Feedback Control Scheduling in Energy-Efficient and Thermal-Aware Data Centers

Xiaogang Zhao, Tao Peng, Xiao Qin, *Senior Member, IEEE*, Qiping Hu, Ling Ding, and Zhijun Fang

***Abstract*—This paper presents a model-predictive control-based scheduling strategy called ThermoRing to reduce cooling costs in data centers. ThermoRing makes use of an online feedback control mechanism to improve thermal management of energy-efficient clusters in a data center. ThermoRing aims at keeping the maximum inlet temperatures of the nodes under a redline temperature limit with little stability errors. Importantly, the ThermoRing approach is capable of dealing with emergency conditions (e.g., node fan shutdown and unexpected rising task arrival rates) by dynamically balancing load among the nodes. ThermoRing incorporates a heat distribution matrix to model the thermal characteristics of a data center housing cluster. ThermoRing is conducive to thermal management in data centers with high-scheduling performance and stability. Using a real-world online bookstore trace, we conduct extensive experiments to compare the performance of ThermoRing with three existing solutions (i.e., C-Oracle, Ad-hoc, and MinHR). The experimental results show that ThermoRing improved the system throughput by more than 10% under regular load conditions and by 40% in emergency cases. ThermoRing also significantly improves the energy efficiency of MinHR, which is a thermal-aware scheduler.**

***Index Terms*—Emergency, energy efficient, feedback, heat recirculation, multiple-input multiple-output (MIMO), model prediction control (MPC), redline temperature.**

Manuscript received July 23, 2014; accepted February 28, 2015. Date of publication June 1, 2015; date of current version December 14, 2015. This work was supported in part by the Basic Research Program of China Surveying and Mapping Bureau under Grant 216153651, in part by the Natural Science Foundation of Hubei Province under Grant 201FFB04505, and in part by the National Natural Science Foundation of China under Grant 61272272. The work of X. Qin was supported by the U.S. National Science Foundation under Grant CCF-0845257 (CAREER), Grant CNS-0917137 (Computer Systems Research), and Grant CCF-0742187 (Computing Processes and Artifacts). This paper was recommended by Associate Editor N. N. Xiong.

X. Zhao and Q. Hu are with the International School of Software, Wuhan University, Wuhan 430079, China (e-mail: zxgang302@whu.edu.cn; huqp@whu.edu.cn).

T. Peng is with the School of Mathematics and Computer, Wuhan Textile University, Wuhan 430073, China (e-mail: pt@wtu.edu.cn).

X. Qin is with the Department of Computer Science and Software Engineering, Shelby Center for Engineering Technology, Samuel Ginn College of Engineering, Auburn University, Auburn, AL 36849-5347 USA (e-mail: xqin@auburn.edu).

L. Ding is with the Department of Computer Science, Hubei University of Science and Technology, Xianning 437100, China (e-mail: dn0715dn@163.com).

Z. Fang is with the College of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai 201620, China (e-mail: zjfang@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2015.2434797

I. Introduction

COOLING costs for data centers are significantly affected by the supply temperatures of computer room air conditioners (CRACs). Traditional off-line thermal-aware scheduling strategies minimize inlet temperatures under an assumption that power consumption is given in *a priori*. Emergencies like the failures of cooling fans or air conditioners can unexpectedly drive inlet temperatures up. Such a dramatic increase in inlet temperatures has not yet been addressed in conventional methods. In this paper, we solve the temperature fluctuation problem in various circumstances. We also show how our temperature feedback controller—ThermoRing—guarantees temperature stability without considerable performance degradation.

A. Motivations

Our proposed ThermoRing is motivated by the following three reasons.

1) To reduce ever growing cooling costs for data centers through thermal-ware task scheduling strategies.
2) To constraint real-time temperature fluctuation caused by uncertain inner disturbances (e.g., power consumption model variation) and outer disturbances (e.g., cooling fan failures).
3) To achieve good thermal management while taking into account heat recirculation among multinodes.

Cooling energy consumption accounts for a significant portion of the total energy consumption in a data center [1]–[3]. A key challenge toward optimizing the operation of a data center is to improve overall energy efficiency by minimizing the data center's cooling cost. Cooling energy is mainly consumed by computer room air conditioners or CRACs, which keep inlet temperatures of computing nodes in a data center below a threshold (i.e., redline temperature) to improve hardware reliability [4]. Raising the supply temperature of CRACs could reduce cooling costs. Tang *et al.* [5] developed a genetic algorithm that offers approximate optimal solutions to minimize inlet temperatures. Li *et al.* [6] proposed a method to predict inlet temperatures and to save cooling costs by managing workload. These algorithms are inadequate for online job scheduling due to either high-computational complexity or the lack of a comprehensive scheduler.

It is appealing to develop online thermal-aware job schedulers that minimize heat recirculation by making thermal-friendly scheduling decisions. Mukherjee *et al.* [7] proposed a spatial-temperature thermal-aware job-scheduling

2168-2216 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

method, which extends the Tang algorithm to minimize inlet temperatures through online job scheduling. By estimating task arrival rates, an online dynamic server provision and workload distribution mechanism designed by Abbasi *et al.* [8] aims to minimize inlet temperatures. Their mechanisms do not take into account real-time temperature changes caused by workload variations, nor do their schemes have the ability to resist emergency disturbances.

Feedback controllers have been integrated with online job schedulers in distributed systems to achieve good transient and stable performance [9]–[11]. Lu *et al.* [12] proposed a feedback control real-time scheduler to satisfy system performance (e.g., missing rate) in unpredictable workload environments. Raghavendra *et al.* [13] applied a feedback controller to save energy through coordinating the power management of data centers at the virtual machine, node, and cluster levels. Feedback controllers are able to efficiently handle thermal emergencies, because they can quickly capture changes in outputs while manipulating inputs to stabilize systems temperatures. With a proportional-integral-derivative (PID) feedback controller in place, C-Oracle [14] reduced task arrival rates to avert components (e.g., CPUs) temperatures from exceeding a redline temperature during the course of an emergency due to failures of cooling fans. The downside of C-Oracle is that it neglects the importance of heat recirculation among multiple nodes and inner disturbances.

Recently, a modern feedback control algorithm—model prediction control (MPC)—has been applied to online job schedules to optimize system power consumption under complicated system constraints. Wang and Chen [15] and Wang and Wang [16] made use of MPC to capping power of clusters by dynamically changing CPU frequencies according to workload conditions. MPC was also employed to control multicore processor power consumption under temperature constraints [17].

This paper presents ThermoRing, a thermal-aware scheduler that incorporates both MPC and heat recirculation to achieve high-energy efficiency in data centers. Inlet temperatures are sampled during each sampling period and task allocations are adjusted based on tracked reference temperature under changing workload. ThermoRing aims to cope with thermal emergencies by controlling task allocations to computing nodes. Compared with the existing schedulers, ThermoRing significantly reduces the cooling costs of clusters. The main contributions of this paper are summarized as follows.

1) We designed an online scheduler called ThermoRing that incorporates a feedback controller to improve energy efficiency of data centers. Jobs are dynamically distributed to nodes according to heat recirculation among these nodes.
2) With a given inlet temperature, ThermoRing avoids job scheduling errors that cause inlet temperatures exceed a reference temperature. The temperature tracking stability is guaranteed by ThermoRing even when workload conditions are dynamically changing.
3) When emergencies (e.g., fan failures, etc.) occur, ThermoRing can immediately adjust input task admission rates of related nodes to resist these disturbances while keeping the peak inlet temperature under the reference temperature.
4) We use real-world bookstore traces to evaluate the stability, performance, and energy efficiency of ThermoRing. The experimental results indicate that ThermoRing not only manages temperatures during emergencies better than dynamic scheduling approaches like Ad-hoc and C-Oracle-Ex, but it also improves cooling efficiency of the static scheduling methods like Coolest-inlet and MinHR.

The remainder of this paper is organized as follows. Section II describes the characteristics of data centers. ThermoRing, a temperature-aware job scheduler, is proposed in Section III. Performance evaluation is presented in Section IV. Section V outlines previous studies related to energy-efficient computing, cooling cost saving, and emergency management. We conclude this paper in Section VI.

II. System Modeling of Data Centers

In this section, we describe four models used to characterize CPU utilization (see Section II-A), power consumption (see Section II-B), heat recirculation (see Section II-C), and cooling cost of data centers (see Section II-D). These models provide a foundation for building our online thermal-aware job scheduler based on a feedback control mechanism.

A. CPU Utilization Model

The CPU utilization of nodes is linearly proportional to task arrival rates [8], [18], [19]. We express the CPU utilization as follows:

$$u(kT) = c\lambda(kT) \tag{1}$$

where λ is the task arrival rate, T is the time during which tasks are processed. And u is the CPU utilization. kT means the kth sampling period. A task can cause the CPU utilization increases by c.

B. Power Consumption Model

The power consumption of the ith node in a data center can be derived from the node's CPU utilization. Thus, the power consumption of node i is expressed as [20], [21]

$$P_i = b + mu_i \tag{2}$$

where b is the power consumption of the node when it is sitting idle, m is the power consumption difference between a fully utilized node and an idle node. The power consumption of node i in time period kT can be written as below

$$P_i(kT) = b + mu_i(kT). \tag{3}$$

The power consumption of node i can be derived from (1) and (3)

$$P_i(kT) = P_i((k-1)T) + mc\Delta\lambda_i(kT). \tag{4}$$

$P_i((k-1)T)$ is the power consumption of node i during time interval $(k-1)T$, $\Delta\lambda_i(kT)$ is the change in the task arrival rate of node i in period kT compared to period $(k-1)T$.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

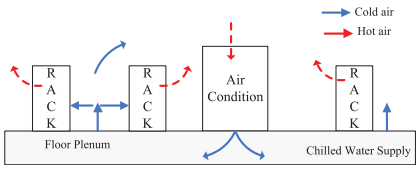


Fig. 1. Heat recirculation in data center.

C. Heat Recirculation Model

Heat recirculation plays a key contribution to increase the costs of cooling data centers. It is very important to characterize the heat recirculation and its impact on inlet temperatures of computing nodes in a data center. A handful of approaches have been designed to model heat recirculation in data centers [5], [6], [14], [22]. Those models are well investigated and well validated by real-world thermal data measured with sensors. Compared with the computational fluid dynamics (CFD) models, these models can more quickly and accurately predict inlet temperatures. Heat recirculation plays a key contribution to increase the costs of cooling data centers. It is very important to characterize the heat recirculation and its impact on inlet temperatures of computing nodes in a data center. A handful of approaches have been designed to model heat recirculation in data centers [5], [6], [14], [22]. Those models are well investigated and well validated by real-world thermal data measured with sensors. Compared with the CFD models, these models can more quickly and accurately predict inlet temperatures.

In a typical data center, hot air is extracted from the ceiling and cold air is supplied from the raised floor [6]. Fig. 1 shows that cold air is drawn by the chassis. The temperature T^i_{in} of air drawn by the ith chassis is called inlet temperature of chassis i. The inlet temperature and power consumed by the node contribute to the outlet heat Q^i_{out} of node i. Outlet heat Q^i_{out} at time interval kT can be obtained from the following expression:

$$Q^i_{\text{out}}(kT) = Q^i_{\text{in}}(kT) + P_i(kT). \tag{5}$$

In the above equation, the inlet heat Q^i_{in} of node i at time interval kT can be written as

$$Q^i_{\text{in}}(kT) = \rho c_p f_i T^i_{\text{in}}(kT) \tag{6}$$

where ρ, c_p, f_i are thermo-physical values. Specifically, ρ is density of air (a typical value of ρ is 1.19 kg/m^3), c_p is the heat of air (a typical value of c_p is 1005 J Kg^{-1} K^{-1}), and f_i is air flow rate in node i (a typical value of f_i is 520 CMF $=$ 0.2454 m^3/s) [5]. Here, we use a cross-interference coefficient matrix $\mathbf{A}_{n\times n} = \{\alpha_{ji}\}$ to characterize the heat recirculation, where α_{ji} denotes the fraction of the outlet heat of node j contributes to the inlet heat of node i. Therefore, the inlet heat of node i at time kT could be calculated as

$$Q^i_{\text{in}}(kT) = \sum_{j=1}^{n} \left(\alpha_{ji}\left(Q^j_{\text{in}}(kT) + P_j(kT)\right)\right) + Q^i_{\text{sup}}(kT). \tag{7}$$

By applying the thermodynamic constants to (7), we can model the inlet temperatures of computing nodes. Let K_i equal to constant $\rho f_i c_p$, then, we have $\mathbf{K}_{n\times n} = \text{diag}\{K_1, K_2, \ldots, K_n\}$. The inlet-temperature vector $\mathbf{t}_{\text{in}}$ can be expressed as [5]

$$\mathbf{t}_{\text{in}} = \mathbf{t}_{\text{sup}} + \mathbf{Dp}, \mathbf{D} = \left[\left(\mathbf{K} - \mathbf{A}^T\mathbf{K}\right)^{-1} - \mathbf{K}^{-1}\right] \tag{8}$$

where $\mathbf{D}$ is the heat distribution matrix that converts power supplied by $\mathbf{p}$ into the temperature domain. We can obtain from (8) the inlet temperature of node i in time period kT

$$\mathbf{t}_{\text{in}}(kT) = \mathbf{t}_{\text{sup}}(kT) + \mathbf{Dp}(kT). \tag{9}$$

As inlet temperatures have to be less than or equal to a redline temperature, it follows that

$$t^i_{\text{in}}(kT) \le t_{\text{red}}, \ \ \forall i = 1, 2, \ldots, n. \tag{10}$$

D. Cooling Cost Model

Each component of a computing node (e.g., CPU, disk, and memory) must be kept under a certain temperature. Being active upon a redline temperature too long can inevitably shorten the computing node's lifetime. A cooling system is deployed to ensure that inlet temperatures are below the redline. The supply temperature T_{sup} of the cooling system affects the energy efficiency of data centers. The cooling system's efficiency is quantified in terms of coefficient of performance (COP) [20]

$$\text{COP}(T_{\text{sup}}) = 0.0068T^2_{\text{sup}} + 0.0008T_{\text{sup}} + 0.458. \tag{11}$$

The COP value increases when supply temperature goes up; increasing supply temperatures T_{sup} results in high-cooling efficiency. The below equation shows how to derive the cooling cost from the COP value

$$P_{AC} = \frac{P_C}{\text{COP}\left(T_{\text{sup}}\right)} \tag{12}$$

where P_C is the total power consumed by all the computing nodes in a data center.

Air temperature $T_{\text{sup}}(kT)$ supplied by a cooling system can be adjusted to a higher supply temperature $T'_{\text{sup}}(kT)$ to reduce the cooling cost. The following expression shows how to select an appropriate supply temperature to achieve cooling cost savings:

$$T'_{\text{sup}}(kT) + \max(\mathbf{Dp}(kT)) = t_{\text{red}}, T'_{\text{sup}}(kT) = T_{\text{sup}}(kT) + \Delta \tag{13}$$

where Δ is the margin between peak inlet temperature and t_{red}. And reducing $\max(\mathbf{Dp}(kT))$ helps in increasing $T'_{\text{sup}}(kT)$.

III. Design of ThermoRing

We start this section by formulating the thermal-aware scheduling problem. Then, we outline the architecture of ThermoRing—a feedback control-based thermal-aware scheduler. Next, we propose the MPC controller used in ThermoRing. We also study the stability of ThermoRing. Here, some symbols and annotations used in this paper are described in Table I.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SYMBOL AND ANNOTATION

Symbol	Annotation
λ	task arrival rate
t_{sup}	supply temperature of CRAC
t_{red}	redline temperature: usually 25°
t_{ref}	reference temperature: $t_{sup} <= t_{ref} <= t_{red}$
$\mathbf{t}_{in}$	the vector of inlet temperature $\{t_{in}^i\}$
$n1$	temperature disturbance caused by uncertain reasons
c	cpu rise caused by one request
u	cpu utilization
m	power difference between fully utilized node and idle node
p	the vector of power consumption$\{P_i\}$
D	Heat distribution matrix

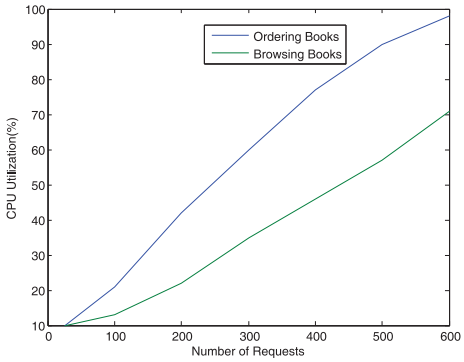


Fig. 2. Impact of system load on CPU utilization [23].

A. Problem Statement

Upon the arrival of a task at an energy-efficient data center, the task is usually scheduled to the node that has the least heat recirculation factor (HRF) (see [7], [8]). A challenging issue is the outer disturbances which make the inlet temperature of this node exceed a redline temperature t_{red} due to a combination of reasons (e.g., the failures of cooling fans or air conditioners). Outer disturbances are handled by reducing the number of tasks assigned to the nodes with high temperatures through the PID controller [14]. Unfortunately, the PID controller does not reduce inlet temperatures by taking into account the heat recirculation among computing nodes.

If the maximum inlet temperature is given *a priori*, an appropriate supply temperature could be configured in advance to optimize the cooling cost. Nevertheless, another grand challenge to be overcome is the inner disturbances caused by the uncertainty of the power consumption model [see (4)], which could degrade the effectiveness of cooling cost reduction. This is because determining the appropriate number of active computing nodes to satisfy task needs largely depends on an accurate power profiling [7], [8]. Real-world traces show that the power consumption model of tasks is time-variant according to request types (see [23, Fig. 2]). The maximum inlet temperature would always oscillate around a reference inlet temperature [see (9)], which makes it nontrivial to manage the supply temperature. In fact, the precise power consumption model could not be got at the design time, so many factors such as user subjective decisions on task power estimations would also make the maximum inlet temperature fluctuate.

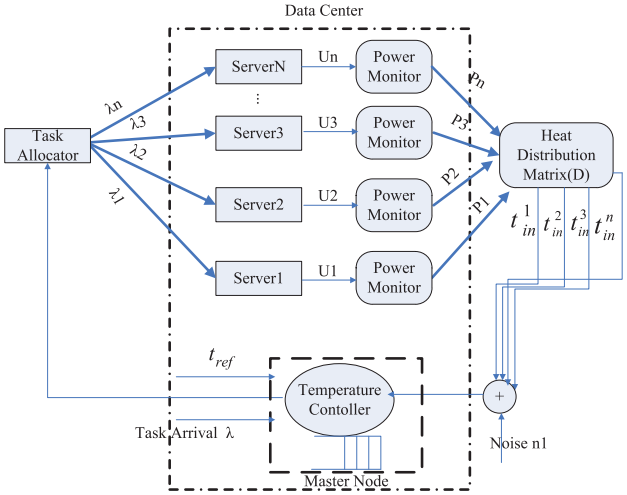


Fig. 3. Feedback control loop of ThermoRing.

Thermal-aware task scheduling for data centers is a grand challenge that raises several key problems.

1) *Challenge 1:* Given the max inlet temperature that equals to a redline temperature, can our scheduler get better stability compared with *ad hoc* methods while considering heat interference among the nodes (see Section IV-C). How to guarantee stability even when task power model [see (4)] is dynamically changing under various workloads (see Section IV-F).
2) *Challenge 2:* When emergencies (e.g., fan failures) dramatically drive node temperatures up, how to immediately manage task arrival rates to keep temperatures below the redline temperature and minimize performance overhead such as throughput (see Sections IV-D and IV-E).
3) *Challenge 3:* Can our temperature-aware scheduler offer energy savings compared with other scheduling strategies (see Section IV-G).

B. ThermoRing Overview

In this section, we describe the architecture of ThermoRing, which features a multiple-input multiple-output feedback control loop that dynamically adjusts task arrival rates of nodes to enforce the reference temperature t_{ref}.

Let us consider a cluster that comprises of n computing nodes. λ is the task arrival rate to the cluster in a data center. There is a head node managing newly arrived tasks in a queue; $\lambda_1, \lambda_2, \ldots,$ and λ_n are task dispatch rates to the n nodes managed by the head node. The overall task arrival rate is the summation of the dispatch rates of all the nodes; thus, we have $\lambda = \lambda_1 + \lambda_2 \ldots + \lambda_n$. System noise $n1$ is triggered by unpredictable events like cooling fan failures or air conditioner breakdown, etc., which cause inlet temperature increases in the data center. The feedback control loop (see Fig. 3) in ThermoRing is periodically invoked according to the following policy.

1) After tasks assignment, CPU utilization vector $\mathbf{u}(u_1, u_2, \ldots, u_n)$ of the nodes are updated.
2) Power consumption vector $\mathbf{p}(P_1, P_2, \ldots, P_n)$ is calculated using the power model (see Section II-B).
3) Inlet temperature vector $\mathbf{t}_{\text{in}}(t_{\text{in}}^1, t_{\text{in}}^2, \ldots, t_{\text{in}}^n)$ is derived from heat distributed matrix (**D**). The output

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

temperatures coupled with system noise $n1$ are provided to the temperature controller. The system noise is zero in nonemergency cases.

4) The temperature controller determines dispatch rates (i.e., manipulated variables) based on the difference between ideal trajectory (t_{ref}) and current temperatures (i.e., controlled variables).

From (9), we can deduce that inlet temperature of data center in period kT which assumes that $\mathbf{t}_{\text{sup}}$ does not change in a relatively long period $t(0 < iT < t, i = 1, 2, \ldots, k, \ldots, n)$

$$\mathbf{t}_{\text{in}}(kT) = \mathbf{t}_{\text{in}}((k-1)T) + \mathbf{D}\Delta\mathbf{p}(kT) \tag{14}$$

where $\Delta\mathbf{p}(kT)$ is power consumption difference of data center between time interval $(k-1)T$ and kT. The inlet temperature vector $\mathbf{t}_{\text{in}}$ of the data center during period kT is derived from (4) and (14). Thus, we have

$$\mathbf{t}_{\text{in}}(kT) = \mathbf{t}_{\text{in}}((k-1)T) + mc\mathbf{D}\Delta\lambda(kT). \tag{15}$$

Hardware emergencies like cooling fan failures on certain node are referred as outer disturbances, which can slow down the air flow speed in servers and thereby raise CPU and hard disk temperatures [24]. Unexpected heat accumulation due to cooling fan failures boosts the inlet temperature of this node [25] as well as correlated nodes [see (6), (8)]. Due to node localities, computing nodes have a wide range of heat recirculation coefficients, which have significant impacts on inlet temperatures. Ideally, temperature fluctuations should be contained through temperature feedback control. These unstable inlet temperature fluctuations are incorporated into system noise $n1$ (see Fig. 3).

Inner disturbances such as system model variation would also make inlet temperature fluctuate. For example, system parameter c in (1) can be configured in a way to represent the access patterns of submitted tasks. In a dynamic computing environment, it is very challenging to accurately predict the CPU utilization for unknown access patterns like that in Fig. 2. The uncertainty of the CPU utilization model would make the inlet temperature oscillate according to (15). Based on these two kinds of disturbances, the actual inlet temperature vector $\mathbf{t}_{\text{in}}$ during period kT can be estimated as

$$\mathbf{t}_{\text{in}}(kT) = \mathbf{t}_{\text{in}}((k-1)T) + mgc\mathbf{D}\Delta\lambda(kT) + n1. \tag{16}$$

gc is the actual CPU utilization change according to one request arrival. g is the difference between estimated utilization and real utilization. The basic emergency handling principle is minimizing cooling cost and maximizing system utilization while not changing system structure. This means we should not reduce supply temperature or sleep certain nodes to reduce inlet temperatures in the normal method. So we can reduce task dispatching rates to nodes to reduce inlet temperatures which would make $\lambda > \lambda_1 + \lambda_2 + \cdots + \lambda_n$. So some tasks would be rejected and the appropriate controller should make these task rejection events happened as little as possible.

C. Temperature Controller

At the heart of ThermoRing is a temperature controller, which is designed around the model predictive control theory or MPC [26]. MPC is a model that predicts future evolution of a process to optimize control signals [27]. The controller solves an optimal control problem over a finite future horizon of N steps called the prediction horizon. An input trajectory includes control inputs in M control periods, $\Delta\lambda(k)$, $\Delta\lambda(k+1|k)$, $\ldots$, $\Delta\lambda(k+M-1|k)$, where M is the control horizon. The notation $\Delta\lambda(k+i|k)$ means that the value of variable $\Delta\lambda$ at time $(k+i)T$ depends on the value at time kT, where T is a control period.

At the end of each control period, the controller computes control input $\Delta\lambda(k)$ aiming to minimize the following cost function:

$$V(k) = \sum_{i=1}^{P} \left\| \mathbf{t}_{\text{in}}(k+i|i) - \mathbf{t}_{\text{ref}}(k+i|k) \right\|^2_{\mathbf{Q(i)}} + \sum_{i=0}^{M-1} \left\| \Delta\lambda(k+i|k) + \lambda(k+i|k) - \lambda_{\max} \right\|^2_{\mathbf{R(i)}} \tag{17}$$

where P is the prediction horizon, M is the control horizon, $Q(i)$ is the tracking error weight, and $R(i)$ is the control penalty weight vector; control horizon M is smaller than prediction horizon P (i.e., $M < P$).

The first term in the cost function [see (17)] quantifies the tracking error, which is the difference between node temperature vector $\mathbf{t}_{\text{in}}(k+i|k)$ and reference temperature $\mathbf{t}_{\text{ref}}$. By minimizing the tracking error, the closed-loop system converges to temperature set point $\mathbf{t}_{\text{ref}}$, provided that the cluster is stable. Here, we explore the ability of MPC to minimize the peak inlet temperature while maximizing the throughput of every node. If the temperatures of all the nodes are approaching the reference temperature, the heat recirculation experienced by the cluster can be reduced and the peak inlet temperature can be minimized. In doing so, we are able to cut the cluster's cooling cost.

The second term in (17) is the control penalty. The feedback controller in ThermoRing optimizes system cooling efficiency by minimizing the difference between the highest task dispatch rate threshold (i.e., $\lambda_{\max}$) and the new dispatch rate (i.e., $\lambda(k+i+1|k) = \Delta\lambda(k+i|k) + \lambda(k+i|k)$) along the control horizon. The control weight vector, **R(i)**, is commonly configurable and can be tuned to represent preferences among nodes. In this paper, all the nodes share the same control weight. The second item is focused on utilization; ThermoRing aims at maximizing the system utilization. If the task dispatch rates are all approaching to $\lambda_{\max}$, the system throughput is high with optimized cooling cost.

The feedback controller in ThermoRing is subject to two constraints. First, the task dispatch rate of each node should be within a reasonable range, which is a configurable parameter determined by the performance of each node. Second, inlet temperatures should not be higher than the specified reference

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: ThermoRing: The Thermal-Aware Scheduler Based on a Feedback Control Mechanism

```
Procedure INITIALIZATION( )
    Set the t_ref for every node according to requirement;
    Set the initial task rate to every node;
end Procedure
Event INVOCATION_IN_SAMPLING_PERIOD( )
pinlet = max(t_in^1, t_in^2 ... t_in^n);
if pinlet < t_ref then
    foreach node i in data center do
        MPC controller increase tasks dispatching rate Δλ_i to node
        i which satisfy (Δλ_i + λ_i < λ_max);
        Dispatch tasks to node i;
    end
else
    MPC controller adjust the task distribute rates to all related
    nodes to lower the pinlet according to (17) while
    complying with (18);
    Dispatch tasks to all related nodes;
end
end Event
```

temperature. These two constraints can be formally expressed as follows:

$$\lambda_{\min} \le \Delta\lambda_i(k) + \lambda_i(k) \le \lambda_{\max}(1 \le i \le N)$$
$$t_{\text{in}}^i \le t_{\text{ref}}. \tag{18}$$

Equation (18) prevents under-utilization problems by making temperatures of all the nodes close to the reference temperature under high-CPU utilization. The under-utilization problem inevitably causes poor application performance (e.g., low-quality videos); whereas the CPU overloaded problem leads to degraded system performance (e.g., long response time). To make good tradeoff, ThermoRing controls CPU utilization in the way that task dispatch rates vary anywhere between $\lambda_{\min}$ and $\lambda_{\max}$. If a node's task dispatch rate is dropped to $\lambda_{\min}$, the node becomes a good candidate being placed into the low-power mode to further reduce its outlet temperature. Combining (17) and (18), we can obtain a temperature optimization strategy in ThermoRing, which is orthogonal to power management mechanisms. ThermoRing can be seamlessly integrated with existing power management schemes to handle emergency cases.

D. ThermoRing Scheduling Algorithm

The nodes of a cluster share the same redline temperature. Algorithm 1 illustrates the pseudocode of ThermoRing—the thermal-aware scheduler based on a feedback control mechanism. The INITIALIZATION procedure assign initial values to system parameters. In each sampling period kT, event INVOCATION_IN_SAMPLING_PERIOD is invoked and the MPC controller provides a solution—task dispatch rate $\Delta\lambda_i(k) + \lambda_i(k)$ to computing nodes in order to minimize $V(kT)$ [see (17)] under given constraints [see (18)].

In regular cases, where inlet temperatures are lower than the redline temperature (i.e., pinlet $< t_{\text{ref}}$), the MPC controller increases each node task dispatch rate and allows inlet temperatures to go up close to t_{ref} while improving system utilization. In emergency cases, where inlet temperatures are higher than the redline temperature (i.e., pinlet $\ge t_{\text{ref}}$), the MPC controller attempts to reduce the corresponding task dispatch rate.

E. Overhead Analysis

We implement the ThermoRing system using the lsqlin solver in MATLAB, because the mpccon function is inadequate for modeling input and output constraints. The computational complexity of lsqlin is polynomial with respect to the number of nodes and the control/prediction horizons. The simulations were carried out on a personal computer workstation, which has a Core 2 Duo CPU 3.00 GHz and 4 GB RAM. Each invocation of the solver offered by MATLAB takes less than 6 ms to manage 500 servers in ThermoRing. If the solver were implemented in the C language, the solver's execution time could be further reduced. It is noteworthy that such a small overhead of ThermoRing corresponds to less than 1% of CPU utilization when the sampling period is set to 1 s.

Today large-scale data centers are housing more than 100 000 servers, which means the overhead of our ThermoRing would substantially increase to compute 10 000 control variables (i.e., one variable per ten servers). To address the scalability issue of ThermoRing, we take a distributed approach by partitioning servers into small groups (e.g., each group consists of 80–300 servers), each of which is controlled by ThermoRing. Such a distributed approach has been widely adopted in real-world data centers (see Microsoft's WAS [28] and Google's GFS [29]).

F. Stability Analysis

In this section, we analyze the stability of ThermoRing under dynamically changing workload conditions. A fundamental benefit of the feedback control-based scheduler lies in its theoretical confidence for system stability under dynamically changing workload. When a cooling fan fails or a computing component breaks down, the HRFs may change correspondingly, which also affect inlet temperatures. A computing cluster is stable if all inlet temperatures $\mathbf{t}_{\text{in}}(k)$ converge to the desired set point $\mathbf{t}_{\text{ref}}$ (i.e., $\lim_{k\to\infty}\mathbf{t}_{\text{in}}(k) = \mathbf{t}_{\text{ref}}$). The MPC controller in ThermoRing solves a finite horizon optimal tracking problem. The controlled input $\lambda(k)$ is computed from the current inlet temperature value $\mathbf{t}_{\text{in}}(k)$, the inlet temperature set point of the cluster $\mathbf{t}_{\text{ref}}$, and task dispatch rate $\lambda(k-1)$ determined in the previous time period.

In what follows, we outline the three-step process for analyzing the stability of a data center housing clusters under various workload conditions.

1) We compute control inputs $\lambda(k)$ that minimize the cooling cost function based on an initial system model [see (17)] with estimated parameter c.
2) We construct an improved system model with refined parameter $c' = gc$, where g represents unknown system gain caused by workload variation.
3) The closed-loop system model is executed by substituting the control inputs derived in the first step into the improved system model. We analyze the stability of the closed-loop system by computing all the poles of the closed loop system.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

If ThermoRing minimizes the tracking error, the maximum inlet temperature of all the nodes in a cluster will converge to set point $\mathbf{t}_{\text{ref}}$. The allowed variation range of g can be established by computing the values of g that cause the poles to move across the unit circle.

Three types of requests accessing an online bookstore system include browsing, shopping, and ordering [23]. The access patterns of these three request types are different. For example, we have $c_{\text{browsing}} = c_{\text{shopping}} = 1.5c_{\text{ordering}}$. The three-step analysis process ensure that when g is in a range between 0 and 3.58 (i.e., $0 < g < 3.58$), the ThermoRing system is stable. Any emergency caused by c variation can be completely handled by ThermoRing (see Section IV-F).

IV. Performance Evaluation

A. Experimental Environment

We run real-world workload in servers to collect traces used to drive our event-driven simulator, where the thermal-aware feedback control scheduler and the power monitor are implemented. The simulator resembles an online bookstore system that contains many Web servers and databases [23]. The task dispatch rate of each node is controlled by the ThermoRing scheduler. We make use of the power monitor to measure power consumption as well as the temperature vector, which are input parameters of the MPC controller. The controller computes the control input, $\Delta\lambda$, and return the dispatch rate back to ThermoRing.

We make use of OpenFoam, a CFD simulation software to conduct thermal simulations to obtain the heat distribution matrix (**D**) incorporated into ThermoRing. Fig. 4 depicts the physical layout of the data center we simulated by OpenFoam. The 50 nodes data center has two rows of industry standard 42U racks with physical dimensions $7.6 \times 7.6 \times 7.6$ m^3. The cold air is supplied by one CRAC with the flow rate 8.5 m^3/s. The CRAC is in the ceiling and 2.79 m away from the x-axis. There are ten racks, each of which has five 7U chassis. The first row is 2.5 m away from x-axis, whereas the second row is 4.2 m. In our extensive experiments, we set the number of computing nodes to 4, 20, 30, and 50, respectively. We also keep the corresponding physical layout and cooling configuration unchanged. All the heat recirculation matrices (D_4–D_{50}) are obtained by these simulations. In an initial experiment, we simulated a four-node cluster in one rack to investigate the impact of ThermoRing on the behavior of the nodes. The power consumption parameters of computing nodes resemble those of Dell PowerEdge 1855 Servers. A similar experimental setup can be found in [5]. Each node in the cluster contains ten blade servers. The power consumption of an idle node is 820 W; the power consumption difference between a fully used node and an idle node is 500 W. The order of the four nodes from top to floor is [4, 2, 3, 1]. In this paper, we set the reference temperature t_{ref} for the four nodes to 16.6°. When the size of the cluster goes up to 50 nodes, the maximum inlet temperature would be 30° and the t_{ref} could be set to t_{red} that equals to 25° for a typical large data center. The sampling period T used in our experiment is 1 s.

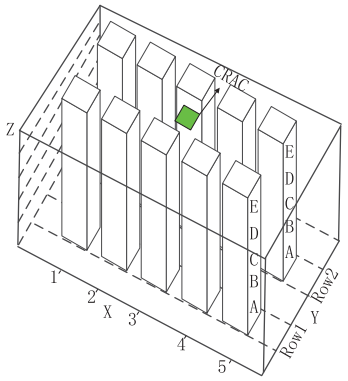


Fig. 4. Physical layout of 50 nodes data center. There is one CRAC which is in the ceiling and positioned between the two rows of racks.

B. Baselines

We compare the ThermoRing scheduler against two online-baseline schedulers, namely, OPEN and C-Oracle-Ex. OPEN is an open-loop algorithm that keeps task dispatch rate as a constant. OPEN assigns task priorities based on differences between current temperature and the reference temperature. C-Oracle [14] features a feedback control loop that controls temperature by dynamically adjusting task dispatch rates. It is worth noting that C-Oracle is a single-node temperature-control algorithm. C-Oracle-Ex, an extended version of C-Oracle, is designed to control temperatures of multiple nodes in a distributed system. We show the strength of MPC-based ThermoRing by comparing it with C-Oracle-Ex powered by a simple linear controller.

C. Control Accuracy

In this group of experiments, we analyze the stability error of ThermoRing compared with other dynamic scheduling schemes under the same temperature reference. The findings show that ThermoRing improves the stability performance of the traditional scheduling methods in nonemergency scheduling cases (Challenge 1 in Section III-A).

ThermoRing and C-Oracle-Ex are two dynamic schedulers that incorporate heat recirculation among computing nodes to reduce cooling cost. In addition to C-Oracle-Ex, we compare TheroRing with an Ad-hoc control-based thermal-aware scheduler, which was proposed in [15]. Ad-hoc increases task dispatch rates to most heat-inefficient nodes when peak temperature pinlet is lower than reference temperature t_{ref}; Ad-hoc decreases task rates to most heat-efficient node when peak temperature pinlet is higher than reference temperature t_{ref}. When pinlet is lower than t_{ref}, increasing the task dispatch rates of heat-inefficient nodes help to improve the utilization of all the nodes. If pinlet is higher than t_{ref}, increasing task dispatch rates of heat-efficient nodes makes temperatures drop efficiently.

We deploy the Ad-hoc control-based scheduler on the four-node cluster. In this experiment, temperature t_{ref} is set to 16.6° coupled with the corresponding maximal task dispatch rate $\lambda_{\max}$. Fig. 5 shows the peak temperature pinlet of the four nodes in the 35 sampling periods. Fig. 5(a) illustrates that managed by the Ad-hoc scheduler, the maximum inlet temperature of all the nodes reach 16.3° with the initial utilization of 50%.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

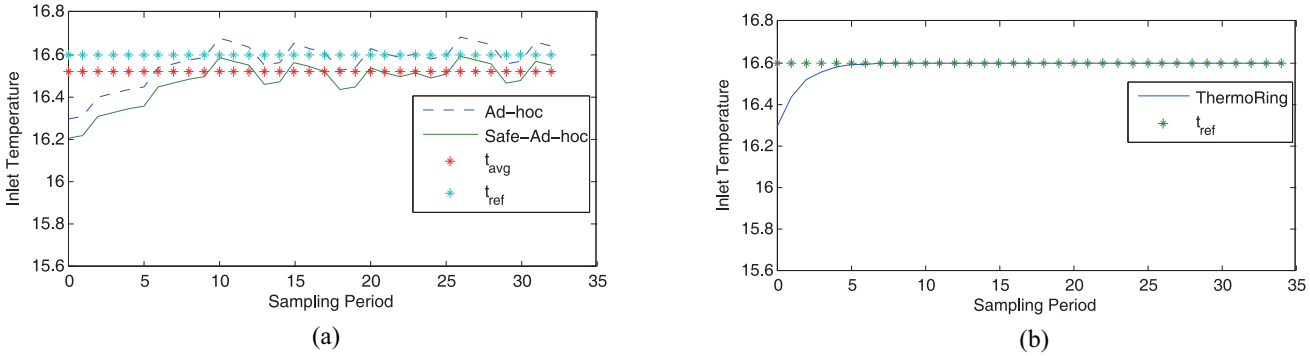


Fig. 5. Stable state error comparison of peak inlet temperature of four nodes data center using ThermoRing and Ad-hoc strategies. (a) Ad-hoc. (b) ThermoRing.

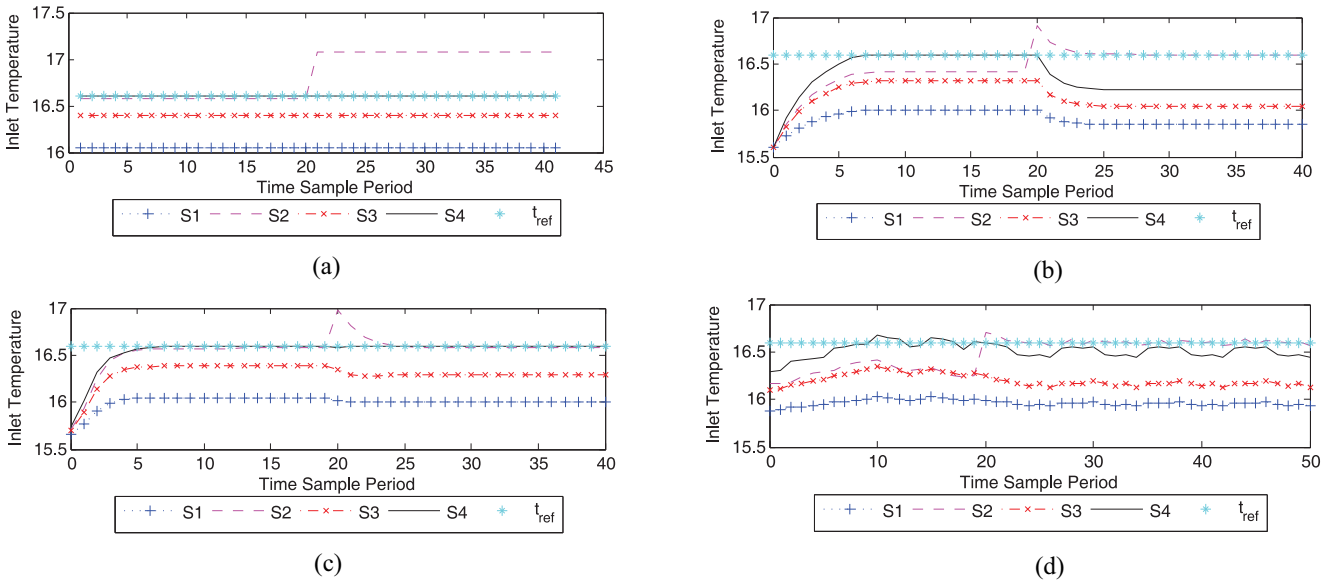


Fig. 6. Stability of the four schedulers (a) OPEN, (b) C-Oracle-Ex, (c) ThermoRing, and (d) Ad-hoc, during the fan failure of node 2. S1–S4 means the corresponding four nodes (Server1–Server4) in the four nodes cluster.

We set parameter k to 10%. After ten sampling periods, the maximum temperature approximates to 16.6°; Ad-hoc only increases the task rate of one node during one sampling period. In the following 25 sampling periods, the temperature oscillates up and down around 16.6°; the task dispatch rate of each node oscillates in a round-robin fashion. This performance trend can be attributed to passive k dispatching rate up-down attempt between different nodes for reference temperature tracking. Safe Ad-hoc could maintain peak temperature under t_{ref} but sacrifices throughput.

Fig. 5(b) shows the performance results of ThermoRing, where prediction horizon P is set to 4 and control horizon M is set to 2. The maximum inlet temperature pinlet reaches the reference temperature quickly during the first five sampling periods. After that, temperature pinlet is very close to the reference temperature. Unlike Ad-hoc, ThermoRing manages task dispatch rates of all the nodes by considering heat recirculation among the nodes. As a result, ThermoRing improves the temperature-prediction accuracy for all the nodes. Since Ad-hoc has no capability of forecasting temperature information, it simply passively adjusts task dispatch rates to control inlet temperatures.

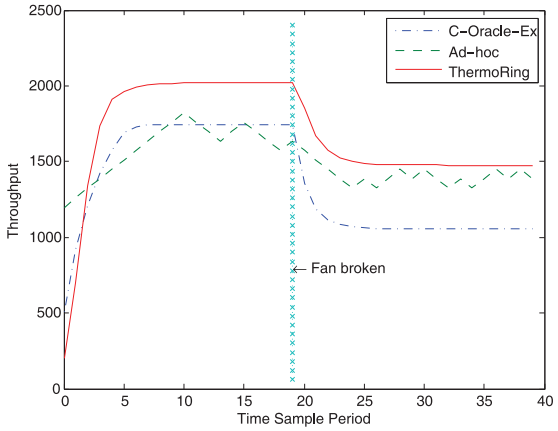


Fig. 7. Throughput of the three schedulers during the fan failure of node 2.

D. Emergency Handling in One Node

In this group of experiments, we evaluate the performance of ThermoRing, OPEN, C-Oracle-Ex, and ThermoRing under a thermal emergency (e.g., fan failures). We show that ThermoRing is able to resist outer disturbance without

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

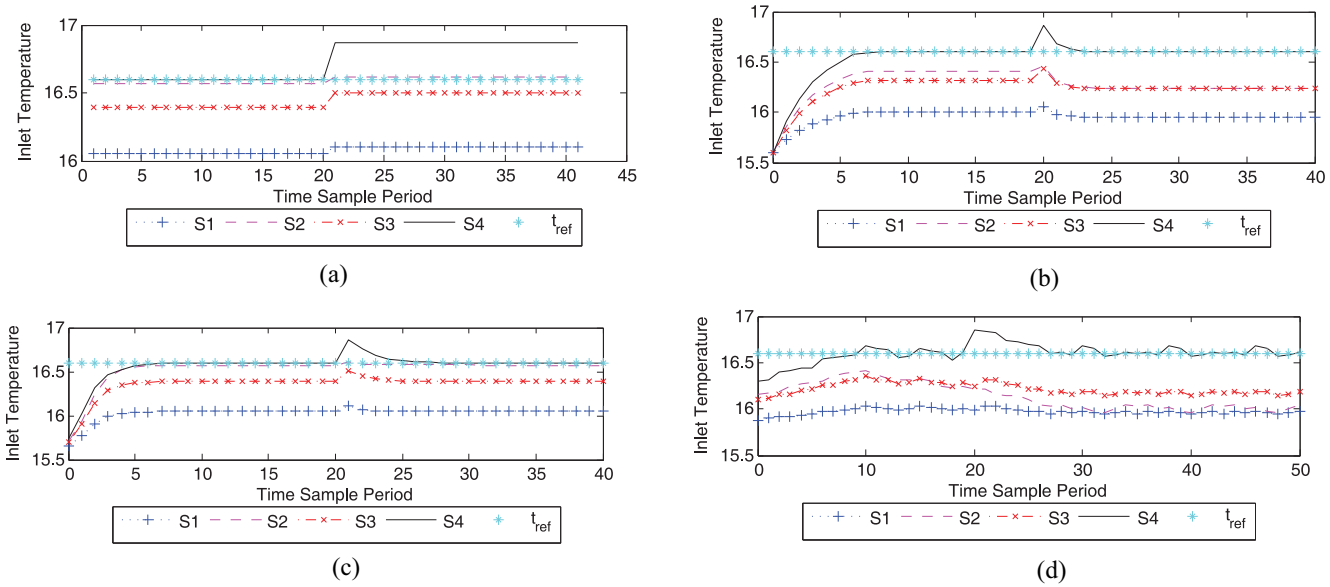


Fig. 8. Stability of the four schedulers (a) OPEN, (b) C-Oracle-Ex, (c) ThermoRing, and (d) Ad-hoc, when the task dispatch rate of node 3 increases to 30%.

significant performance degradation (see Challenge 2 in Section III-A). We set the reference temperature of the cluster to be 16.6°. Due to a fan failure, the node temperature goes up by 0.5°. We inject a fan failure after the 20th sampling period. The OPEN scheme well manages node temperatures by manipulating task dispatch rates, keeping the temperatures below the reference point t_{ref} when the cluster is stable [see Fig. 6(a)]. When the cooling fan of node 2 fails, the temperature of node 2 quickly rises above t_{ref}; OPEN is unable to take any action to decrease the temperature under this thermal emergency. C-Oracle-Ex is capable of handling the temperature increases incurred by the fan failure by the virtue of the feedback controller [see Fig. 6(b)]. Unfortunately, C-Oracle-EX aggressively decreases all the nodes' temperatures by forcing every node's task dispatch rate down to the minimum level. In contrast, ThermoRing judiciously adjusts the dispatch rates of all the nodes, thereby reducing node 2's temperature without imposing much negative performance impacts on the other three nodes. Fig. 6(c) shows that the temperatures of the other three nodes decrease during the four periods and then recover back to their original temperatures. ThermoRing takes advantage of incorporating heat recirculation in the process of emergency handling. Fig. 6(d) demonstrates that Ad-hoc keeps the inlet temperature of node 2 under t_{ref} while offering poor stability performance.

Fig. 7 shows that except for OPEN, the throughput of the cluster managed by the three schemes drops to reduce increased temperature. ThermoRing is conducive to maintaining high throughput during the period of emergency. For example, in the regular case, ThermoRing's throughput is 15% higher than those of C-Oracle-Ex and Ad-hoc. In the case of emergency, ThermoRing improves the throughput of Ad-hoc and C-Oracle-Ex by 10% and 50%, respectively. We observe that when a fan fails, C-Oracle-Ex reduces the dispatch rate of the node to the lowest level; similarly, Ad-hoc keeps the task rates as low as that of node 4, which is the most heat efficient node. Unlike Ad-hoc and C-Oracle-Ex, ThermoRing only changes the task rates of nodes 2 and 4 to a lower level while increasing the task rate of node 3 to achieve improved performance. ThermoRing makes scheduling decisions by keeping track of predicted temperatures $\mathbf{t}_{\text{in}}(k+i|i)$.

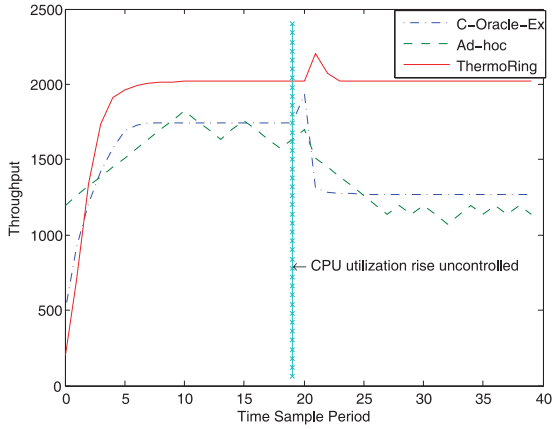


Fig. 9. Throughput of the three schedulers when the task dispatch rate of node 3 increases by 30%.

E. Emergency Handling in Multiple Nodes

Now, we are in position to study the emergency handling issues in multiple nodes, where temperatures of multiple nodes sharply increase due to some node's task dispatch rate change. The power consumption of a node is affected by its task rate, the increased power of the node with the high-CPU utilization makes adverse impacts on the neighboring nodes. Such impacts are modeled and expressed by (16). These experiments show that ThermoRing effectively solved the temperature fluctuation problem in data centers without causing noticeable adverse impacts (see Challenge 2 in Section III-A).

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

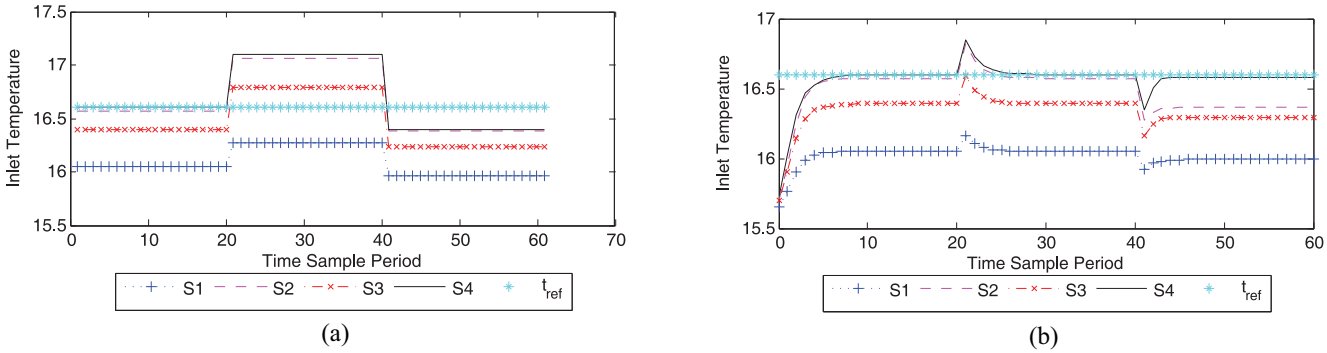


Fig. 10. Stability analysis under dynamically changing workload conditions. Workload is changed from browsing to ordering in an online bookstore system. (a) OPEN. (b) ThermoRing.

Suppose that the task rate of node 3 goes up to 30% suddenly after the 20th period due to an emergency or user access burst. Fig. 8(a)–(d) shows that the temperature of node 4 increases to the highest level even though node 4s task dispatch rate does not change, because the heat recirculation of node 3 is significant and node 4 is positioned on the top of the chassis. The dramatic temperature increase of node 3 makes temperatures of the other nodes go up. Hence, the task dispatch rates of the other nodes should be decreased accordingly to reduce their temperatures. OPEN is unable to handle this emergency case [see Fig. 8(a)]. The temperatures of all the nodes in ThermoRing [see Fig. 8(c)], C-Oracle-Ex [see Fig. 8(b)], and Ad-hoc [see Fig. 8(d)] are increasing in the 20th period due to the heat recirculation made by node 3. Nevertheless, these three schemes manage the emergency very well. ThermoRing and C-Oracle-Ex successfully reduce the increased temperatures four sampling periods after the emergency occurs; where Ad-hoc cools down the temperatures eight periods after the emergency is injected.

ThermoRing outperforms Ad-hoc and C-Oracle-Ex in terms of throughput in both regular and emergency cases (see Fig. 9). For example, the throughput of ThermoRing in the regular case is 15% higher than those of C-Oracle-Ex and Ad-hoc; ThermoRing improves the throughput of C-Oracle-Ex and Ad-hoc by more than 40% during the emergency period. We observe that all the task dispatch rates decrease when temperatures go up due to fan failures. In the cases of Ad-hoc and C-Oracle-Ex, the task dispatch rates decline to a low level to cool down the temperatures. Importantly, it only takes ThermoRing three sampling periods to get the task rates back to track reference while the task rate of node 3 continues decreasing. ThermoRing is conducive to managing node temperatures thanks to the consideration of heat distribution matrix, which makes it possible to maintain high throughput during any emergency period.

F. Dynamically Changing Workload

Now, we focus on the system stability of ThermoRing when the workload is dynamically changing. This experiment demonstrates that ThermoRing properly handled the inner disturbance caused by the dynamically changing workload (see Challenge 1 in Section III-A). To evaluate system stability, after the 20th sampling period we change the access workload from the ordering pattern to the browsing pattern. Fig. 10(b) confirms that ThermoRing performs very well under a dynamically changing workload. For example, the new c value becomes 1.5 times of the original c value; such a change could result in a significant change in the system temperature model expressed in (16). Fig. 10(b) shows that the peak inlet temperature suddenly exceeds t_{ref} during the workload change. With the feedback controller in place, ThermoRing keeps all the inlet temperatures under the t_{ref} by judiciously reducing the task rates. Here, Ad-hoc and C-Oracle-Ex are not mentioned for their low throughput but they could handle this situation for their feedback nature. On the other hand, the inlet temperatures could not be controlled by OPEN [see Fig. 10(a)]. At the 40th sampling period, the new c becomes 0.8 times of the original c value. Again, ThermoRing increases task dispatch rates of all the nodes to improve system performance without exceeding the reference temperature.

G. Static Energy Savings

In this section, we compare energy savings of ThermoRing temperature-aware job schedule strategy with other three static famous schedule strategies: 1) load balance; 2) MinHR; and 3) coolest inlet. Coolest inlet will always assign tasks to the node with the lowest inlet temperature. Load balance will distribute tasks evenly to all nodes. MinHR would assign tasks according to the ratio of each nodes' HRFs to the sum of all HRFs [5]. We change data center size from 20, 30, to 50 nodes. The supply temperature is 15°. These experimental results indicate that compared with the traditional scheduling strategies, ThermoRing is capable of greatly reducing cooling cost of data centers (see Challenge 4 in Section III-A).

In this set of experiments, we evaluate the impacts of the number of nodes on system performance. We set the number of nodes to 20, 30, and 50, while keeping the utilization at the level of 70%. Fig. 11 reveals that when the number of nodes increases, the cooling cost goes up dramatically. More importantly, the cooling cost discrepancy between ThermoRing and the other three schemes (i.e., MinHR, load balance, and coolest inlet) is widened for more nodes' spatial-thermal difference when we scale up the cluster from 20 to 50 nodes. Increasing cluster size, of course, drives power consumption and cooling cost up.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

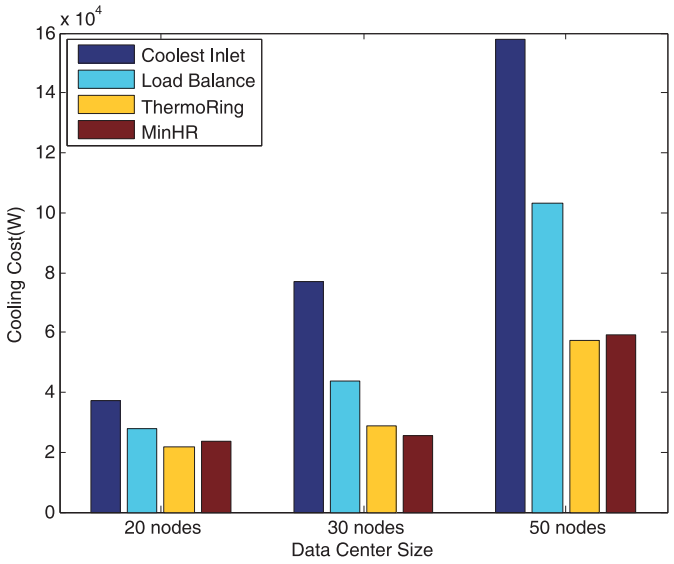


Fig. 11. Cooling costs comparison among ThermoRing, coolest inlet, load balance, and MinHR. The number of nodes is set to 20, 30, and 50, respectively.

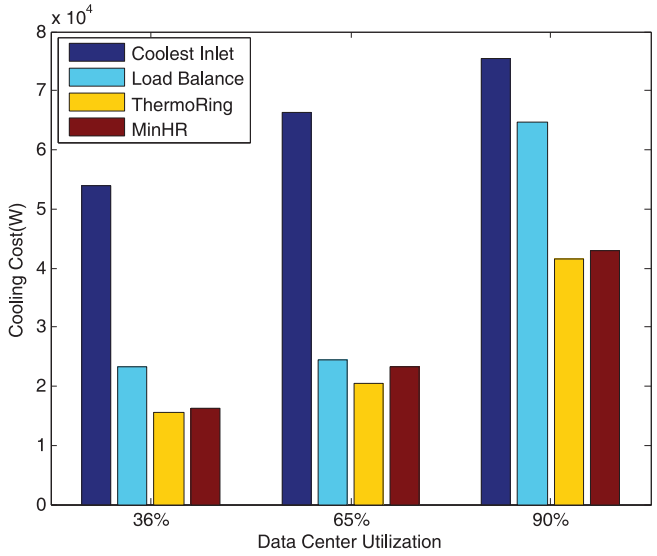


Fig. 12. Cooling costs comparison among ThermoRing, coolest inlet, load balance, and MinHR. The number of nodes is set to 50, and the system utilization is set to 36%, 65%, and 90%, respectively.

The cooling costs of coolest inlet and load balance increase sharply when the cluster size increases, because these two schemes do not take into account the heat recirculation. Compared with the alternative schemes, ThermoRing and MinHR are less sensitive to the cluster size thanks to considering the heat interference factors. ThermoRing outperforms 8% than MinHR in the 50 nodes cluster and could get large energy savings if it works a long time or the size of data center gets bigger.

Fig. 12 plots the cooling cost of the 50 nodes cluster when the system utilization is set to 36%, 65%, and 90%, respectively. The experimental results show that the cooling cost goes up when the workload becomes heavier. Again, ThermoRing achieves better cooling efficiency than load balance, coolest inlet, and MinHR under different loads. Compared with MinHR we can see ThermoRing could get the best energy-saving effects (10%) under moderate workload (65%), because it could not distribute enough tasks to different nodes to show scheduling effects in light workloads; and almost all nodes are overloaded and few scheduling strategies could be made for heavy loads.

V. Related Work

High-energy costs of data centers are contributed by power consumed by data nodes as well as cooling systems. Prior studies confirmed that cooling cost is almost half of the total energy cost of a data center [30], [31]; therefore, saving cooling energy is an important step toward reducing data center operational cost.

In the past decade, a rich set of energy-conservation techniques have been developed for large-scale data centers. These techniques generally fall into two categories, namely, node level and data center-level energy-saving techniques. Dynamic voltage and frequency scaling (DVFS) is a popular technique to save energy at the node level [32], [33]. The DVFS technique aims at changing a processor frequency according to load to minimize energy consumption. A handful of data-center-level techniques adopt the right-sizing approach, in which the size of a data center is adjusted according to workload demands [34]. This approach conserves energy by turning as many servers as possible to the power-saving mode while maintaining desirable system performance [35].

Thermal-aware workload placement strategies have been investigated in recent studies [5], which show that energy inefficiency of cooling systems largely depend on heat recirculation among nodes in a data center. As such, reducing the negative impacts of heat recirculation is an important step toward energy-efficient cooling systems. For example, Moore *et al.* [20] designed the zone-based discretization scheme that assigns tasks inversely proportional to the inlet temperature of nodes. A genetic algorithm [5] was developed to manage workload in a way to reduce the maximum inlet temperatures. These approaches can be integrated with the sever-consolidation technique to reduce the computational energy by selecting a set of active servers. Abbasi *et al.* [8] estimated workload and provided limited number of active machine according to workload, thereby reducing energy consumption. Wang *et al.* [36] proposed an energy-saving technique through a virtual server consolidation and provision scheme. The aforementioned strategies are focused on computing nodes and used a linear power model driven by CPU utilization.

VI. Conclusion

In this paper, we formulated the task-temperature dynamic model of data centers and proposed a feedback control-based thermal-aware scheduler called ThermoRing that aims at reducing cooling costs. ThermoRing copes with thermal emergencies by controlling task allocations to computing nodes. Inlet temperatures are monitored by ThermoRing during each sampling period and task allocations are adjusted based on tracked reference temperature under changing workload.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.

After implementing ThermoRing, we replayed real-world bookstore traces to evaluate the stability, performance, and energy efficiency of ThermoRing. Compared with the other two dynamic thermal management schedulers (i.e., Ad-hoc and C-Oracle-Ex) ThermoRing could get larger throughput whenever in normal cases or during emergencies while holding the peak inlet temperature under the redline temperature. In addition, ThermoRing achieves much higher cooling efficiency than those of the static schedulers (i.e., coolest inlet and load balance which have no heat recirculation consideration). It can also save much more cooling costs than MinHR (a thermal-aware scheduler) in a long-time running.

Future work will focus on integrating ThermoRing with dynamic server provision and server consolidation in the context of heterogeneous computing platforms. It would provide more energy saving space when taking into account computing power reduction with fewer active servers in light workloads, and the temperature stability could be guaranteed by ThermoRing while considering nodes' heterogeneity.

REFERENCES

[1] J. Hardcastle. *Forecast: Data Centers Worldwide in 2010-2015*. Gartner Group, Stamford, CT, 2011. [Online]. Available: http://www.gartner.com/

[2] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for Internet-scale systems," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 123–134, Aug. 2009.

[3] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, 2012, pp. 1431–1439.

[4] C. Bash and G. Forman, "HPL-2007-62 cool job allocation: Measuring the power savings of placing jobs at cooling-efficient locations in the data center," HP Lab., Palo Alto, CA, USA, Tech. Rep. HPL-2007-62, Aug. 2007.

[5] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 11, pp. 1458–1472, Nov. 2008.

[6] L. Li *et al.*, "ThermoCast: A cyber-physical forecasting model for datacenters," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, San Diego, CA, USA, 2011, pp. 1370–1378.

[7] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S. K. Gupta, and S. Rungta, "Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers," *Comput. Netw.*, vol. 53, no. 17, pp. 2888–2904, 2009.

[8] Z. Abbasi, G. Varsamopoulos, and S. K. S. Gupta, "Thermal aware server provisioning and workload distribution for Internet data centers," in *Proc. 19th ACM Int. Symp. High Perform. Distrib. Computing (HPDC)*, Chicago, IL, USA, 2010, pp. 130–141.

[9] Q. Wu, P. Juang, M. Martonosi, L.-S. Peh, and D. Clark, "Formal control techniques for power-performance management," *IEEE Micro*, vol. 25, no. 5, pp. 52–62, Sep./Oct. 2005.

[10] N. Xiong, X. Défago, X. Jia, Y. Yan, and Y. He, "Design and analysis of a self-tuning proportional and integral controller for active queue management routers to support TCP flows," in *Proc. 25th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Barcelona, Spain, 2006, pp. 1–12.

[11] N. Xiong *et al.*, "A distributed efficient flow control scheme for multirate multicast networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 9, pp. 1254–1266, Sep. 2010.

[12] C. Lu, J. Stankovic, S. Son, and G. Tao, "Feedback control real-time scheduling: Framework, modeling, and algorithms*," *Real-Time Syst.*, vol. 23, nos. 1–2, pp. 85–126, 2002.

[13] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No 'power' struggles: Coordinated multi-level power management for the data center," in *Proc. 13th Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, Seattle, WA, USA, 2008, pp. 48–59.

[14] L. Ramos and R. Bianchini, "C-oracle: Predictive thermal management for data centers," in *Proc. IEEE 14th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Salt Lake City, UT, USA, 2008, pp. 111–122.

[15] X. Wang and M. Chen, "Cluster-level feedback power control for performance optimization," in *Proc. IEEE 14th Symp. High Perform. Comput. Archit. (HPCA)*, Salt Lake City, UT, USA, 2008, pp. 101–110.

[16] X. Wang and Y. Wang, "Coordinating power control and performance management for virtualized server clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 2, pp. 245–259, Feb. 2011.

[17] Y. Wang, K. Ma, and X. Wang, "Temperature-constrained power control for chip multiprocessors with online model estimation," in *Proc. 36th Annu. Int. Symp. Comput. Archit. (ISCA)*, Austin, TX, USA, 2009, pp. 314–324.

[18] Y. Zhou, Y. Zhang, H. Liu, N. Xiong, and A. V. Vasilakos, "A bare-metal and asymmetric partitioning approach to client virtualization," *IEEE Trans. Services Comput.*, vol. 7, no. 1, pp. 40–53, Jan./Mar. 2014.

[19] C. Lu, X. Wang, and X. Koutsoukos, "Feedback utilization control in distributed real-time systems with end-to-end tasks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 6, pp. 550–561, Jun. 2005.

[20] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling 'cool': Temperature-aware workload placement in data centers," in *Proc. Annu. Conf. USENIX Annu. Tech. Conf.*, Berkeley, CA, USA, 2005, pp. 61–74.

[21] J. Moore, J. Chase, and P. Ranganathan, "Weatherman: Automated, online and predictive thermal mapping and management for data centers," in *Proc. IEEE Int. Conf. Auton. Comput. (ICAC)*, Dublin, Ireland, 2006, pp. 155–164.

[22] J. Choi *et al.*, "Modeling and managing thermal profiles of rack-mounted servers with thermostat," in *Proc. IEEE 13th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Scottsdale, AZ, USA, 2007, pp. 205–215.

[23] Q. Zhang, L. Cherkasova, and E. Smirni, "A regression-based analytic model for dynamic resource provisioning of multi-tier applications," in *Proc. 4th Int. Conf. Auton. Comput. (ICAC)*, Jacksonville, FL, USA, 2007, p. 27.

[24] W. Huang *et al.*, "TAPO: Thermal-aware power optimization techniques for servers and data centers," in *Proc. IEEE Int. Green Comput. Conf. Workshops (IGCC)*, Orlando, FL, USA, 2011, pp. 1–8.

[25] X.-F. Jiang *et al.*, "Thermal modeling and analysis of cloud data storage systems," *J. Commun.*, vol. 9, no. 4, pp. 299–311, 2014.

[26] J. M. Maciejowski, *Predictive Control With Constraints*. Harlow, U.K.: Prentice Hall, 2002.

[27] N. Xiong *et al.*, "Comparative analysis of quality of service and memory usage for adaptive failure detectors in healthcare systems," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 4, pp. 495–509, May 2009.

[28] B. Calder *et al.*, "Windows azure storage: A highly available cloud storage service with strong consistency," in *Proc. 23rd ACM Symp. Oper. Syst. Prin.*, Cascais, Portugal, 2011, pp. 143–157.

[29] D. Ford *et al.*, "Availability in globally distributed storage systems," in *Proc. 9th USENIX Conf. Oper. Syst. Design Implement. (OSDI)*, Vancouver, BC, Canada, 2010, pp. 1–7.

[30] R. Sawyer, *Calculating Total Power Requirements for Data Centers*. Amer. Power Convers., West Kingston, RI, USA, 2004.

[31] L. Li and Z. Sun, "Dynamic energy control for energy efficiency improvement of sustainable manufacturing systems using Markov decision process," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1195–1205, Sep. 2013.

[32] O. Sarood and L. Kale, "A 'cool' load balancer for parallel applications," in *Proc. Int. Conf. High Perform. Comput. Netw. Storage Anal. (SC)*, Seattle, WA, USA, 2011, pp. 1–11.

[33] N. Bansal, T. Kimbrel, and K. Pruhs, "Speed scaling to manage energy and temperature," *J. ACM*, vol. 54, no. 1, pp. 3:1–3:39, Mar. 2007.

[34] M. Lin, A. Wierman, L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," in *Proc. IEEE INFOCOM*, Shanghai, China, 2011, pp. 1098–1106.

[35] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," in *Proc. 11th Int. Joint Conf. Meas. Model. Comput. Syst. (SIGMETRICS)*, New York, NY, USA, 2009, pp. 157–168.

[36] X. Wang, Z. Du, and Y. Chen, "An adaptive model-free resource and power management approach for multi-tier cloud environments," *J. Syst. Softw.*, vol. 85, no. 5, pp. 1135–1146, 2012.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.



Xiaogang Zhao received the Ph.D. degree in computer architecture from the Huazhong University of Science and Technology, Wuhan, China, in 2009.

He is currently an Assistant Professor with the International School of Software, Wuhan University, Wuhan. His current research interests include computer architecture, storage systems, and energy-efficient systems.



Tao Peng received the Ph.D. degree in computer software from the Huazhong University of Science and Technology, Wuhan, China, in 2011.

He is currently an Associate Professor with the School of Mathematics and Computer, Wuhan Textile University, Wuhan. His current research interests include data mining and pattern recognition.



Xiao Qin (S'00–M'04–SM'09) received the B.S. and M.S. degrees in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 1992 and 1999, respectively, and the Ph.D. degree in computer science from the University of Nebraska-Lincoln, Lincoln, NE, USA, in 2004.

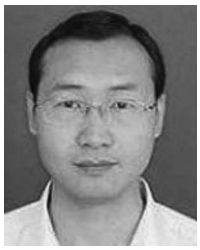
He is currently an Associate Professor with the Department of Computer Science and Software Engineering, Auburn University, Auburn, AL, USA. His current research interests include parallel and distributed systems, storage systems, fault tolerance, real-time systems, and performance evaluation.

Dr. Qin was a recipient of the U.S. National Science Foundation (NSF) Computing Processes and Artifacts Award, the NSF Computer System Research Award in 2007, and the NSF CAREER Award in 2009.



Qiping Hu received the Ph.D. degree in photogrammetry and remote sensing from Wuhan University, Wuhan, China, in 2001.

He is currently a Professor with the International School of Software, Wuhan University. His current research interests include software engineering and geographic information system.



Ling Ding received the M.S. degree in computer science from Wuhan University, Wuhan, China, in 2009, where he is currently pursuing the Ph.D. degree with the School of Computer Science.

He is an Assistant Professor with the Department of Computer Science, Hubei University of Science and Technology, Xianning, China. His current research interests include wireless sensor networks.



Zhijun Fang received the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China.

He is currently a Professor with the College of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai. His current research interests include image processing, video coding, and pattern recognition.

Dr. Fang was a recipient of the Millions of Talents Project of New Century Award in the Jiangxi Province, the Middle-Aged Academic Leader Award, and the First Young Scientists Training Plan Award in the Jiangxi Province. He is a member of the Gan Po Elite 555 Project and a Senior Member of the China Computer Federation.

Authorized licensed use limited to: Auburn University. Downloaded on November 18,2026 at 18:29:48 UTC from IEEE Xplore. Restrictions apply.