Spatio-Temporal Thermal-Aware Scheduling for Homogeneous High-Performance Computing Datacenters

Hongyang Sun[a,*], Patricia Stolf[b], Jean-Marc Pierson[b]

[a] *Ecole Normale Superieure de Lyon & INRIA, France*
[b] *IRIT, University of Toulouse, France*

Abstract

Datacenters have become an important part of today's computing infrastructure. Recent studies have shown the increasing importance of thermal considerations to achieve effective resource management. In this paper, we study thermal-aware scheduling for homogeneous high-performance computing (HPC) datacenters under a thermal model that captures both spatial and temporal correlations of the temperature evolution. We propose an online scheduling heuristic to minimize the makespan for a set of HPC applications subject to a thermal constraint. The heuristic leverages the novel notion of thermal-aware load to perform both job assignment and thermal management. To respect the temperature constraint, which is governed by a complex spatio-temporal thermal correlation, dynamic voltage and frequency scaling (DVFS) is used to regulate the job executions during runtime while dynamically balancing the loads of the servers to improve makespan. Extensive simulations are conducted based on an experimentally validated datacenter configuration and realistic parameter settings. The results show improved performance of the proposed heuristic compared to existing solutions in the literature, and demonstrate the importance of both spatial and temporal considerations. In contrast to some scheduling problems, where DVFS introduces performance-energy tradeoffs, our findings reveal the benefit of applying DVFS with both performance and energy gains in the context of spatio-temporal thermal-aware scheduling.

Keywords: HPC datacenters, thermal model, spatio-temporal correlation, thermal-aware scheduling, makespan, energy consumption, DVFS

1. Introduction

Datacenters have become an important part of today's computing infrastructure. With the ever increasing power consumption and high packing density of servers, both the heat dissipated in datacenters and the temperature have increased dramatically. High temperature is undesirable in the operation of a datacenter for several reasons: (1) It reduces the reliability of the servers. In particular, some studies have shown that the failure rate of computing nodes will double for every 10°C increase in temperature [17, 38]. (2) Increased temperature induces a larger cooling cost, which has been shown to increase nonlinearly with the temperature [27, 42]. (3) A higher temperature also leads to more leakage current, which in turn increases the static power consumption of the servers [23]. As a result, thermal management has been widely recognized as an important technique for optimizing the application performance and reducing the energy consumption in modern datacenters.

Modeling the thermal behavior of datacenters is an essential first step to the design of effective thermal management techniques. The literature contains three main approaches to characterize the thermal map of a datacenter. The first approach approximates the temperatures of the servers

*Corresponding author
Email addresses: hongyang.sun@ens-lyon.fr (Hongyang Sun), patricia.stolf@irit.fr (Patricia Stolf), jean-marc.pierson@irit.fr (Jean-Marc Pierson)

using simple *analytical models*, which are usually based on classical heat transfer laws and cyber-physical properties of the datacenters [35, 42, 45]. The second approach applies complex *predictive models*, which use more sophisticated techniques, such as machine learning and neural networks, to predict the temperatures at various locations of a datacenter [26, 49]. The last approach employs elaborate *computational fluid dynamics (CFD)* simulations to model the temperature evolution in a datacenter [2, 10, 21]. Although the CFD-based approach offers the best accuracy, it is too slow to facilitate fast and real-time decision making. In contrast, the first two approaches incur much lower overheads while offering reasonable temperature estimates, which can be validated offline by CFD simulations or calibrated online by ambient and onboard temperature sensors (if available). Hence, many researchers have relied on the first two approaches for modeling the temperature in datacenters and for designing scheduling solutions.

In this paper, we present a *spatio-temporal* analytical model to characterize the thermal behavior of datacenters, thus allowing the resource management system to make fast and online scheduling decisions in real time. Indeed, recent studies [42, 45] have shown that the server temperature in a datacenter exhibits both spatial and temporal correlations. Spatially, the inlet temperatures of the servers are related to each other via a heat-distribution matrix, which is determined by the complex airflow and heat recirculation in the datacenter. Temporally, the temperature of each server at any time is related to both its current power consumption and its historical temperature due to the physical law of cooling. Although the literature has considered either model separately (see Section 2 for details), to the best of our knowledge, no previous work has studied a holistic spatio-temporal thermal model, which is capable of providing more accurate approximations to the datacenter thermal map. Moreover, the considered spatio-temporal model does not require any knowledge or characteristic of the benchmarks, and conforms only to physical laws and datacenter configurations.

Based on this spatio-temporal model, we study a thermal-aware scheduling problem for homogeneous high-performance computing (HPC) datacenters. The objective is to minimize the makespan for a set of computation-intensive applications subject to a temperature threshold, which cannot be violated at any time during the execution. Indeed, such a threshold is imposed in many resource management systems for either energy reduction considerations or reliability concerns [27, 38]. To tackle this problem, we introduce a novel notion, called *thermal-aware load*, to capture more precisely the loads of the servers under the thermal constraint. We propose an online scheduling heuristic that applies this notion to both job assignment and thermal management aspects of the scheduling decision. For job assignment, we strategically choose the server to assign each arriving job in such a way that leads to well-balanced loads (in the thermal-aware sense) among all servers, which helps to minimize the makespan. For thermal management, we rely on dynamic voltage and frequency scaling (DVFS) to regulate the job executions during runtime for respecting the temperature threshold. To further improve the makespan, thermal-aware load is again used for prioritizing the servers while applying DVFS to cope with the complex space-time correlation of the temperature evolution. The proposed scheme guarantees to respect the thermal threshold while reducing the makespan with low computational overhead. The use of DVFS also allows the heuristic to significantly reduce the energy consumption.

We conduct extensive simulations to evaluate the effectiveness of our approach based on an experimentally validated datacenter configuration and realistic parameter settings. The results confirm that our algorithm outperforms several existing solutions in the literature, and hence demonstrate the importance of both spatial and temporal considerations in the context of thermal-aware scheduling. Finally, in contrast to some other scheduling problems, where DVFS introduces performance-energy tradeoffs, our findings reveal the benefit of applying DVFS with both performance and energy gains in the context of spatio-temporal thermal-aware scheduling.

The main contributions of this paper are summarized as follows:

- An analytical thermal model that captures both spatial and temporal behaviors of the temperature evolution in datacenter environments.
- The formulation of a spatio-temporal thermal-aware scheduling problem for high-performance computing (HPC) applications in homogeneous datacenters.

- An online scheduling heuristic that applies the notion of thermal-aware load for both job assignment and thermal management.
- A comprehensive set of simulations to demonstrate the effectiveness of the proposed heuristic under a wide range of parameter settings in a realistic datacenter configuration.

Finally, we stress once again that our proposed solution works for computation-intensive jobs and we leave the consideration of I/O-intensive and communication-intensive jobs for future work. The rest of this paper is organized as follows. Section 2 reviews some related work on thermal modeling and scheduling. Section 3 presents the spatio-temporal thermal model, based on which we formulate a thermal-aware scheduling problem in Section 4. Section 5 describes our thermal-aware scheduling heuristic. The simulation results are presented in Section 6. Finally, Section 7 concludes the paper with future directions.

2. Related Work

Many papers have studied thermal-aware scheduling in datacenters with either a spatial model or a temporal model. In this section, we review some related work on thermal modeling and scheduling. Interested readers can refer to [8] for a recent survey of the field.

2.1. Work on Spatial Correlation

To characterize the spatial correlation of the temperatures in a datacenter, Moore et al. [27] first introduced the notion of heat recirculation. They also proposed "Weatherman", a software tool to predict the thermal profile of a datacenter by taking the topology and heat flow into account [26]. Tang et al. [42] formally defined a heat-distribution matrix via an abstract heat flow model, and applied it in the optimization of the cooling cost of a datacenter. This abstract spatial model has been subsequently adopted by a series of research, and it was also successfully validated by computational fluid dynamics (CFD) simulations in [43, 36]. Pakbaznia and Pedram [31] considered the problem of minimizing the total energy consumption of a datacenter by performing server consolidation while accounting for heat recirculation. Mukherjee et al. [28] considered a similar problem while taking the temporal job placements into account (but without a temporal thermal model). Sun et al. [41] studied performance-energy tradeoff in heterogeneous datacenters with heat recirculation effect. They also proposed effective server placement strategies in order to minimize the cooling cost. The latter problem was independently studied by Pahlavan et al. [30], who utilized integer linear programming (ILP)-based methods to find the optimal location of each server in the datacenter. By assuming specific heat recirculation patterns, Mukherjee et al. [29] designed approximation algorithms for a couple of related thermal-aware scheduling problems.

2.2. Work on Temporal Correlation

The temporal temperature correlation has also attracted much attention. Ramos and Bianchini [35] presented "C-Oracle", a software infrastructure to predict the servers' temperatures in a datacenter based on simple temporal models governed by heat transfer laws. Skadron et al. [39] was the first to apply the lumped-RC model to capture the transient behavior of temperatures in processors. They also developed "HotSpot", a thermal modeling and simulation tool for microprocessor architectures [40]. This simple temporal model has then been widely adopted by a lot of subsequent research. Wang et al. [45] applied the lumped RC model to predict the temperatures of the servers in a datacenter in order to make workload placement decisions. Rajan and Yu [34] relied on the same model to maintain the temperature threshold of the system by using DVFS while maximizing the throughput. Zhang and Chatha [48] designed polynomial-time approximation schemes for the discrete version of the problem (where there is only a discrete set of DVFS levels) with the objective of minimizing the makespan. Yang et al. [46] proposed intelligent ordering of the jobs based on their thermal characteristics for reducing the number of thermal violations. Bansal et al. [6] designed competitive algorithms for the online scheduling problem of minimizing the maximum temperature of a server subject to the deadlines of the jobs. Chrobak

[11] proposed an online algorithm for the problem of maximizing the number of jobs that meet their deadlines subject to a thermal threshold.

2.3. Other Work on Thermal-Aware Scheduling

Many other papers have been devoted to thermal-aware scheduling and resource management for datacenters from different perspectives. Chavan et al. [9] proposed TIGER, a thermal-aware technique specifically designed to reduce the cooling cost due to storage systems in datacenters. Meng et al. [25] considered communication-bound HPC applications and studied joint optimization of cooling and communication costs via job allocation. Piątek [32] studied thermal-aware load balancing with fan management in air-cooled server systems in order to improve the energy efficiency. Polverini et al. [33] proposed a provably-efficient thermal-aware scheduler to dispatch jobs across geographically distributed datacenters while taking the energy consumption (including that due to cooling) into consideration. Abbasi and Gupta [1] considered a similar problem for geo-distributed datacenters, but with the additional constraint of carbon capping requirement. They proposed a predictive solution to handle the tradeoffs of energy cost and carbon footprint. Cupertino et al. [12] provided a holistic approach, considering workload and application profiles, power and cooling models, as well as resource management and scheduling policies while minimizing the energy consumption at different levels of a datacenter. Finally, Sarood et al. [37] proposed thermal-aware load balancing for HPC datacenters using DVFS and implemented the algorithm in the Charm++ runtime system. However, they considered neither heat recirculation nor the temporal temperature evolution while minimizing the cooling energy consumption.

In this paper, we consider a spatio-temporal thermal model and an induced scheduling problem by capturing both dimensions of the temperature correlation. To the best of our knowledge, this is the first time such a model is studied in a datacenter environment and used by a scheduling algorithm.

3. Spatio-Temporal Thermal Model

In this section, we present a spatio-temporal model to characterize the thermal behavior for a set $\mathcal{M} = \{M_1, M_2, \cdots, M_m\}$ of m homogeneous computing nodes (or servers) in a typical datacenter environment.

3.1. Spatial Model

Typical datacenters exhibit the *heat recirculation* phenomenon [27], where the hot air from the server outlets recirculates in the room and is mixed with the supplied cool air from the Computer Room Air Conditioning (CRAC) unit, causing the temperature at the server inlets to be higher than that of the supplied cool air. We characterize this effect by a *heat-distribution matrix* [42], which is a m-by-m matrix $\mathbf{D}$ and each element $d_{i,k} \in \mathbf{D}$ denotes the temperature increase at the inlet of server M_i per unit of power consumed by server M_k. The inlet temperature $T_i^{in}(t)$ of server M_i at time t thus satisfies:

$$T_i^{in}(t) = T_{sup} + \sum_{k=1}^{m} d_{i,k} P_k(t) \ , \tag{1}$$

where $P_k(t)$ denotes the power consumption of server M_k at time t and T_{sup} denotes the temperature of the supplied air by the CRAC unit. In this paper, we assume that T_{sup} is fixed and leave the investigation of varying the supplied air temperature (e.g., for cooling optimization) for future work.

In an ideal (but unrealistic) datacenter without any heat recirculation, the heat generated by all servers returns directly back to the CRAC unit, and the heat-distribution matrix becomes an all-zero matrix. The inlet temperature of all nodes is then the same as the supplied air temperature. In a typical datacenter, where the cool air is supplied through raised floors, previous research [27, 42] has observed that the inlet temperature of servers at upper levels of the racks is affected

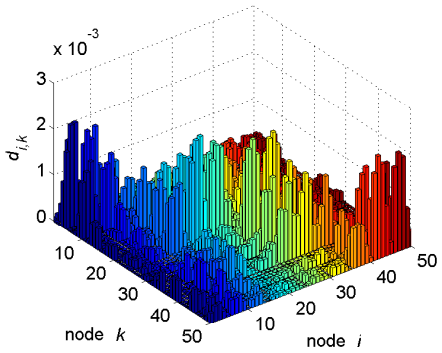


Figure 1: The heat-distribution matrix of a datacenter consisting of 50 servers.

more by the power consumption of the servers at lower levels, but not the other way around. This is because of the fact that hot air tends to move upwards. In general, the heat-distribution matrix of a datacenter depends on the pattern of air movement, which is usually fixed for a given datacenter configuration and physical layout.

Figure 1 plots the heat-distribution matrix of a datacenter consisting of 50 servers used for the experimental study in Section 6. Note that the diagonal of the matrix is not zero, meaning that the inlet temperature of a server is also affected by its own power consumption with heat recirculation.

3.2. Temporal Model

We apply the *lumped RC model* [40] to characterize the temporal behavior of a computing node's temperature. In this model, the temperature is assumed to obey the *Newton's law of cooling*, that is, its rate of decrease is proportional to the temperature difference between the node and the ambient environment, which in the case of a datacenter is the air from the server inlet. The rate of temperature increase, on the other hand, is proportional to the server's power consumption. Let $T_i(t)$ denote the temperature of node $M_i \in \mathcal{M}$ at time t. The overall rate of change for $T_i(t)$ can be described by the following ordinary differential equation:

$$\frac{\mathrm{d}T_i(t)}{\mathrm{d}t} = -\frac{1}{RC}(T_i(t) - T_i^{in}(t)) + \frac{1}{C}P_i(t) \ , \tag{2}$$

where R and C denote the *thermal resistance* and *thermal capacitance* of server M_i, respectively, $P_i(t)$ denotes the power consumption of server M_i at time t, and $T_i^{in}(t)$ denotes the inlet temperature of server M_i at time t. We assume that time is divided into equal-length intervals, which are called *time steps*, and each time step has length Δt. Solving Equation (2), we can get the temperature of node M_i at time t as follows:

$$T_i(t) = \int_{t-\Delta t}^{t} \left(\frac{P_i(t')}{C} + \frac{T^{in}(t')}{RC} \right) e^{-\frac{t-t'}{RC}} \mathrm{d}t' + T_i(t-\Delta t)e^{-\frac{\Delta t}{RC}} \ . \tag{3}$$

For simplicity, we scale the length of a time step so that $\Delta t = 1$. Now, define $f = e^{-\frac{1}{RC}}$, and suppose the inlet temperature and power consumption of node M_i are constant during time step t or interval $(t-1, t]$, and they are denoted by $T_i^{in}(t)$ and $P_i(t)$, respectively. We can then simplify $T_i(t)$ as:

$$T_i(t) = (1-f)\left(P_i(t)R + T_i^{in}(t)\right) + fT_i(t-1) \ . \tag{4}$$

According to Equation (4), the temperature of a node at any time t is affected by several factors, namely, the node's temperature at previous time $t-1$, the thermal resistance, the power consumption as well as the inlet temperature during time step t.

Figure 2 plots the temperature variation of a server during the execution of a job alongside its power consumption based on the thermal parameters described in Section 6.1. In this example,

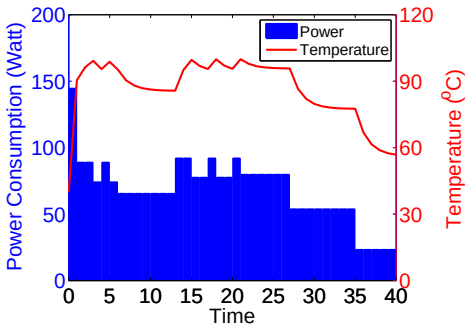


Figure 2: The temperature variation of a server with fixed inlet temperature.

the inlet temperature of the server is fixed, while, in practice, the inlet temperature could vary due to the activities of other nodes according to the spatial model described above. While the actual workload used in this example is not important, the point is to illustrate the temporal thermal model by showing the temperature evolution.

3.3. Spatio-Temporal Model

The preceding two subsections presented a spatial model and a temporal model that characterize, respectively, the thermal behaviors of two different components in a datacenter. Specifically, the spatial model describes the correlated behavior for the inlet temperatures of all the servers in the room, while the temporal model describes the temperature evolution for an individual computing node inside each server. Traditionally, these two different types of temperatures have not been considered simultaneously by the literature: the inlet temperature is often linked to cooling optimization, while the node temperature often comes as an optimization objective or constraint. Since the two phenomena co-exist in practical datacenters and are orthogonal to each other, both of them should be taken into account in order to accurately model the temperature evolution of a computing node (as the temporal behavior of a computing node depends on its inlet temperature, which is in turned spatially correlated with other servers). In this subsection, we present a spatio-temporal thermal model by combining the lumped RC model and heat recirculation model.

To this end, substituting Equation (1) into Equation (4), we get the following expression for the temperature of computing node M_i at any time t:

$$T_i(t) = (1-f)\left(P_i(t)R + T_{sup} + \sum_{k=1}^{m} d_{i,k}P_k(t)\right) + fT_i(t-1)\ . \tag{5}$$

Define the *steady-state* temperature of node M_i at time t as

$$T_i^{ss}(t) = P_i(t)R + T_{sup} + \sum_{k=1}^{m} d_{i,k}P_k(t)\ . \tag{6}$$

Then, we can write $T_i(t)$ as

$$T_i(t) = (1-f)T_i^{ss}(t) + fT_i(t-1)\ . \tag{7}$$

The thermal model described by Equation (7) essentially shows that the temperature evolution of a computing node is determined by both spatial and temporal correlations. With constant power consumption and supplied air temperature, the node's *transient* temperature will converge exponentially to the *steady state* after sufficiently long time. Due to heat recirculation, varying the power consumption of any particular node may also lead to a new steady-state temperature for all nodes.

While both spatial and temporal models have been separately studied and validated by the literature (see Section 2), a holistic thermal model that incorporates both dimensions appears to

be novel and has not been considered. Given the characteristics of the temperature evolution in a datacenter, it is mandatory to consider such a model while designing effective resource management and scheduling solutions.

4. Thermal-Aware Scheduling Problem

In this section, we consider a thermal-aware scheduling problem motivated by the spatio-temporal thermal model for the online scheduling of high-performance computing (HPC) applications in homogeneous datacenters.

4.1. Models and Objective

We consider the scheduling of computation-intensive HPC applications (such as linear algebra kernels: matrix multiplications or factorizations, etc.) in homogeneous datacenters. These applications arise in many scientific computing and machine learning domains, and can be implemented as parallel jobs that execute on servers with multiple processors. Specifically, let $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$ denote a set of n independent jobs, and let $\mathcal{M} = \{M_1, M_2, \cdots, M_m\}$ denote a set of m homogeneous servers, where each server may contain multiple processors or cores. Each server $M_i \in \mathcal{M}$ has a *static power consumption* P_{static} when it is idle. Each job $J_j \in \mathcal{J}$ is characterized by a *release time* r_j, a *processing time* (or *work*) w_{ij} and a *dynamic power consumption* p_{ij} on server M_i. Since we assume homogeneous servers, the work and power are the same on all servers, i.e., $w_{ij} = w_j$ and $p_{ij} = p_j$ for all $1 \leq i \leq m$. Jobs arrive over time in an *online* manner, and the scheduler is unaware of a job before it arrives, but the work and power characteristics of the job become known after arrival by prior profiling of the application or user estimates (e.g., based on the size of the matrix to be factorized). We assume the *moldable* scheduling model [16], in which a job upon arrival is configured to execute on all the processors of a server in parallel to minimize the execution time with better data locality. Thus, each server is assumed to be able to host only one job at any time for efficiency. Job migration is not allowed, as the cost associated with migrating the job across different servers could be expensive in HPC datacenters. However, idle time can be inserted during a job's execution, which at times is necessary to cool the server down so as to prevent its temperature from exceeding a critical threshold.

Dynamic voltage and frequency scaling (DVFS) is an effective technique to manage both power and temperature on modern architectures. The dynamic power consumption of a server is well known to be a convex function of its processing speed [7, 47]. Suppose the processing time w_j and dynamic power p_j of a job J_j are measured with respect to the *maximum speed* s_{max} of the servers, which is scaled to be 1. The dynamic power consumption of a server when executing job J_j at speed $s \in [0, 1]$ can then be approximated by $s^\alpha p_j$, where $\alpha > 1$ denotes the *power parameter* (usually 2 or 3 for CMOS-based processors), and the execution time of the job is w_j/s. Since practical systems only have a few discrete speeds to choose from, let $\mathcal{S} = \{s_0, s_1, \cdots, s_{max}\}$ denote the set of available speeds. For convenience, we include the null speed $s_0 = 0$ in $\mathcal{S}$, which corresponds to the server in the idle state without running any job. Let $x_{ij}(t) \in \{0, 1\}$ be a binary variable that takes value 1 if job J_j is executed on server M_i at time step t and 0 otherwise. Let $s_i(t) \in \mathcal{S}$ denote the processing speed of server M_i at time step t. The total power consumption of server M_i at time step t can be expressed as follows:

$$P_i(t) = P_{static} + \sum_{j=1}^{n} x_{ij}(t) s_i(t)^\alpha p_j \ . \tag{8}$$

The temperature $T_i(t)$ of the server at time t is governed by the spatio-temporal model presented in the preceding section.

The *completion time* c_j of job J_j is defined as the smallest time instance by which all the work of the job is completed. The objective is to minimize the maximum completion time of the jobs, or

the *makespan*, subject to a *temperature threshold* T_{thresh} for all servers at all time. The following shows the optimization problem:

$$\begin{aligned} &\text{minimize } C_{max} = \max_j c_j \\ &\text{subject to } T_i(t) \leq T_{thresh}, \forall i, t \end{aligned}$$

The temperature threshold is usually imposed by the datacenter resource management system for the following considerations: (1) To reduce the additional energy cost (e.g., due to cooling and linkage power) [27, 23]. (2) To reduce the failure rate of the processors for reliability concerns [38]. (3) To avoid hardware interrupt triggered by some chips when the temperature exceeds a certain redline value, which can cause severe performance degradation and energy increase [46].

4.2. Dynamic Energy Consumption

Furthermore, we consider the dynamic energy consumed by executing the jobs as an additional metric, which is defined as

$$E_{dyn} = \int_0^{C_{max}} \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}(t) s_i(t)^{\alpha} p_j \mathrm{d}t \ . \tag{9}$$

We point out that energy is not an optimization objective in this paper; it is considered simply to observe the impact of DVFS on the energy consumption, which can usually be traded off against a performance metric (e.g., makespan) in many scheduling problems. In Section 6, we make the interesting observation that makespan and energy can be improved simultaneously for the thermal-aware scheduling problem presented above. Note that we only consider the dynamic energy due to computing. For a fixed supplied air temperature T_{sup}, the literature [27, 42, 28] suggests that reduced computing energy also leads to reduce cooling energy, thus improving the total energy consumption. The quantitative optimization of cooling energy is out of scope of this paper, and we leave it as future work.

4.3. Peak Power and Thermal Characterization

Based on the previously described thermal model and scheduling model, we now define the peak power consumption of a server and derive a thermal characterization. First, since the contributions of the supplied air and static power to the servers' temperatures do not change over time, by scaling the initial steady-state temperatures of the servers, we can simplify the model by setting $T_{sup} = 0$ and $P_{static} = 0$. According to Equation (7), the temperature of server M_i at any time t then becomes

$$T_i(t) = (1-f)\left(P_i^{dyn}(t)R + \sum_{k=1}^{m} d_{i,k} P_k^{dyn}(t)\right) + fT_i(t-1) \ , \tag{10}$$

where $P_i^{dyn}(t)$ denotes the dynamic power consumption of M_i at time t. We define the peak power of any server (also known as the *thermal design power*) as follows.

Definition 1. *The* peak power P_i^{peak} *of a node M_i is the maximum dynamic power that can be consumed on the node such that the temperature threshold can be feasibly maintained, assuming the node's temperature starts at zero (e.g., by inserting sufficient idle time) and no cross interference from other nodes. From Equation (10), by setting $T_i(t-1) = 0$, $P_k^{dyn}(t) = 0$ for $\forall k \neq i$, and by enforcing $T_i(t) \leq T_{thresh}$, we can get $(1-f)\left(P_i^{dyn}(t)R + d_{i,i}P_i^{dyn}(t)\right) \leq T_{thresh}$. This leads to the following expression for the peak power (or maximum dynamic power) of node M_i:*

$$P_i^{peak} = \frac{T_{thresh}}{(1-f)(R+d_{i,i})} \ . \tag{11}$$

Thus, a job with dynamic power consumption above a server's peak power cannot possibly be scheduled using the maximum speed without violating the temperature threshold. However, the power consumptions of practical workloads rarely exceed the thermal design power of the chips. Furthermore, the temperature of an idle server cannot exceed the thermal threshold due to heat recirculation alone, because the major contribution of heat still comes from local job execution. Thus, for any server M_i at any time t, we should have $\sum_{k=1}^{m} d_{i,k} P_k^{dyn}(t) \leq T_{thresh}$ if $P_k^{dyn}(t) \leq P_k^{peak}$ for all $1 \leq k \leq m$. This implies, for any server M_i, the following property:

$$\sum_{k=1}^{m} \frac{d_{i,k}}{(1-f)(R+d_{k,k})} \leq 1 \ . \tag{12}$$

Note that Equation (12) provides a thermal characterization of a datacenter system, which should be satisfied regardless of the workloads and scheduling strategies.

5. Spatio-Temporal Thermal-Aware Scheduling

In this section, we discuss the challenge of the thermal-aware scheduling problem and propose a heuristic algorithm that consists in a job assignment policy and a thermal management policy based on the novel notion of thermal-aware load.

5.1. Challenge and Strategy

The thermal-aware scheduling problem poses a complex optimization challenge due to its thermal constraints from both temporal and spatial dimensions. The offline version of this problem contains the classical multiprocessor makespan scheduling problem, which is known to be NP-hard [18], as a special case. Hence, the thermal-aware problem is NP-hard as well. In this paper, we focus on the online version of this problem. Besides the usual constraint that job arrival information is not available to the scheduler a priori, the following describes two additional challenges faced by the design of a scheduling solution because of the thermal constraints.

- From the *temporal* point of view, the thermal threshold may prevent the (full-speed) execution of a power-intensive job on a node at certain time steps.
- From the *spatial* point of view, the execution of a local job may also be restricted by the cross interference from other nodes due to heat recirculation.

In particular, the second challenge requires the local scheduling decisions on each node to be made with a more global perspective. To cope with the challenges, we complement conventional scheduling with a thermal management policy, which uses DVFS to resolve any potential conflict that may arise when the servers execute simultaneously their jobs. Specifically, we design policies for the following two aspects of the scheduling problem.

- *Job assignment*: Decides to which server each arrived job should be assigned. The assigned jobs are maintained in a local queue $\mathcal{Q}_i$ for each server M_i. Each server has a load (commonly referred to as the sum of the work of all the jobs currently in its local queue). Since the loads are generally different on different servers due to the diversity in the jobs' work, this policy will strategically choose a server to assign each job so that more balanced loads can be achieved, which helps to minimize the makespan.
- *Thermal management*: Decides at which speed each server should execute its local jobs during each time step in order to respect the temperature threshold. This policy may reduce the speed of, or even completely idle, a server. In the latter case, the temperature threshold of the node is guaranteed to be respected because of the thermal characterization described in Section 4.3. Note that, because of the speed adjustment, the makespan is also indirectly affected by the thermal management policy.

5.2. Thermal-Aware Load

We first introduce the notion of *thermal-aware load*, based on which the scheduling decisions are performed. The following gives the relevant definitions.

Definition 2. *The* critical power P_i^{crit} *of a node* M_i *is the dynamic power that can be continuously consumed on the node, such that the steady-state temperature does not violate the temperature threshold, assuming no cross interference from other nodes. From the thermal model given by Equation (7) and by setting* $T_{sup} = 0$ *and* $T_i(t-1) = T_i(t)$*, we get*

$$P_i^{crit} = \frac{T_{thresh}}{R + d_{i,i}} \ . \tag{13}$$

Note the difference between the peak power of a node given in Equation (11) and the critical power given in Equation (13). The former may require inserting idle times during the execution in order to respect the temperature threshold, while the latter could maintain the threshold without idle times. Both quantities assume no interference from other nodes.

Definition 3. *The* critical speed s_{ij}^{crit} *of executing job* J_j *on node* M_i *is the largest available speed in* $\mathcal{S}$ *to run the job such that the critical power of the node is not exceeded, i.e.,*

$$s_{ij}^{crit} = \max\left\{s \in \mathcal{S} : s^{\alpha} p_j \leq P_i^{crit}\right\} \ , \tag{14}$$

where p_j *denotes the full-speed power consumption of job* J_j*.*

Intuitively, the critical speed represents the fastest continuous speed a job can be completed on a node. If the value s_{ij}^{crit} obtained above is 0, the critical speed is then defined as $s_{ij}^{crit} = \sqrt[\alpha]{\frac{P_i^{crit}}{p_j}}$, whose value is not in $\mathcal{S}$ but ensures that the notion of thermal-aware work below is properly defined. In this case, the critical speed can be approximated by alternating the job execution between a higher speed and a lower speed in $\mathcal{S}$.

Definition 4. *The* thermal-aware work $w_{ij}^T(t)$ *of job* J_j *on node* M_i *at any time* t *is the time to execute the remaining work of the job using the critical speed, i.e.,*

$$w_{ij}^T(t) = \frac{w_j(t)}{s_{ij}^{crit}} \ , \tag{15}$$

where $w_j(t)$ *denotes the remaining work of job* J_j *at time* t*.*

Definition 5. *The* thermal-aware load $L_i^T(t)$ *of node* M_i *at any time* t *is the sum of the thermal-aware work of all the jobs currently in its local queue, i.e.,*

$$L_i^T(t) = \sum_{J_j \in \mathcal{Q}_i} w_{ij}^T(t) \ . \tag{16}$$

In contrast to the thermal-aware load, the traditional definition of a server's load is the sum of the work (unit-speed execution time) of the jobs assigned to it. In this context, we can adapt the traditional load definition by the following one using the sum of the jobs' remaining work, and we call it *work-based load*.

Definition 6. *The* work-based load $L_i^W(t)$ *of node* M_i *at any time* t *is the sum of the remaining work of all the jobs currently in its local queue, i.e.,*

$$L_i^W(t) = \sum_{J_j \in \mathcal{Q}_i} w_j(t) \ . \tag{17}$$

Algorithm 1 Job Assignment Policy

Input: newly arrived jobs during $(t-1, t]$, and existing load $L_i(t)$ of each server $M_i \in \mathcal{M}$ at time t
Output: job assignment decision for time step $t+1$

1: **if** new jobs arrived **then**
2: **for** each arrived job J_j **do**
3: $i^* = 0$ and $L_{\min} = \infty$
4: **for** $i = 1$ to m **do**
5: **if** $L_i(t) + w_{ij}(t) < L_{\min}$ **then**
6: $L_{\min} = L_i(t) + w_{ij}(t)$ and $i^* = i$
7: **end if**
8: **end for**
9: assign job J_j to node M_{i^*}
10: update $L_{i^*}(t) = L_{i^*}(t) + w_{i^*j}(t)$
11: **end for**
12: **end if**

Note that the term "load" is used in both definitions to follow the literature convention. It also enables a generic approach to design job assignment and thermal management policies presented shortly in the next two subsections. We point out that both load definitions do not take the cross interference of the servers into account, thus can not guarantee exact prediction on the actual execution time of a job. This would require knowledge of interaction among all servers at all time steps, which is very difficult (if not impossible) to model with tractable complexity.

Compared to the work-based load, however, thermal-aware load is able to provide a more relevant measure of a server's actual load in the thermal-aware context, and therefore better approximates the time to execute the jobs without interference from the other servers. An important contribution of our thermal-aware scheduling heuristic is to apply the concept of thermal-aware load to make scheduling decisions in both job assignment (Section 5.3) and thermal management (Section 5.4). The limitation of neglecting cross interference is handled by thermal management policy, which prioritizes the servers based on their thermal-aware loads, and dynamically adjusts priorities and speed assignments during runtime by incorporating the cross interference.

5.3. Job Assignment Policy

Algorithm 1 presents the generic job assignment policy, which is invoked at time t (the beginning of time step $t+1$) for all $t \in \mathbb{Z}_{\geq 0}$. Specifically, if there are newly arrived jobs during the previous time step t, the policy assigns each new job to a server that results in the smallest cumulative load, which shares the same spirit as the well-known *LIST* scheduling algorithm [19] for homogeneous servers or the *HEFT* scheduling algorithm [44] for heterogeneous servers. Job assignment policy is also commonly referred to as load balancing policy in the literature.

An interesting feature of the policy lies in its generality in a sense that different notions of "load" can be applied for job assignment. Depending on the specified "load", the part of the pseudocode highlighted in red (Lines 5, 6 and 10) can be replaced by the corresponding definition. For example, in case of thermal-aware load, we have $L_i(t) = L_i^T(t)$ and $w_{ij}(t) = w_{ij}^T(t)$. If work-based load is used instead, we have $L_i(t) = L_i^W(t)$ and $w_{ij}(t) = w_j(t)$.

The complexity of assigning each job is linear in the number m of servers. If thermal-aware load is used, the complexity is $O(m \log K)$, where $K = |\mathcal{S}|$ denotes the number of available speeds in $\mathcal{S}$. This is due to the computation of the job's critical speed on each server by performing a binary search on the list of speeds.

5.4. Thermal Management Policy

Algorithm 2 shows the thermal management policy, which applies DVFS to regulate the temperatures of the servers during their execution. First, the servers are prioritized according to their current loads: heavier load implies higher priority. In the pseudocode (Line 1), $\gamma(\cdot)$ denotes a permutation of the servers sorted in the non-increasing order of load at time t. Again, as in the

Algorithm 2 Thermal Management Policy

Input: local queue $\mathcal{Q}_i$, temperature $T_i(t)$ and load $L_i(t)$ of each server $M_i \in \mathcal{M}$ at time t
Output: thermal management decision for time step $t+1$

1: sort the servers in non-increasing order of load, i.e., $L_{\gamma(1)}(t) \geq L_{\gamma(2)}(t) \geq \cdots \geq L_{\gamma(m)}(t)$
2: **for** $i = 1$ to m **do**
3: compute $\widehat{T}_i(t+1) = \frac{T_{thresh} - f \cdot T_i(t)}{1-f}$
4: **end for**
5: **for** $i = 1$ to m **do**
6: **if** $\mathcal{Q}_{\gamma(i)} \neq \emptyset$ **then**
7: compute server $M_{\gamma(i)}$'s power slack $\widehat{P}_{\gamma(i)}(t+1) = \min\left(\frac{\widehat{T}_{\gamma(1)}(t+1)}{d_{\gamma(1),\gamma(i)}}, \cdots, \frac{\widehat{T}_{\gamma(i-1)}(t+1)}{d_{\gamma(i-1),\gamma(i)}}, \frac{\widehat{T}_{\gamma(i)}(t+1)}{R+d_{\gamma(i),\gamma(i)}}\right)$
8: find largest speed $s \in \mathcal{S}$ that satisfies $s \leq \sqrt[\alpha]{\frac{\widehat{P}_{\gamma(i)}(t+1)}{p_j}}$, where p_j is the full-speed power consumption of the first job $J_j \in \mathcal{Q}_{\gamma(i)}$, and set $s_{\gamma(i)}(t+1) = s$
9: update $\widehat{T}_{\gamma(i)}(t+1) = \widehat{T}_{\gamma(i)}(t+1) - s^\alpha p_j R$
10: **for** $k = 1$ to m **do**
11: update $\widehat{T}_k(t+1) = \widehat{T}_k(t+1) - s^\alpha p_j d_{k,\gamma(i)}$
12: **end for**
13: **end if**
14: **end for**

job assignment policy, the notion of "load" is also generic here: either thermal-aware load $L^T_{\gamma(i)}(t)$ or work-based load $L^W_{\gamma(i)}(t)$ can be applied (highlighted in red).

In the event of potential conflict, the policy will try to maintain faster speed for servers with higher priority (thus heavier load) while reducing the speed of low-priority servers. Intuitively, this strategy provides better dynamic load balancing during runtime so as to minimize the overall execution time. Specifically, the strategy relies on the following concepts.

Definition 7. *The* temperature slack $\widehat{T}_i(t)$ *of node* M_i *in any time step* t *is the remaining steady-state temperature allowed on the node so as to respect the temperature threshold.*

Definition 8. *The* power slack $\widehat{P}_i(t)$ *of node* M_i *in any time step* t *is the remaining power consumption allowed on the node so as to respect the temperature slacks of all nodes.*

For each node M_i, the algorithm computes its temperature slack $\widehat{T}_i(t+1)$ for the next time step $t+1$ based on the temporal thermal model (Lines 2-4). In particular, by solving $T_i(t+1) \leq T_{thresh}$ from Equation (7), we can get

$$T_i^{ss}(t+1) \leq \frac{T_{thresh} - f \cdot T_i(t)}{1-f} = \widehat{T}_i(t+1) \ . \tag{18}$$

Since the steady-state temperature is related to the power consumptions of all servers based on the spatial thermal model (see Equation (6)), the temperature slack of a server will drop as jobs start to be assigned. The goal is to maximize the execution speeds (so as to maximize the throughput), while keeping the temperature slacks nonnegative for all servers. To this end, servers are considered one by one in the prioritized order, so that a high-priority server receives speed assignment first, which helps to dynamically balance the loads of all servers during runtime. Hence, it is possible for a high-priority server to delay the execution of a low-priority one, but not the other way around.

For each server $M_{\gamma(i)}$, the algorithm computes its power slack $\widehat{P}_{\gamma(i)}(t+1)$ (Line 7), based on the temperature slack of the server itself as well as those of the higher priority servers, i.e., $M_{\gamma(1)}, \cdots, M_{\gamma(i-1)}$. These servers are considered because they have already received speed assignments, and we need to ensure that their temperature slacks will not become negative due to the new speed assignment for server $M_{\gamma(i)}$. The lower priority servers, i.e., $M_{\gamma(i+1)}, \cdots, M_{\gamma(m)}$ need not be considered, since no speed has been assigned to them yet for this time step. Hence, the characteristic of the system (Equation (12)) guarantees that their temperatures will not exceed the threshold (or equivalently, their temperature slacks will not be negative).

From the spatio-temporal thermal model (Equation (6)), we know that the power consumption of server $M_{\gamma(i)}$ affects its own temperature slack through thermal resistance R and the temperature slacks of all the nodes (including itself) through heat-distribution matrix D. Therefore, to keep the thermal slacks nonnegative, the power slack $\widehat{P}_{\gamma(i)}(t+1)$ must satisfy the following constraints simultaneously:

$$\widehat{T}_{\gamma(i)}(t+1) \geq \left(R + d_{\gamma(i),\gamma(i)}\right) \widehat{P}_{\gamma(i)}(t+1) \ , \tag{19}$$

$$\widehat{T}_{\gamma(k)}(t+1) \geq d_{\gamma(k),\gamma(i)} \widehat{P}_{\gamma(i)}(t+1), \quad \forall 1 \leq k < i \ . \tag{20}$$

Hence, the power slack $\widehat{P}_{\gamma(i)}(t+1)$ is given by

$$\widehat{P}_{\gamma(i)}(t+1) = \min \left(\frac{\widehat{T}_{\gamma(1)}(t+1)}{d_{\gamma(1),\gamma(i)}}, \frac{\widehat{T}_{\gamma(2)}(t+1)}{d_{\gamma(2),\gamma(i)}}, \cdots, \frac{\widehat{T}_{\gamma(i-1)}(t+1)}{d_{\gamma(i-1),\gamma(i)}}, \frac{\widehat{T}_{\gamma(i)}(t+1)}{R + d_{\gamma(i),\gamma(i)}} \right) . \tag{21}$$

To execute the current job in the local queue $\mathcal{Q}_{\gamma(i)}$, the processing speed of server $M_{\gamma(i)}$ is then set to be the largest one in the set of available speeds such that the resulting power consumption does not exceed the power slack (Line 8). Finally, the temperature slacks are updated by considering the contributions of actual power consumption of server $M_{\gamma(i)}$ to itself by convection (Line 9) and to all the nodes by heat recirculation (Lines 10-12).

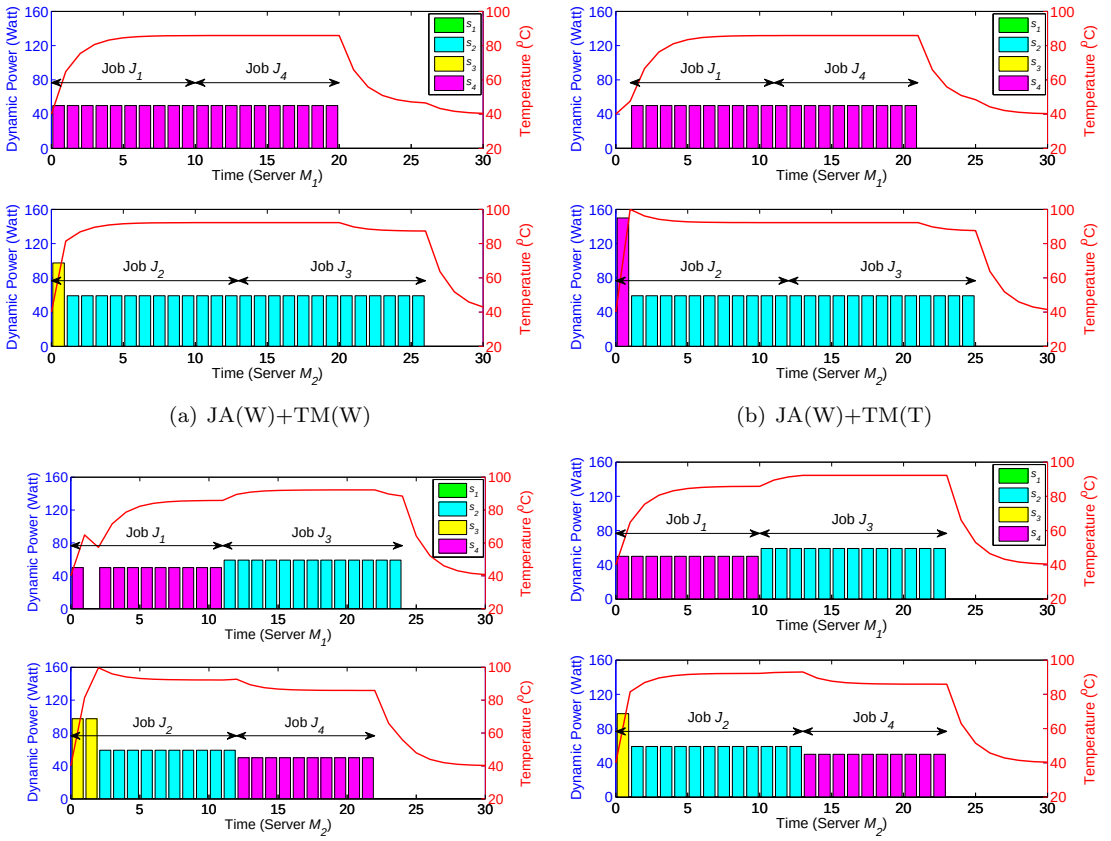
The complexity of this procedure is dominated by the computation of power slack and the update of temperature slack for each server, which takes $O(m^2)$ time. As a typical time step in HPC servers is in the order of milliseconds or seconds, the overhead to compute the scheduling decisions is in practice negligible in front of the job executions for current common datacenters.

Remark. We point out that both job assignment and thermal management policies presented above can be generally applied to processors without DVFS. In this case, the set $\mathcal{S}$ of available speeds contains only two values, i.e., $\{0, 1\}$. Hence, each server either executes a job with the full speed or is left completely idle during a time step. The dynamic energy consumption will not be affected by the scheduling heuristics, and is simply given by the total dynamic energy of all jobs executed in full speed, i.e., $E_{dyn}^{full} = \sum_{j=1}^{n} w_j p_j$.

5.5. An Illustrating Example

We now use a simple example to illustrate the performance of the thermal-aware scheduling heuristic. This example applies the same experimental settings presented in Section 6.1. Specifically, the effective temperature threshold is $T_{thresh} = 60°\text{C}$ (after discounting the static temperature), the thermal parameters are $R = 0.7$, $f = 0.5$, and there are 4 non-idle speeds for the servers: $s_1 = 0.6, s_2 = 0.733, s_3 = 0.866, s_4 = 1$. However, we consider only two servers $\mathcal{M} = \{M_1, M_2\}$ and a 2×2 heat-distribution matrix whose elements are all 0.1. We point out that practical datacenters should have more servers and the values of the heat-distribution matrix are usually much smaller (see Figure 1). Here, we consider two servers for the ease of illustration and the values of the matrix are amplified to demonstrate the cumulative effect of cross interference from multiple servers. A set of four jobs $\mathcal{J} = \{J_1, J_2, J_3, J_4\}$ are released one after another, with work $w_1 = 10$, $w_2 = 9$, $w_3 = 9$, $w_4 = 10$, and dynamic power $p_1 = 50$, $p_2 = 150$, $p_3 = 150$, $p_4 = 50$.

To better understand the scheduling framework, we apply both thermal-aware load and work-based load in the job assignment (JA) and thermal management (TM) policies. This leads to four heuristics, namely, JA(W)+TM(W), JA(W)+TM(T), JA(T)+TM(W), JA(T)+TM(T), where "W" means work-based load is used in the corresponding policy and "T" means thermal-aware load is used. Figure 3 depicts the scheduling decisions made by each heuristic, together with the execution speed, dynamic power consumption and temperature of each server at any time during execution. In this example, we assume that all jobs must start at an integer time instance. We note that, due to the relatively low power consumption of jobs J_1 and J_4, their initial thermal-aware load at time 0 is the same as the work-based load of 10. For jobs J_2 and J_3, although they have a smaller work-based load of 9, because of the high power consumption, the



(a) JA(W)+TM(W)

(b) JA(W)+TM(T)

(c) JA(T)+TM(W)

(d) JA(T)+TM(T)

Figure 3: A simple example with two servers and four jobs to illustrate the performance of four different scheduling heuristics. The red line in each figure represents the evolution of temperature with time, and the bars represent the power consumption. The color of each bar corresponds to the speed used at that time step.

thermal-aware load is larger, which is more than 12.[1] All heuristics assign J_1 to M_1 and then J_2 to M_2. Their differences are in the assignments of J_3 and J_4, as well as the priorities of M_1 and M_2 at runtime. The following explains the decisions made by each of the four heuristics as well as the calculations of the runtime parameters (i.e., temperature slack, power slack).

JA(W)+TM(W). Figure 3(a) shows that this heuristic assigns J_3 to M_2, which has a smaller work-based load at the time of assignment, and finally assigns J_4 to M_1. During the initial execution (i.e., the first step), higher priority is given to M_1 due to its higher work-based load (10+10 = 20 v.s. 9+9=18), despite the fact that M_2 has a higher thermal-aware load (10+10=20 v.s. 12.28+12.28 = 24.56). According to Algorithm 2, the temperature slacks of the two servers are first calculated as $\widehat{T}_1(1) = \widehat{T}_2(1) = \frac{T_{thresh}}{1-f} = \frac{60}{0.5} = 120$. The power slack of M_1 is then calculated as $\widehat{P}_1(1) = \frac{\widehat{T}_1(1)}{R+d_{1,1}} = \frac{120}{0.7+0.1} = 150$ and its execution speed is chosen to be $s_1(1) = 1 < \sqrt[3]{\frac{\widehat{P}_1(1)}{p_1}} = \sqrt[3]{\frac{150}{50}} = \sqrt[3]{3}$. After that, the temperature slacks of the two servers are updated as $\widehat{T}_1(1) = \widehat{T}_1(1) - s_1(1)^3 p_1(R+d_{1,1}) =$

[1]The thermal-aware loads $w_{11}^T(0)$ and $w_{12}^T(0)$ of jobs J_1 and J_2 on server M_1 at time $t = 0$ are computed as follows. From Definition 2, the critical power of M_1 is given by $P_1^{crit} = \frac{60}{0.7+0.1} = 75$. Given the two jobs' power consumptions $p_1 = 50$ and $p_2 = 150$ as well as the set of available speeds $\{0, 0.6, 0.733, 0.866, 1\}$, the critical speeds of J_1 and J_2 on M_1 can be shown to be $s_{11}^{crit} = 1$ and $s_{12}^{crit} = 0.733$, respectively, according to Definition 3. The initial thermal-aware loads of the two jobs on M_1 are therefore computed as $w_{11}^T(0) = \frac{10}{1} = 10$ and $w_{12}^T(0) = \frac{9}{0.733} \approx 12.28$ based on Definition 4. The corresponding values for the other job-server pairs can be similarly computed.

$120 - 50 \cdot (0.7 + 0.1) = 80$ and $\widehat{T}_2(1) = \widehat{T}_2(1) - s_1(1)^3 p_1 d_{2,1} = 120 - 50 \cdot 0.1 = 115$. The power slack of M_2 is then calculated as $\widehat{P}_2(1) = \min\left(\frac{\widehat{T}_1(1)}{d_{1,2}}, \frac{\widehat{T}_2(1)}{R+d_{2,2}}\right) = \min\left(\frac{80}{0.1}, \frac{115}{0.7+0.1}\right) = 143.75$, and its execution speed is chose to be $s_2(1) = 0.866 < \sqrt[3]{\frac{\widehat{P}_2(1)}{p_2}} = \sqrt[3]{\frac{143.75}{150}} \approx 0.986$. Hence, M_2 executes with a lower speed rather than at the full speed during the first time step. Since all decisions are made without considering thermal-aware load, this heuristic leads to a fairly unbalanced completion time between the two servers, and results in a makespan of 26.

JA(W)+TM(T). This heuristic, as shown in Figure 3(b), makes the same job assignment decision as JA(W)+TM(W), but runs server M_2 with higher priority, due to its consideration of thermal-aware load for thermal management. This is reflected by the full execution speed and high power consumption of M_2 at the first step of the execution, which raises the temperature of the server to the threshold. The following shows the detailed calculation of the runtime parameters during the first step. As in the previous heuristic, the initial temperature slacks of the two servers are $\widehat{T}_1(1) = \widehat{T}_2(1) = 120$. The power slack of M_2 is first calculated as $\widehat{P}_2(1) = \frac{\widehat{T}_2(1)}{R+d_{2,2}} = \frac{120}{0.7+0.1} = 150$ and its execution speed is $s_2(1) = 1 \leq \sqrt[3]{\frac{\widehat{P}_2(1)}{p_2}} = \sqrt[3]{\frac{150}{150}} = 1$. The temperature slacks of the two servers are then updated as $\widehat{T}_1(1) = \widehat{T}_1(1) - s_2(1)^3 p_2 d_{1,2} = 120 - 150 \cdot 0.1 = 105$ and $\widehat{T}_2(1) = \widehat{T}_2(1) - s_2(1)^3 p_2 (R + d_{2,2}) = 120 - 150 \cdot (0.7 + 0.1) = 0$. The power slack of M_1 becomes $\widehat{P}_1(1) = \min\left(\frac{\widehat{T}_1(1)}{R+d_{1,1}}, \frac{\widehat{T}_2(1)}{d_{2,1}}\right) = \min\left(\frac{105}{0.7+0.1}, \frac{0}{0.1}\right) = 0$, so its execution speed is $s_1(1) = 0$. As a result, M_1 is not able to run any job during this step and has to be left idle; otherwise the thermal threshold of M_2 will be violated due to cross interference. Compared to the previous heuristic, the makespan of this heuristic is improved to 25.

JA(T)+TM(W). Figure 3(c) shows that this heuristic further improves the makespan to 24, because it achieves a better balancing of the loads of the two servers by considering thermal-aware load in the first place. In particular, J_3 is assigned to M_1 instead of M_2, thus avoiding executing two jobs with high thermal-aware load on the same server. The execution of the first step turns out to be the same as the JA(W)+TM(W) heuristic, and the temperatures of the two servers at the end of the first step are given by $T_1(1) = 24.87$ and $T_2(1) = 41.47$ (again, after discounting the static temperature of 40°C). However, this heuristic uses work-based load for thermal management, so it gives higher priority to M_2 after the first step as M_2 has a higher work-based load remaining (18 v.s. 18.134). The temperature slacks, power slacks and execution speeds of the two servers at the second step can be calculated similarly as before and are omitted here. Note that, due to cross interference, an idle slot has to be inserted at the second execution step of M_1, because the temperature slack $T_2(2)$ and power slack $P_2(2)$ of M_2 at that step are again both 0. This results in delayed overall execution, as M_1 in fact has a larger thermal-aware load at this point.

JA(T)+TM(T). The above problem of the JA(T)+TM(W) heuristic can be avoided by utilizing thermal-aware load to make both scheduling decisions, as shown in Figure 3(d). In particular, after the first time step, although M_1 has completed more work, it has a larger remaining thermal-aware load. Hence, JA(T)+TM(T) gives higher priority to M_1, which avoids delaying its execution and eventually leads to the best makespan of 23 among the four heuristics.

This example, albeit simple, confirms the superiority of thermal-aware load in contrast to the work-based load for making job assignment and thermal management decisions. It also demonstrates the importance of spatio-temporal awareness as considered in Algorithm 2 for meeting temperature constraint in the thermal-aware scheduling context. The next section will rely on simulations to further evaluate the performance of these heuristics at a larger scale.

6. Performance Evaluation

In this section, we conduct simulations to evaluate the performance of the proposed thermal-aware scheduling algorithm and to compare it with the traditional work-based scheduling under

various parameter settings.

6.1. Simulation Settings

We simulate a datacenter with $m = 50$ servers using a heat-distribution matrix shown in Figure 1, which has been experimentally validated [43] and used in several previous studies [42, 28, 41]. Following the settings in [24], the processors inside the servers have five discrete (including idle) speeds, which are normalized to be $\mathcal{S} = \{0, 0.6, 0.733, 0.866, 1\}$. The power parameter is set to be $\alpha = 3$ according to the well-adopted *cube-root* model [7]. The thermal resistance (with unit °C/Watt) depends on the cooling and packaging of the processors, and is typically in the range of 0.3-1.5 [48]. We set it to be $R = 0.7$. The thermal capacitance (with unit Joule/°C) is proportional to the thickness and area of the processor die [40]. The literature usually considers the *RC constant*, which is in the order of milliseconds or seconds [40, 46]. Since the length of a time step is in the same order, we set the factor f defined in Equation (4) to be $f = e^{-\frac{\Delta t}{RC}} = 0.5$. This is consistent with the values observed in the literature [40, 46].

Following the guidelines of American Society of Heating, Refrigerating, and Air-Conditioning Engineers (ASHRAE) [5], the supplied air temperature is set to be $T_{sup} = 25$°C. The maximum temperature of the chips (also known as *junction temperature*) is between 85°C and 100°C as shown in [14], and we set it to be 100°C. The static power of the processors varies from chip to chip, but it is typically in the range of 10-30Watt. With the chosen thermal resistance, it contributes 15°C to the temperature of each processor. Therefore, the temperature of an idle server is 25°C+15°C = 40°C. According to the simplification of the model (Section 4.3), the temperature threshold after discounting the static temperature is given by $T_{thresh} = 100$°C − 40°C = 60°C.

The processing time of the jobs follows exponential distribution, which is commonly used to model service demands in computer systems [22, 15]. In our experiment, the mean processing time is set to be 300 time steps. Suppose that each time step takes milliseconds or seconds to run (incorporating the overhead to compute the scheduling decisions and to adjust the processor speeds). The average processing time ranges from a few seconds to tens of minutes, corresponding to the length of typical HPC applications. The dynamic power consumption of the jobs is uniformly distributed in $(0, P_{peak})$, where $P_{peak} = \min_i P_i^{peak}$ is derived from Equation (11) with the thermal parameters defined above.

In the simulations, we will also vary these parameters to study their impacts. Specifically, from Section 6.3.2 to Section 6.3.5, we change the processing time distributions and power consumption ranges, adjust the DVFS settings and heat-distribution matrices, and vary the thermal parameters (R and f) in order to evaluate their impacts on the performance of the scheduling heuristics.

6.2. Evaluated Heuristics

In the simulations, we evaluate and compare the performance of four heuristics mentioned in Section 5.5, as well as two other heuristics that are commonly applied in the literature. The following describes the evaluated heuristics in detail.

- *JA(W)+TM(W)*: Makes both job assignment and thermal management decisions using work-based load.
- *JA(W)+TM(T)*: Makes job assignment decision using work-based load and makes thermal management decision using thermal-aware load.
- *JA(T)+TM(W)*: Makes job assignment decision using thermal-aware load and makes thermal management decision using work-based load.
- *JA(T)+TM(T)*: Makes both job assignment and thermal management decisions using thermal-aware load.
- *RR+Random*: Uses the Round Robin (RR) policy for job assignment, and uses the Random policy for thermal management, i.e., by prioritizing the servers in an arbitrary/random order.
- *Coolest*: Makes both job assignment and thermal management decisions by favoring the server whose current temperature is the lowest [27, 42, 41].

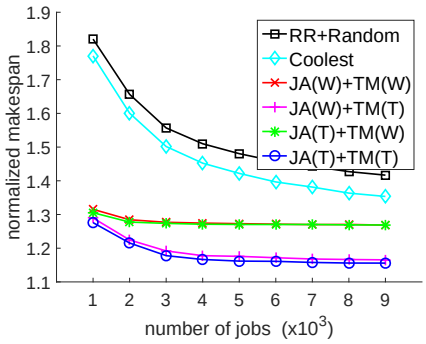


Figure 4: Performance comparison of various scheduling heuristics under different number of jobs.

6.3. Simulation Results

We now present the simulation results, all of which are obtained by averaging 50 runs with different work and power values randomly generated from the respective distributions.

6.3.1. Performance comparison of different heuristics

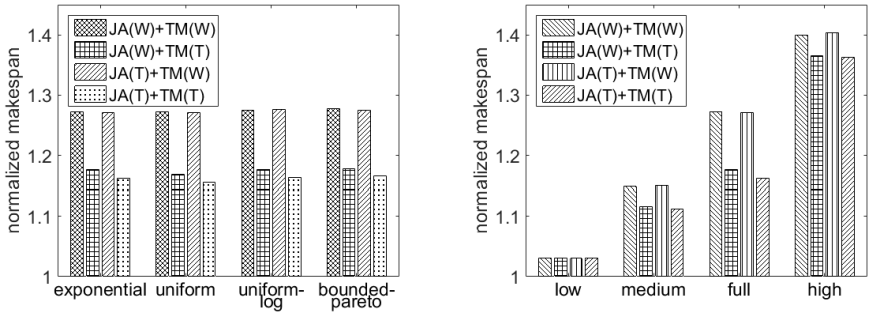
Figure 4 presents the simulation results as the number of jobs is varied from 1000 to 9000. The makespan is normalized by $C^{lb}_{max} = \frac{1}{m}\sum_{j=1}^{n} w_j$, which is a theoretical lower bound even when all jobs are executed at full speed. First, we can see that the two heuristics RR+Random and Coolest perform much worse than the other heuristics in terms of makespan. This is because RR+Random does not consider the actual (thermal-aware) loads of the servers when making both scheduling decisions, and Coolest considers only the current temperatures of the servers but ignores the balancing of the system load in the long run. As a result, both heuristics lead to very unbalanced loads among the servers, and thus fare poorly for makespan. In contrast, the other four heuristics have much better performance. Specifically, the normalized makespan of the thermal-aware heuristic JA(T)+TM(T) improves with the load of the system, while the work-based heuristic JA(W)+TM(W) is less sensitive to the load. In particular, the thermal-aware heuristic improves makespan by up to 10% at medium to heavy load, most of which is due to thermal management. *The results confirm the advantage of using thermal-aware load as the load indicator, especially when making dynamic thermal management decisions.*

In the subsequent experiments, we will not consider the RR+Random and Coolest heuristics, due to their poor performance. Furthermore, we will only focus on the medium workload scenario with 5000 jobs.

6.3.2. Sensitivity to processing time distribution and power consumption range

Some studies that analyze real workload logs in supercomputing centers have shown that the processing time distributions of some parallel jobs are in fact heavy-tailed [15]. Therefore, besides the exponential distribution, we test the sensitivity of the results by using a heavy-tail distribution as well as two other distributions that appeared in the scheduling literature for the job processing times. Specifically, the following distributions are used in the experiments: *uniform* [4, 3], *uniform-log* [13, 15] and *bounded-pareto* (heavy-tail) [20, 3, 15]. We experiment with these distributions with the same mean processing time of 300, while setting the lower bound to 60, the upper bound to 1200, and the pareto index to 3 for bounded-pareto distribution. Furthermore, we also experiment with jobs that have different power consumption ranges. Specifically, besides the full range $(0, P_{peak})$, we also consider the low range $(0, P_{crit})$, the high range (P_{crit}, P_{peak}), and the medium range $\left(\frac{P_{crit}}{2}, \frac{P_{crit}+P_{peak}}{2}\right)$, where $P_{crit} = \frac{1}{m}\sum_{i=1}^{m} P_i^{crit}$ denotes the average critical power of all nodes.

Figure 5 shows the normalized makespan of four heuristics. We can see that their relative performance is barely affected by the processing time distribution, and the thermal-aware heuristic maintains its advantage for all distributions. The power consumption range has a more interesting



(a) Job work distribution

(b) Power consumption range

Figure 5: Impact of job work distributions and power consumption ranges on the makespan of different heuristics.

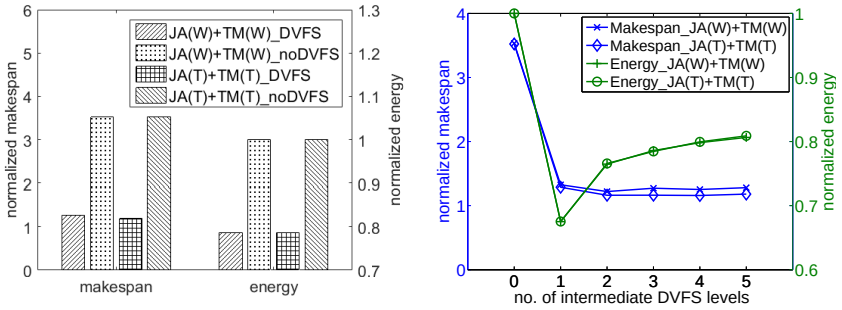
impact on the makespan. For the low range, the power of the jobs is small enough such that they can be safely executed even without thermal management. In fact, the critical speeds of all the jobs in this case become the full speed of the servers, and the thermal-aware load is equivalent to the work-based load. Thus, all heuristics have the same performance, which is very close to the theoretical lower bound. When the jobs' power is in the medium or high range, the critical speeds of the jobs are reduced, since thermal management becomes essential to maintaining the temperature threshold. Naturally, the makespan of all heuristics degrades. Compared to the full range, however, the performance gap among different heuristics is also smaller. The reason is because the critical speeds of the jobs, although reduced, fall again in a similar range, which in turn makes the ratios of thermal-aware loads similar to those of work-based loads. *The results show that the thermal-aware heuristic offers the best performance when the jobs exhibit a large variation on the power consumption.*

In the rest of the experiments, we will not consider the two mixed heuristics JA(W)+TM(T) and JA(T)+TM(W). Instead, we will just focus on the purely work-based heuristic JA(W)+TM(W) and purely thermal-aware heuristic JA(T)+TM(T) in order to demonstrate the advantage of using thermal-aware load under other parameter settings.

6.3.3. Impact of DVFS

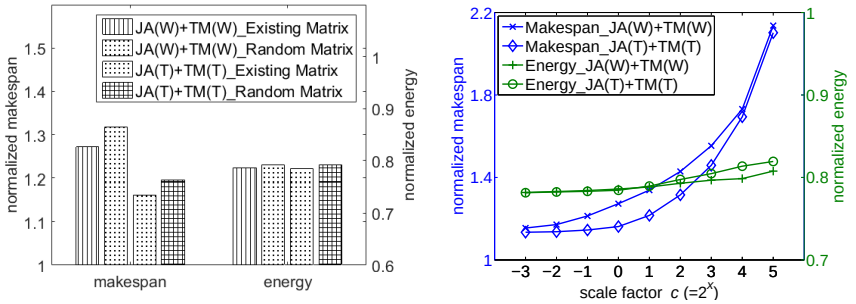
We study how DVFS affects the makespan as well as the energy consumption, which is normalized by the total dynamic energy $E_{dyn}^{full} = \sum_{j=1}^{n} w_j p_j$ of all jobs when executed at full speed. Figure 6(a) shows the result for JA(W)+TM(W) and JA(T)+TM(T) with and without using DVFS. For both heuristics, using DVFS improves the makespan by more than 65% and at the same time improves the dynamic energy by about 20%. In contrast to many scheduling problems, where the use of DVFS results in a tradeoff between performance and dynamic energy, the thermal-aware scheduling problem benefits from DVFS for the two otherwise conflicting objectives. The reason is because, by reducing the execution speeds, DVFS enables the computing nodes to respect the thermal threshold, to decrease the energy consumption, and at the same time to provide better dynamic load balancing, which directly translates to improvement in makespan. *The results reinforce the usefulness of DVFS in the context of thermal-aware scheduling with simultaneous performance and energy gains.*

In another experiment, we change the number of intermediate DVFS levels between the null speed 0 and full speed 1, and observe its impact on the performance. To be coherent with the existing setting, the lowest non-zero speed is always set to be 0.6, and the different intermediate speeds are equally spaced in $[0.6, 1]$. Note that the existing setting, i.e., $\{0, 0.6, 0.733, 0.866, 1\}$, corresponds to having 3 intermediate DVFS levels, and having 0 intermediate level means that DVFS is not used. We can see from Figure 6(b) that the makespan improves dramatically by having at least two DVFS levels due to the flexibility to execute the jobs with higher intermediate speeds but not at the full speed. Having additional DVFS levels, however, does not seem to improve the makespan further. This is because, as more speed levels are available, higher execution speeds



(a) (b)

Figure 6: Impact of DVFS on the makespan and energy of thermal-aware heuristic JA(T)+TM(T) and work-based heuristic JA(W)+TM(W).



(a) (b)

Figure 7: Impact of heat-distribution matrix on the makespan and energy of thermal-aware heuristic JA(T)+TM(T) and work-based heuristic JA(W)+TM(W).

tend to be used. And when the speeds become sufficiently high, the cross interference among the servers as well as the temporal correlation of the servers' temperature will prevent the jobs from being executed continuously under such high speeds. Hence, idle times or lower execution speeds must be inserted in order to respect the thermal threshold, which adversely affects the makespan. On the other hand, the energy consumption also improves significantly by starting to use DVFS, but after that, it increases with the number of DVFS levels. Again, this is because more jobs can be feasibly executed with higher intermediate speeds, and using higher speeds means consuming more energy due to the convexity of the power function.

6.3.4. Impact of heat-distribution matrix

We now study the impact of heat-distribution matrix on the performance of the scheduling heuristics. First, we compare the experimentally validated matrix from [43] with a random matrix, which has the same average cross-interference factor $d_{mean} = \frac{1}{m^2}\sum_{i=1}^{m}\sum_{k=1}^{m} d_{i,k}$, but whose elements are uniformly generated in the range $[0, 2d_{mean}]$. Figure 7(a) shows the result. We can see that the random matrix renders a slightly worse performance for both heuristics, but the relative performance of the two heuristics are unaffected by the matrix of choice.

Next, we vary the original matrix by scaling its elements by a factor of c, which varies from 1/8 to 32, thus representing different levels of cross interference in the datacenter. The result is shown in Figure 7(b). When the cross interference is weak (small c), thermal management becomes less important in the scheduling decision, which according to the discussion in Section 6.3.1 is a key policy to influence the makespan. Hence, both heuristics have similar performance. When the cross interference is too strong (large c), thermal-aware load is no longer sufficient to characterize the priorities of the servers. Hence, the makespan of both heuristics is again similar and the energy

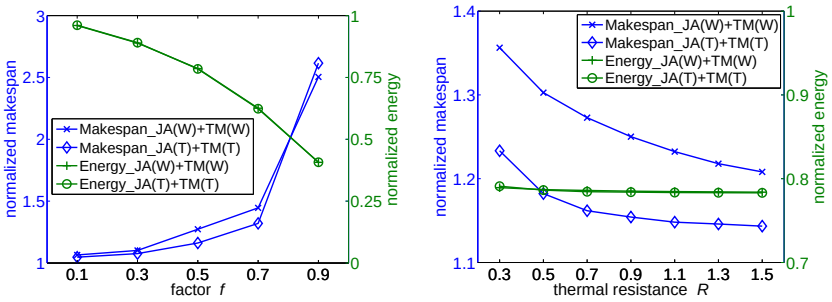


Figure 8: Impact of factor f and thermal resistance R on the makespan and energy of thermal-aware heuristic JA(T)+TM(T) and work-based heuristic JA(W)+TM(W).

increases due to increased makespan. *The result demonstrates that the thermal-aware heuristic has the best relative performance compared to the work-based one with moderate levels of cross-interference among servers, which is practically the case in most realistic datacenters.*

6.3.5. Impact of thermal parameters

Finally, We study the impact the thermal parameters on the performance of the scheduling heuristics. Specifically, we vary factor f from 0.1 to 0.9 and vary thermal resistance R from 0.3 to 1.5. Note that, in our simulation, the power consumption of the jobs changes in accordance with the thermal parameters (see Equation (11)), which is true in practice since even the same job would consume different amount of power on different processors or architectures (thus having different thermal behaviors).

Figure 8 shows the results for the two heuristics JA(W)+TM(W) and JA(T)+TM(T). First, the increase of factor f increases the jobs' power consumption, which strengthens the cross-interference effect of the servers, and hence increases the makespan. However, increasing the thermal resistance R reduces the power, weakens the cross interference, and thus decreases the makespan. The makespan gap of the two heuristics is relatively stable, except for very large values of f, in which case the cross interference becomes so strong that the thermal-aware load alone is not sufficient to define the priorities of the servers. However, since a large f would imply a very high value of thermal resistance or thermal capacitance, this situation is unlikely to happen according to the practical values of thermal parameters [40]. Lastly, the energy consumed by both heuristics is always very similar, suggesting that the speed scaling patterns are largely determined by the power profiles of the jobs, which are decided by the thermal parameters but are independent of the scheduling heuristics. *The result shows that the performance of the thermal-aware heuristic is consistent on practical datacenter systems with different thermal parameters.*

7. Conclusion and Future Work

In this paper, we studied a thermal-aware scheduling problem for homogeneous high-performance computing datacenters. We presented a spatio-temporal thermal model that captures both dimensions of the temperature evolution in datacenters. We proposed a scheduling algorithm with both job assignment and thermal management policies based on the notion of thermal-aware load and dynamic voltage and frequency scaling. By means of simulations, we have shown that the proposed algorithm outperforms the existing heuristics in the literature on an experimentally validated datacenter configuration over a wide range of parameter settings. The results also confirm the benefits of using DVFS in the context of thermal-aware scheduling.

The thermal model and scheduling algorithm presented in this paper can be readily extended to heterogeneous datacenter environments, where the execution time and power consumption of the jobs are server dependent. Such heterogeneity may arise from the use of heterogeneous servers or from processor variations. For future work, we plan to extend the scheduling problem to include cooling cost and static power in the optimization, which have been shown to contribute

significantly to the total energy consumption of today's datacenters. While computation-intensive applications are the focus of this paper, many HPC applications running in modern datacenters are accessing a massive amount of data. Hence, another important direction is to consider I/O-intensive or communication-intensive applications by incorporating the cost associated with data movement in the scheduling framework.

Acknowledgment

We thank the anonymous reviewers for their valuable suggestions, which have greatly improved the quality of this paper. This research was supported in part by the European Commission under contract 288701 through the project "CoolEmAll", and by the LABEX MILYON (ANR-10-LABX-0070) of Université de Lyon, within the program "Investissements d'Avenir" (ANR-11-IDEX-0007) operated by the French National Research Agency (ANR).

References

[1] Z. Abbasi and S. K. S. Gupta. Holistic management of sustainable geo-distributed data centers. In *Proceedings of the IEEE International Conference on High Performance Computing (HiPC)*, Bangalore, India, 2015.

[2] W. A. Abdelmaksoud, T. Q. Dang, H.E. Khalifa, R.R. Schmidt, and M. Iyengar. Perforated tile models for improving data center CFD simulation. In *Proceedings of the IEEE Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, San Diego, USA, 2012.

[3] S. Albers and B. Schröder. An experimental study of online scheduling algorithms. *Journal of Experimental Algorithmics*, 7:3, 2002.

[4] O. Arndt, B. Freisleben, T. Kielmann, and F. Thilo. A comparative study of online scheduling algorithms for networks of workstations. *Cluster Computing*, 3(2):95–112, 2000.

[5] ASHRAE guidelines. `http://tc99.ashraetcs.org/`.

[6] N. Bansal, T. Kimbrel, and K. Pruhs. Speed scaling to manage energy and temperature. *Journal of the ACM*, 54(1):3:1–3:39, 2007.

[7] D. M. Brooks, P. Bose, S. E. Schuster, H. Jacobson, P. N. Kudva, A. Buyuktosunoglu, J.-D. Wellman, V. Zyuban, M. Gupta, and P. W. Cook. Power-aware microarchitecture: Design and modeling challenges for next-generation microprocessors. *IEEE Micro*, 20(6):26–44, 2000.

[8] M. T. Chaudhry, T. C. Ling, A. Manzoor, S. A. Hussain, and J. Kim. Thermal-aware scheduling in green data centers. *ACM Computing Surveys*, 47(3):39:1–39:48, 2015.

[9] A. Chavan, M. I. Alghamdi, X.-F. Jiang, X. Qin, J.-F. Zhang, M.-H. Jiang, and M. Qiu. TIGER: Thermal-aware file allocation in storage clusters. *IEEE Transactions on Parallel and Distributed Systems*, 27(2):558–573, 2016.

[10] J. Choi, Y. Kim, A. Sivasubramaniam, J. Srebric, Q. Wang, and J. Lee. A CFD-based tool for studying temperature in rack-mounted servers. *IEEE Transactions on Computers*, 57(8):1129–1142, 2008.

[11] M. Chrobak, C. Dürr, M. Hurand, and J. Robert. Algorithms for temperature-aware task scheduling in microprocessor systems. In *Proceedings of the International Conference on Algorithmic Aspects in Information and Management (AAIM)*, Shanghai, China, 2008.

[12] L. Cupertino, G. Da Costa, A. Oleksiak, W. Piatek, J-M. Pierson, J. Salom, L. Siso, P. Stolf, H. Sun, and T. Zilio. Energy-efficient, thermal-aware modeling and simulation of data centers: the CoolEmAll approach and evaluation results. *Ad Hoc Networks*, 25:535-553, 2015.

[13] A. B. Downey. A parallel workload model and its implications for processor allocation. *Cluster Computing*, 1:133–145, 1998.

[14] K. Ebrahimi, G. F. Jones, and A. S. Fleischer. A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities. *Renewable and Sustainable Energy Reviews*, 31(C):622-638, 2014.

[15] D. G. Feitelson. Workload modeling for computer systems performance evaluation (1st ed.). *Cambridge University Press*, New York, NY, USA, 2015.

[16] D. G. Feitelson and L. Rudolph. Towards convergence in job schedulers for parallel supercomputers. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, 1996.

[17] W.-C. Feng. Making a case for efficient supercomputing. *Queue*, 1(7):54–64, 2003.

[18] M. R. Garey and D. S. Johnson. Computers and intractability, a guide to the theory of NP-completeness. *W.H. Freeman and Company*, 1979.

[19] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal on Applied Mathematics*, 17(2), 416–429, 1969.

[20] M. Harchol-balter. The effect of heavy-tailed job size distributions on computer system design. In *Proceedings of ASA-IMS Conf. on Applications of Heavy Tailed Distributions in Economics*, 1999.

[21] T. Heath, A.P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini. Mercury and freon: Temperature emulation and management for server systems. *SIGOPS Operating Systems Review*, 40(5):106–116, 2006.

[22] R. K. Jain. The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling. *Wiley-Interscience*, New York, NY, 1991.

[23] N.S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J.S. Hu, M.J. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *Computer*, 36(12):68–72, 2003.

[24] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks. System level analysis of fast, per-core DVFS using on-chip switching regulators. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Salt Lake City, USA, 2008.

[25] J. Meng, E. Llamosí, F. Kaplan, C. Zhang, J. Sheng, M. Herbordt, G. Schirner, and A. K. Coskun. Communication and cooling aware job allocation in data centers for communication-intensive workloads. *Journal of Parallel and Distributed Computing*, 96:181–193, 2016.

[26] J. Moore, J. Chase and P. Ranganathan. Weatherman: Automated, online, and predictive thermal mapping and management for data centers. In *Proceedings of the IEEE International Conference on Autonomic Computing (ICAC)*, Dublin, Ireland, 2006.

[27] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making scheduling "cool": temperature-aware workload placement in data centers. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference (ATEC)*, Anaheim, USA, 2005.

[28] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S. K. S. Gupta, and S. Rungta. Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers. *Computer Networks*, 53(17):2888–2904, 2009.

[29] K. Mukherjee, S. Khuller and A. Deshpande. Algorithms for the thermal scheduling problem. In *Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Boston, USA, 2013.

[30] A. Pahlavan, M. Momtazpour, and M. Goudarzi. Power reduction in HPC data centers: a joint server placement and chassis consolidation approach. *Journal of Supercomputing*, 70(2):845–879, 2014.

[31] E. Pakbaznia and M. Pedram. Minimizing data center cooling and server power costs. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, San Fancisco, USA, 2009.

[32] W. Piątek, A. Oleksiak, and G. Da Costa. Energy and thermal models for simulation of workload and resource management in computing systems. *Simulation Modelling Practice and Theory*, 58:40–54, 2015.

[33] M. Polverini, A. Cianfrani, S. Ren, and A. V. Vasilakos. Thermal-aware scheduling of batch jobs in geographically distributed data centers. *IEEE Transactions on Cloud Computing*, 2(1):71–84, 2014.

[34] D. Rajan and P. S. Yu. Temperature-aware scheduling: When is system-throttling good enough? In *Proceedings of the International Conference on Web-Age Information Management (WAIM)*, Zhangjiajie, China, 2008.

[35] L. Ramos and R. Bianchini. C-Oracle: Predictive thermal management for data centers. In *Proceedings of the IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Salt Lake City, USA, 2008.

[36] A. Sansottera and P. Cremonesi. Cooling-aware workload placement with performance constraints. *Performance Evaluation*, 68(11):1232-1246, 2011.

[37] O. Sarood, P. Miller, E. Totoni, and L. V. Kale. "Cool" load balancing for high performance computing data centers. *IEEE Transactions on Computers*, 61(12):1752–1764, 2012.

[38] O. Sarood, E. Meneses, and L. V. Kale. A 'cool' way of improving the reliability of HPC machines. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, Denver, USA, 2013.

[39] K. Skadron, T. Abdelzaher, and M. R. Stan. Control-theoretic techniques and thermal-RC modeling for accurate and localized dynamic thermal management. In *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA)*, Boston, USA, 2002.

[40] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Transactions on Architecture and Code Optimization*, 1(1):94–125, 2004.

[41] H. Sun, P. Stolf, J.-M. Pierson, and G. Da Costa. Energy-efficient and thermal-aware resource management for heterogeneous datacenters. *Sustainable Computing: Informatics and Systems*, 4(4):292–306, 2014.

[42] 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 Transactions on Parallel and Distributed Systems*, 19(11):1458–1472, 2008.

[43] Q. Tang, T. Mukherjee, S. K. S. Gupta, and P. Cayton. Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters. In *Proceedings of the Fourth International Conference on Intelligent Sensing and Information Processing (ICISIP)*, Bangalore, India, 2006.

[44] H. Topcuouglu, S. Hariri, and M-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260-0274, 2002.

[45] L. Wang, S. U. Khan, and J. Dayal. Thermal aware workload placement with task-temperature profiles in a data center. *Journal of Supercomputing*, 61(3):780–803, 2012.

[46] J. Yang, X. Zhou, M. Chrobak, Y. Zhang, and L. Jin. Dynamic thermal management through task scheduling. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and software (ISPASS)*, Austin, USA, 2008.

[47] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of the Annual Symposium on Foundations of Computer Science (FOCS)*, Milwaukee, USA, 1995.

[48] S. Zhang and K. S. Chatha. Approximation algorithm for the temperature-aware scheduling problem. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, San Jose, USA, 2007.

[49] K. Zhang, S. Ogrenci-Memik, G. Memik, K. Yoshii, R. Sankaran, and P. Beckman. Minimizing thermal variation across system components. In *Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, Hyderabad, India, 2015.