a domain expert using 6SigmaDCX. The accuracy of these two CFD models will be evaluated in §5.2.2. Here, we present their compute overheads. Table 2 shows the compute times for solving one of the CFD models when the number of used CPU cores varies. Note that the 6SigmaDCX software package made available to us supports parallel computing with up to 32 CPU cores on the same computer. The evaluation shows that a single CFD model solving takes up to several hours and the solving time decreases with the number of used CPU cores. However, the model solving speed (i.e., the reciprocal of the solving time) is sub-linear to the number of used CPU cores. This suggests that the CFD computation is not completely divisible and the communications among the paralleled units matter. Thus, even if the 32-core limit is lifted, the attempt to use more CPU cores across multiple computers may face performance bottlenecks due to the cross-computer communication overheads. With 32 CPU cores, the CFD model solving time is about half an hour. This solving time still renders the heuristic search-based model calibration approaches impractical, since they generally need a large number of iterations (e.g., hundreds as shown shortly). Note that GPU acceleration has been introduced to another commercial CFD software package [3]. However, it brings 3.7x acceleration only [3], which does not change the impracticality of the heuristic search-based model calibration approaches.

5.1.2 Error metric and settings. We use *mean absolute error* (MAE) to measure the errors of the CFD models in temperature prediction.

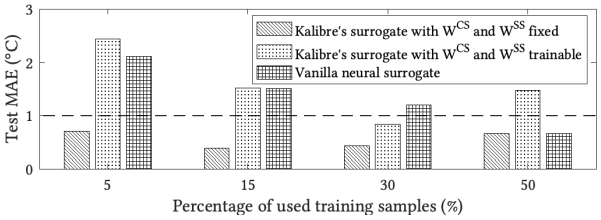

Figure 7: MAEs of three neural surrogates trained with 5%, 15%, 30%, and 50% of samples in the training dataset.

Specifically, $MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$, where N is the number of deployed sensors, y_i and $\hat{y}_i$ are the i^{th} sensor measurement and the prediction made by the CFD, respectively. Table 3 shows the hyperparameter settings of Kalibre. These settings include the empirical bounds and the hyperparameters of the surrogate training and differential evolution search. They are selected based on advice from the domain expert or extensive experimental tests.

5.2 Evaluation Results

5.2.1 Performance of neural surrogates. As the neural surrogate's efficiency in learning from small data is a key merit, we conduct experiments to investigate the impact of training data volume on the accuracy of the neural surrogate. We consider three designs of the neural surrogate: Kalibre's neural surrogate with $\mathbf{W}^{\text{cs}}$ and $\mathbf{W}^{\text{ss}}$ fixed and trainable, respectively, and a vanilla neural surrogate. The vanilla neural surrogate has three fully-connected layers consisting of 518, 128, and 32 neurons. Before the experiments, we solve Hall A's CFD model to generate 213 training data samples. Each model solving is based on an $\boldsymbol{\alpha}$ with each of its element sampled randomly and uniformly from $[\alpha_{\text{l}}, \alpha_{\text{u}}]$. Then, we divide the generated data samples into training and test datasets following a 8:2 ratio. Fig. 7 shows the MAEs measured on the test dataset when the three neural surrogates are trained using 5%, 15%, 30%, and 50% samples of the training dataset. First, we compare Kalibre's neural surrogates with $\mathbf{W}^{\text{cs}}$ and $\mathbf{W}^{\text{ss}}$ fixed and trainable. We can see that for all amounts of used training data, the neural surrogate with $\mathbf{W}^{\text{cs}}$ and $\mathbf{W}^{\text{ss}}$ fixed and initialized with spatial distance reciprocals outperforms the others. This is because when the two adjacency matrices are given, a, b, c and d are the only parameter sets that we need to learn, i.e. $\mathbf{W}=\{a, b, c, d\} \in \mathbb{R}^{4n}$. In contrast, the neural surrogate with $\mathbf{W}^{\text{cs}}$ and $\mathbf{W}^{\text{ss}}$ trainable has $(l \times n + m \times n)$ more weights to be learned, which require more training samples to avoid overfitting. Thus, in the rest of this paper, we fix $\mathbf{W}^{\text{cs}}$ and $\mathbf{W}^{\text{ss}}$ at their initial settings based on spatial distance reciprocals. Second, we examine the results of the vanilla neural surrogate. From Fig. 7, when 5% training samples are used, the vanilla neural surrogate produces 2.11°C MAE, compared with Kalibre neural surrogate's 0.69°C MAE. Although the vanilla neural surrogate's MAE decreases with the amount of used training data and eventually achieves comparable MAE as Kalibre's neural surrogate when 50% training samples are used, the results clearly suggest that the vanilla neural surrogate has lower learning efficiency on small data. This is consistent with our understanding since the vanilla neural surrogate has thousands trainable weights and thus needs more training data to avoid overfitting.

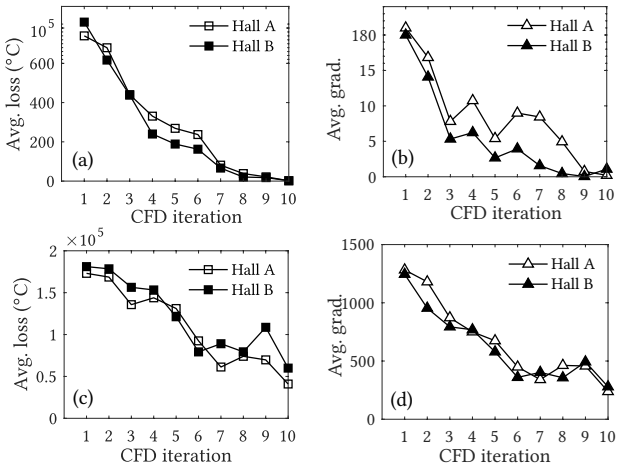

Figure 8: Average loss and gradient over Kalibre's iterations. (a)-(b): with differential evolution; note that the y-axis scale is not uniform. (c)-(d): without differential evolution.

5.2.2 Convergence and effectiveness of Kalibre. In this set of experiments, we evaluate Kalibre's convergence speed and the effectiveness of its calibration. To initiate Kalibre's four-step iterations, we solve the CFD model to generate three training data samples by setting each element of $\boldsymbol{\alpha}$ to the upper and lower bounds, as well as a mid point between the two bounds. To mitigate the initial overfitting, we augment the three-sample dataset by adding Gaussian noises as in [9]. The augmented batch size for each sample is 16. Thus, we have a total of 48 samples for the initial training process. In each four-step iteration, a new training data sample will be generated by solving the CFD model configured with the $\boldsymbol{\alpha}^*$ found by the neural surrogate. This new training data sample is aggregated to the training dataset. In this set of experiment, Kalibre terminates after ten iterations. As presented in §4.2, Kalibre adopts a hybrid approach combining the gradient backpropagation widely used for neural net training and the differential evolution to find $\boldsymbol{\alpha}^*$. In our experiments, the gradient backpropagation is implemented by the Adam optimizer. Fig. 8(a) and (b) show the average loss and gradient over the four-step iterations. In the first iteration, the average loss and gradient are very large, reaching around 10^5 and 180, respectively. A closer examination shows that the regularization penalty of the loss function $\mathcal{L}_2$ is large in the very early iterations. However, in the subsequent iterations, the average loss sharply decreases and converges to zero in the tenth iteration. The average gradient also approaches zero. For comparison, we adopt a baseline of using the Adam optimizer only to find $\boldsymbol{\alpha}^*$. Figs. 9(c) and (d) show the results of this baseline. We can see that the convergence is slow and the average loss remains large (about 0.5×10^5) after ten iterations. The results show that the differential evolution effectively accelerates the convergence of Kalibre.

Then, we show the effectiveness of the model calibration. Figs. 9(a) and 10(a) show the two halls' thermal planes computed based on the original CFD models presented in §5.1.1. We can see that the temperature distribution is uneven in both the cold and hot aisles. Figs. 9(c) and 10(c) shows the temperatures predicted by the two

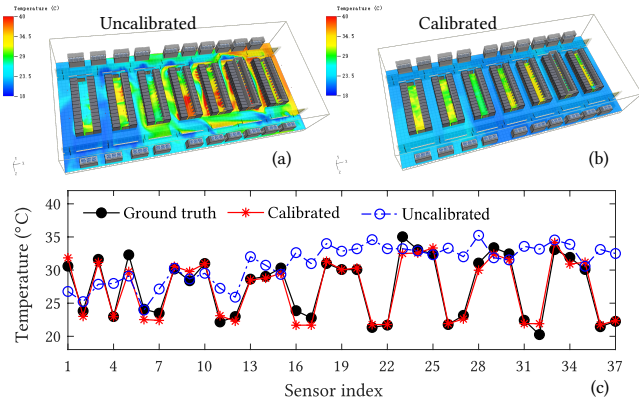

Figure 9: Hall A temperature distribution. (a) Thermal plane produced by the original CFD model; (b) Thermal plane produced by the calibrated CFD model; (c) CFD-predicted and ground-truth temperatures at the sensor locations.

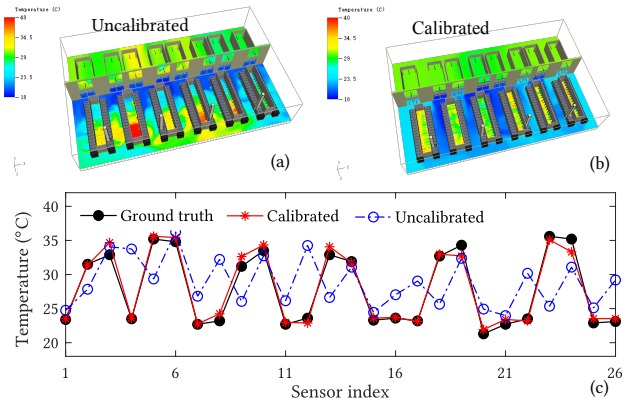

Figure 10: Hall B temperature distribution. (a) Thermal plane produced by original CFD model; (b) Thermal plane produced by calibrated CFD model; (c) CFD-predicted and ground-truth temperatures at the sensor locations.

halls' original CFD models at the sensor locations and the ground-truth values measured by the sensors. The original CFD models' prediction errors are from 3°C to 6°C. Such large errors are due to the inaccurate estimation of the server air flow rates. Figs. 9(b) and 10 (b) show the thermal planes computed based on the CFD models after ten calibration iterations of Kalibre. We can see that the temperature distributions become more uniform, compared with the results shown in Figs. 9(a) and Figs. 9(b). From Figs. 9(c) and 10(c), the temperatures predicted by the calibrated CFD models well match the ground-truth values, with MAEs of 0.81°C and 0.75°C for the two halls, respectively. The above results show the effectiveness of Kalibre for large-scale data halls.

5.2.3 Comparison with baseline approaches. We compare Kalibre with three baseline approaches discussed in §1:

Manual calibration involves extensive tuning of the server air flow rates by a CFD expert with years of experience. Specifically, if the CFD-predicted temperature at a sensor location is higher than the ground-truth temperature, the expert empirically increases the flow rates of nearby servers and vice versa.

Heuristic parameter search uses the covariance matrix adaptation evolution strategy (CMA-ES) to search good $\boldsymbol{\alpha}$. It solves the CFD model every search iteration. CMA-ES is a gradient-free numerical optimization method. It applies the (1+1) strategy described in [24] to generate one candidate solution per iteration. If the MAE of the new offspring is smaller, it becomes the parent. The mutation rate is set to $\sigma = 5$ and updated for each iteration by following the 1/5 successful evolution rule described in [24].

Vanilla neural surrogate uses the neural net presented in §5.1.1 that consists of three fully-connected layers. It also follows Kalibre's four-step iterations to perform the model calibration.

Table 4 shows the MAEs achieved by different approaches with ten calibration iterations, as well as the lowest MAEs achieved and the needed iterations. With ten calibration iterations, Kalibre achieves lower MAEs compared with the baseline approaches for both halls. After about 15 calibration iterations, Kalibre's MAEs converge to 0.76°C and 0.59°C for the two halls. As shown in §5.1.1, the vanilla neural surrogate requires more training data samples to well represent the CFD model. Thus, with ten calibration iterations, its calibrated CFDs still yield MAEs higher than manual calibration. The heuristic parameter search cannot find good configurations with the same CFD iteration times. Its MAEs saturate at high levels of 3.49°C and 2.61°C even after 120 CFD model solving processes for the two halls, respectively. Thus, we can see that the heuristic parameter search and vanilla neural surrogate are inefficient to find the optimal configuration under the same computation time. Although the manual calibration reduces the MAE to 1.32°C to 1.1°C, it is labor-intensive. In addition, its MAEs are higher than Kalibre's. The lower MAEs achieved by Kalibre further improves the fidelity of the CFD results. If such results are used to guide data center operations, the risks caused by the errors can be further reduced. In sum, systematic approaches to improve the fidelity of data center digital twin are always desirable.

5.2.4 Compute time. From the results in Table 4, Kalibre achieves sub-1°C MAEs with 10 calibration iterations and the lowest MAEs with about 15 calibration iterations. The compute time breakdown of each iteration is as follows: about 200 seconds for surrogate training, about 100 seconds for configuration search, about 26 minutes for CFD model solving with 32 CPU cores. Kalibre's compute time for calibrating a hall's CFD is about 5 hours and 7.5 hours for 10 and 15 iterations, respectively. For vanilla neural net to reach similar accuracy, it requires more iterations, resulting in 3x~5x compute time compared with Kalibre's to generate enough data for training.

6 DISCUSSIONS AND CONCLUSION

We now discuss several worth-noting issues. First, the primary purpose of the surrogate is to improve the efficiency of parameter search. The surrogate does not provide a full-fledged approximation of the CFD model. For instance, it does not model the temperatures at the locations without sensors, which are modeled by the CFD model in contrast. Thus, only the calibrated CFD model shall be

Table 4: MAE achieved with 10 calibration iterations, as well as the lowest MAE achieved and the needed iterations.

	Approach	MAE (°C) 10 iters	MAE (°C) lowest	Needed iters	Auto?
Hall A	Manual	1.32	N/A	N/A	✗
	Heuristic	4.75	3.49	127	✓
	Vanilla	1.44	1.09	50	✓
	Kalibre	**0.81**	**0.76**	**14**	✓
Hall B	Manual	1.1	N/A	N/A	✗
	Heuristic	3.80	2.61	130	✓
	Vanilla	1.33	0.80	28	✓
	Kalibre	**0.75**	**0.59**	**15**	✓

used as a digital twin for the run-time temperature evaluation in improving energy efficiency and reducing operational risks. Second, the surrogate architecture described in this paper is for data halls with hot-aisle containment. Thus, heat recirculation is not considered. To address the data halls without hot-aisle containment, heat recirculation and temperature mixing effects should be added to the neural surrogate's design. Third, this paper mainly focuses on temperature prediction. For other types of prediction, Kalibre can be extended to address their calibration problems with proper surrogate designs. For instance, if air flow rate sensors are deployed, Kalibre can be extended to calibrate the CFD for predicting air velocity distribution.

In conclusion, this paper presents Kalibre, an automatic surrogate-based approach to calibrating data center CFD models. The design of Kalibre's neural surrogate integrates prior knowledge including the thermal relations among the key variables of the physical infrastructure. Thus, it reduces the demand on the amount of training data generated by the compute-intensive CFD model solving. We demonstrate its effectiveness on two CFD models built for two production data halls that host thousands of servers. The CFD models calibrated by Kalibre achieve temperature prediction MAEs of 0.81°C and 0.75°C, respectively. Compared with manual calibration, Kalibre's improvement of up to 0.5°C is significant in CFD modeling due to the sharply increased difficulty in improving accuracy when the errors are already low (i.e., at around 1°C). Kalibre sheds lights on the calibration of other compute-intensive models to pursue high accuracy in approximating complex physical processes.

ACKNOWLEDGMENTS

This work is funded by National Research Foundation (NRF) via the Green Data Centre Research (GDCR) and the Green Buildings Innovation Cluster (GBIC), administered by Info-communications Media Development Authority (IMDA) and Building and Construction Authority (BCA) respectively.

REFERENCES

[1] [n.d.]. 6SigmaDCX. https://www.futurefacilities.com
[2] [n.d.]. Cisco Global Cloud Index: Forecast and Methodology, 2016–2021 White Paper. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html
[3] [n.d.]. GPU-accelerated Ansys Fluent. https://www.nvidia.com/en-sg/data-center/gpu-accelerated-applications/ansys-fluent/
[4] Kazi Masudul Alam and Abdulmotaleb El Saddik. 2017. C2PS: A digital twin architecture reference model for the cloud-based cyber-physical systems. *IEEE Access* 5 (2017), 2050–2062.
[5] John David Anderson and J Wendt. 1995. *Computational fluid dynamics*. Vol. 206. Springer.
[6] Michael J Asher, Barry FW Croke, Anthony J Jakeman, and Luk JM Peeters. 2015. A review of surrogate models and their application to groundwater modeling. *Water Resour. Res.* 51, 8 (2015), 5957–5973.
[7] John W Bandler, Radoslaw M Biernacki, Shao Hua Chen, Piotr A Grobelny, and Ronald H Hemmers. 1994. Space mapping technique for electromagnetic optimization. *IEEE Trans. Microw. Theory Tech.* 42, 12 (1994), 2536–2544.
[8] John W Bandler, Radoslaw M Biernacki, Shao Hua Chen, Ronald H Hemmers, and Kaj Madsen. 1995. Electromagnetic optimization exploiting aggressive space mapping. *IEEE Trans. Microw. Theory Tech.* 43, 12 (1995), 2874–2882.
[9] Chris M. Bishop. 1995. Training with Noise is Equivalent to Tikhonov Regularization. *Neural Comput.* 7, 1 (1995), 108–116.
[10] Jinzhu Chen, Rui Tan, Yu Wang, Guoliang Xing, Xiaorui Wang, Xiaodong Wang, Bill Punch, and Dirk Colbry. 2012. A high-fidelity temperature distribution forecasting system for data centers. In *2012 IEEE 33rd Real-Time Systems Symposium*.
[11] Zhonghua Han, Chenzhou Xu, Liang Zhang, Yu Zhang, Keshi Zhang, and Wenping Song. 2019. Efficient aerodynamic shape optimization using variable-fidelity surrogate models and multilevel computational grids. *Chin. J. Aeronaut.* (2019).
[12] John H Holland. 1992. Genetic algorithms. *Sci.Am.* 267, 1 (1992), 66–73.
[13] M. Jonas, R. R. Gilbert, J. Ferguson, G. Varsamopoulos, and S. K. S. Gupta. 2012. A transient model for data center thermal prediction. In *IGCC*. 1–10.
[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[15] Slawomir Koziel and Leifur Leifsson. 2012. Surrogate-based aerodynamic shape optimization by variable-resolution models. *AIAA J.* 51, 1 (2012), 94–106.
[16] 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*. 3814–3823.
[17] Neda Mohammadi and John E Taylor. 2017. Smart city digital twins. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. 1–5.
[18] Justin Moore, Jeffrey S Chase, and Parthasarathy Ranganathan. 2006. Weatherman: Automated, online and predictive thermal mapping and management for data centers. In *2006 IEEE international conference on Autonomic Computing*.
[19] Shreshth Nagpal, Caitlin Mueller, Arfa Aijazi, and Christoph F Reinhart. 2019. A methodology for auto-calibrating urban building energy models using surrogate modeling techniques. *J. of Build. Perform. Simul.* 12, 1 (2019), 1–16.
[20] Long Phan and Cheng-Xian Lin. 2019. CFD-based response surface methodology for rapid thermal simulation and optimal design of data centers. *Advances in Building Energy Research* (2019), 1–23.
[21] Qinglin Qi and Fei Tao. 2018. Digital twin and big data towards smart manufacturing and industry 4.0: 360 degree comparison. *IEEE Access* 6 (2018), 3585–3593.
[22] Amir Radmehr, Brendan Noll, John Fitzpatrick, and Kailash Karki. 2013. CFD modeling of an existing raised-floor data center. In *29th IEEE Semiconductor Thermal Measurement and Management Symposium*. 39–44.
[23] 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 *ICDCS*. 645–655.
[24] Ingo Rechenberg. 1973. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution, frommann–holzboog.
[25] Mike Shafto, Mike Conroy, Rich Doyle, Ed Glaessgen, Chris Kemp, Jacqueline LeMoigne, and Lui Wang. 2010. Draft modeling, simulation, information technology & processing roadmap. *Technol. Area* 11 (2010).
[26] Umesh Singh, Amarendra Singh, S Parvez, and Anand Sivasubramaniam. 2010. CFD-based operational thermal efficiency improvement of a production data center. In *SustainIT*.
[27] Russell Stewart and Stefano Ermon. 2017. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*.
[28] Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. 2020. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering* 361 (2020), 112732.
[29] Duc Van Le, Yingbo Liu, Rongrong Wang, Rui Tan, Yew-Wah Wong, and Yonggang Wen. 2019. Control of Air Free-Cooled Data Centers in Tropics via Deep Reinforcement Learning. In *BuildSys*. 306–315.
[30] Lin Wang, Yi Zeng, and Tao Chen. 2015. Back propagation neural network with adaptive differential evolution algorithm for time series forecasting. *Expert Syst. Appl.* 42, 2 (2015), 855–863.
[31] Paul M Watson and Kuldip C Gupta. 1997. Design and optimization of CPW circuits using EM-ANN models for CPW components. *IEEE Trans. Microw. Theory Tech.* 45, 12 (1997), 2515–2523.
[32] Montri Wiboonrat. 2014. Data center infrastructure management WLAN networks for monitoring and controlling systems. In *ICOIN*. 226–231.
[33] Deliang Yi, Xin Zhou, Yonggang Wen, and Rui Tan. 2019. Toward efficient compute-intensive job allocation for green data centers: A deep reinforcement learning approach. In *ICDCS*. 634–644.