Modeling and Analysis of the Thermal Properties Exhibited by Cyberphysical Data Centers

Saif U. R. Malik, *Member, IEEE*, Kashif Bilal, *Student Member, IEEE*, Samee U. Khan, *Senior Member, IEEE*, Bharadwaj Veeravalli, *Senior Member, IEEE*, Keqin Li, *Fellow, IEEE*, and Albert Y. Zomaya, *Fellow, IEEE*

***Abstract*—Data centers (DCs) contribute toward the prevalent application and adoption of the cloud by providing architectural and operational foundation. To perform sustainable computation and storage, a DC is equipped with tens of thousands of servers, if not more. It is worth noting that the operational cost of a DC is being dominated by the cost spent on energy consumption. In this paper, we model a DC as a cyberphysical system (CPS) to capture the thermal properties exhibited by the DC. All software aspects, such as scheduling, load balancing, and all the computations performed by the devices, are considered the "cyber" component. The supported infrastructure, such as servers and switches, are modeled as the "physical" component of the CPS. We perform detailed modeling of the thermal characteristics displayed by the major components of the CPS. Moreover, we propose a thermal-aware control strategy that uses a high-level centralized controller and a low-level centralized controller to manage and control the thermal status of the cyber components at different levels. Our proposed strategy is testified and demonstrated by executing on a real DC workload and comparing it with three existing strategies, i.e., one classical and two thermal-aware strategies. Furthermore, we also perform formal modeling, analysis, and verification of the strategies using high-level Petri nets, the Z language, the Satisfiability Modulo Theories Library (SMT-Lib), and the Z3 solver.**

***Index Terms*—Cloud computing, cyberphysical systems (CPSs), data center (DC), formal methods, modeling, verification.**

I. Introduction

DATA centers (DCs) host a large number of servers to improve the services for high-performance computing applications [1], [2]. Because of the high energy requirements of the computing and cooling devices, the energy consumption of DCs can cost millions of dollars. The DC run-time cost is dominated by the cost spent on the energy consumption of computing and cooling technologies. Based on the energy consumption of a Google DC, a report suggested that Google was possibly running about 900 000 servers in 2010 [4]. The computational and operating margins of DCs highly depend on the provision of the QoS. Higher QoS attribute levels lead to higher rates that in turn lead to higher computations. To deliver the specified level of performance, the number of computational devices put in use at all levels of a DC has significantly increased. As a result, the rate at which the heat is emitted by the devices has also increased. The cost to stabilize the temperature in a DC has drastically increased and has become almost equal to the cost of operating computational systems. The increasing cost of energy consumption calls for new strategies to improve the energy efficiency in DCs. Several strategies have been proposed, such as those in [6]–[8], for efficient energy consumption in DCs. In this paper, we model a DC as a cyberphysical system (CPS) to capture the dynamics and evolution of the thermal properties presented by the DC. The phenomena of increase in the temperature of servers as a result of task allocations and the ambient effect of such increase in the temperature that affect other servers are termed as the thermal dynamics of DCs.

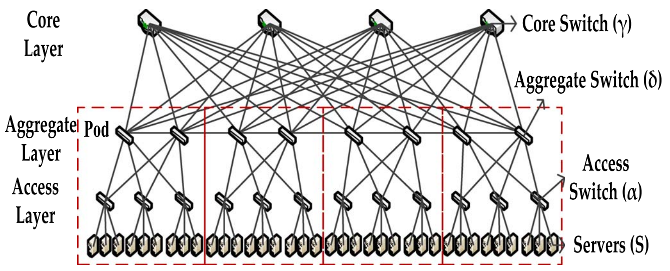


Fig. 1. Three-tier DC architecture.

The software aspects, such as scheduling and computations, performed by the devices are modeled as the "cyber" portion, and the devices, such as servers and switches, are modeled as the "physical" portion of the CPS. Several studies are available that model a DC as a CPS to achieve energy efficiency, such as the works in [9], [10], and [26]. The models proposed in literature are abstract in the sense that they lack detailed analysis of the DC; hence, it becomes difficult to exactly understand the process of heat distribution both from software and infrastructure perspectives. Thus, in this paper, we provide detailed modeling and formulation of the cyber and physical infrastructures, including the heat dissipation of individual components, the heat distribution, and recirculation among the physical portion of the CPS.

The physical infrastructure of the DC follows a hierarchical model (as shown in Fig. 1), where the computing resources reside at the lowest layer. The network infrastructure can be

Manuscript received January 9, 2015; revised June 30, 2015 and August 3, 2015; accepted September 29, 2015. Date of publication December 22, 2015; date of current version March 10, 2017.

S. U. R. Malik and K. Bilal are with the Department of Computer Science, COMSATS Institute of Information Technology, Islamabad 44000, Pakistan (e-mail: saif_ur_rehman@comsats.edu.pk; kashifbilal@ciit.net.pk).

S. U. Khan is with the Department of Electrical and Computer Engineering, College of Engineering, North Dakota State University, Fargo, ND 58105-5285 USA (e-mail: samee.khan@ndsu.edu).

B. Veeravalli is with the Department of Electrical and Computer Engineering, Faculty of Engineering, National University of Singapore, 117576 Singapore (e-mail: elebv@nus.edu.sg).

K. Li is with the Department of Computer Science, School of Science and Engineering, State University of New York at New Paltz, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

A. Y. Zomaya is with the School of Information Technologies, The University of Sydney, Sydney, NSW 2006, Australia (e-mail: albert.zomaya@sydney.edu.au).

Digital Object Identifier 10.1109/JSYST.2015.2493565

considered a multilayer graph, where servers and switches are vertices, and the interconnections among them are the edges. Servers, access switches, and aggregate switches are assembled in modules (referred to as the pod) and are arranged in three layers, i.e., the access, aggregate, and server layers. We perform thorough analysis and modeling of the thermal subtleties involved at each layer. In doing so, we model the heat dissipation of servers and switches (the access, aggregate, and core layers), and the aggregate impact of each component on the overall infrastructure.

Contributions: By exploiting the thermal behavior of discrete elements, we propose a thermal-aware control strategy (TACS) that uses a high-level centralized controller (HLCC) and a low-level centralized controller (LLCC) to manage and control the thermal status of the CPS at different levels, such as the low (server) level, the high (access, aggregate, and core switches) level, the intrapod level, and the interpod level. The complete details of all levels and controllers will be discussed in later sections. We perform the simulation of our proposed strategy on real DC workloads, which were obtained from the Center of Computational Research (CCR), State University New York at Buffalo, Buffalo, NY, USA. The traces have more than 22 000 jobs, and the records are of one month's time. Moreover, we perform a comparative analysis of our proposed strategy with one classical scheduling approach and two thermal-aware approaches, i.e., first come first serve (FCFS), genetic algorithm (GA)-based thermal-aware scheduling [3], and the thermal-aware scheduling algorithm (TASA) [16], respectively.

In this paper, we also made an effort to diminish the level of abstraction through detailed modeling and formal analysis of the CPS. We use high-level Petri nets (HLPNs) and the Z language for the modeling and analysis of the systems. The HLPNs are used to simulate and provide mathematical representation, and analyze the behavior and structural properties of the system. Moreover, we performed the verification of the models using the Satisfiability Modulo Theories (SMT) Library (SMT-Lib) and the Z3 solver. We performed the automated verification of the model by following a bounded model checking technique using SMT-Lib and the Z3 solver. To verify using SMT, the Petri net model is first translated into SMT along with the specified properties. Then, the Z3 solver is used to check whether the model satisfies the properties or not. The contributions of this paper are as follows:

1) formulating the thermal properties of the major component involved in the CPS, the effect of cyber activities on the physical properties of the DC, and vice versa;
2) proposing a TACS that uses an HLCC and an LLCC to manage, control, and coordinate between the cyber and physical portions to maintain a unified thermal threshold range;
3) conducting the simulation and comparison of the proposed strategy on a real DC workload; and
4) modeling and analyzing the CPS in HLPNs and the verification of the model using SMT-Lib and the Z3 solver.

The rest of this paper is organized as follows. Section II will review some of the related work done in the domain of thermal management and CPS modeling of DCs, and the preliminary tools and technologies used in this paper will be presented in Section III. The modeling of thermal properties exhibited by a cyberphysical DC is performed in Section IV, and the proposed control strategies and controllers are described in Section V. The modeling, analysis, and verification of the controllers and strategies are discussed in Section VI, and the comparison results of our strategy with a GA-based approach are demonstrated in Section VII. Finally, Section VIII concludes this paper, followed by the references and the bibliographies of the authors.

II. Related Work

A paradigm shift has occurred in DCs, where the cost of information technology equipment or hardware is no longer the major portion of the overall cost; instead, the cost of power and cooling infrastructure has crept in to be the primary cost driver. The power consumption and thermal properties of the devices are directly proportional to each other. Therefore, in this section, we will discuss both power and thermal strategies. Several strategies have been proposed to balance the tradeoff between the power, cooling, and performance. The power consumption of the servers can be tuned through physical control, such as dynamic voltage and frequency scaling (DVFS) and on–off state control [12]. Moore *et al.* [7] proposed a temperature-aware workload placement approach in a DC. The aforesaid approach is based on the thermodynamics formulation, power, and thermal profiles of the servers. However, precise measurement of the profiles for such a large number and types of jobs is complicated. Moreover, the thermal and power models are not accurate for DCs. In another approach [17], modeling a thermal topology of a DC is discussed that can lead to more efficient workload placement. However, preserving the safe temperature and migration of the resources is not discussed. A DC environmental control system is proposed in [19] that uses a distributed sensor network to manipulate computer room air conditioning (CRAC) units. However, the discussion in [19] is only concentrated on CRAC and did not consider the servers. Kim and Kumar [10] have modeled a DC as a CPS and proposed a control strategy to optimize the tradeoff between the quality of computational and energy costs. However, the heat recirculation and its effect on the other neighboring nodes are not discussed.

Varsamopoulos *et al.* [20] have proposed an analytical transient heat transfer model as a replacement of computational fluid dynamics (CFD) simulations to speed up the evaluation and decision-making process in initially designing and modifying the configurations of the DC. The CFD simulations take considerable amount of time, and such long stretch of simulation time is not suitable for online model-based decision making. To solve the aforesaid problem, Moore *et al.* [7] have proposed a transient heat transfer model that takes a small fraction of the CFD run time and has the ability to introduce logic and triggers, which are hard to implement in CFD. In our paper, we model the heat dissipation of the discrete elements of a cyberphysical DC, such as servers and switches, as a function of the power consumed by the devices when the processing is being performed. Once the thermal values of the devices are computed, we exploit those values to perform task migrations and traffic redirection to avoid hotspots and maintain thermal balance within the DC. Moreover, the thermal values are further used to compute the ambient effect of a server in close proximity by using thermodynamic concepts. Furthermore, the thermal effect of allocating a task to a server and other neighboring servers is also analyzed using the aforesaid thermal values. To model the transmission of heat and its effect on other servers, we used thermodynamic concepts. The thermal analysis is then used to propose a TACS that maintains thermal uniformity within the pods of a DC.

III. Modeling Thermal Characteristics of Cyberphysical DC

We model a DC as a CPS, where the logical classification is made between the computational section and the supporting infrastructure. The computational section, such as scheduling, that participates in the distribution, processing, and flow of tasks constitutes the *cyber* portion. The supporting infrastructure, such as servers and switches, constitutes the *physical* portion. The cyber portion performs computations or any other task to deliver the specified QoS attributes. In return, the physical portion emits thermal energy into the DC environment that raises the temperature. In this paper, we present a methodology that analyzes the thermal characteristics of the cyber and physical portions in a unified way to maintain a specified range of the thermal threshold in the CPS. Generally, there are three main contributors in the power consumption of a DC, i.e., servers (40%–55%), data center networks (DCNs) (10%–25%), and cooling systems, such as CRAC (15%–30%). We perform the thermal modeling and analysis of servers and DCNs only. The CRAC units are reactive systems, where the supplied temperature coming from CRAC depends on the overall temperature of the DC environment. The proposed thermal-aware strategy (see Section V) aims at maintaining unified thermal temperatures within the pods of the DC that will ultimately reduce the overall temperature of the DC. Therefore, by reducing the temperature of the DC, we are indirectly controlling the CRAC supplied temperature, which is derived by the ambient DC temperature. It is worth noting that we are only interested in the modeling of the thermal properties of the DC and not the performance. The DC is logically classified as the combination of the cyber and physical portions, i.e., DC $=$ DC(Cyber) $+$ DC(Physical).

The CPS is comprised of computing resources such as servers and the network infrastructure, such as switches, interconnecting all of the computing resources (see Fig. 1). The CPS follows a hierarchical model, where the computing resources reside at the lowest layer, as depicted in Fig. 1. The network infrastructure can be considered a multilayer graph [21]. The servers, access switches, and aggregate switches are assembled in modules (referred to as the pod) and are arranged in three layers, i.e., the access, aggregate, and server layers. The core layer is used to connect all of the independent pods together. Note that the cyber portion resides within the physical portion. Therefore, we model the DC in a unified way that can accommodate both the cyber and physical sections.

We divided the CPS model into two logical sections, i.e., pods (zones) and core layer switches, as follows:

$$\text{DC} = \text{Pod}_{\forall i \in k}(i) \cup C_{\forall q \in r}(q) \tag{1}$$

where $C(q)$ is the set of core layer switches, and r is the total number of core switches (γ) in the network. Pod(i) is the set of pods, and k is the total number of pods in the network. Each access layer switch (α) is connected to n number of servers (S) in a pod. Moreover, every α is connected to every aggregate switch (δ) in the pod. The number of nodes (including S, α, and δ) in Pod(i) can be calculated as

$$\text{Pod}(i) = S^i_{(n\times m)} \cup \alpha^i_m \cup \delta^i_w \tag{2}$$

where $S^i_{(n\times m)}$ represents a set of servers connected to α in Pod(i). α^i_m represents the access layer switches in Pod(i), where m is the total number of α in Pod(i). δ^i_w represents the aggregate layer switches, and w is the number of δ in Pod(i). The components in the CPS work individually or cooperatively to accomplish the assigned tasks.

According to the law of energy conservation, energy can be neither created nor destroyed, but it can be converted from one form to another. Mechanical energy is consumed by the physical portion when it performs cyber tasks, and almost all the power drawn by the computing devices is dissipated as heat. We model the heat dissipation of every component within the pod, such as S, α, and δ. The heat dissipated by S is represented as ζ_s and can be calculated as follows:

$$\zeta_s^{i,\alpha} = {\zeta_0}^{i,\alpha} + {\zeta_p}^{i,\alpha} + {\zeta_m}^{i,\alpha} \tag{3}$$

where

$$\zeta_p^{i,\alpha} = {\zeta_{\text{rw}}}^{i,\alpha} + {\zeta_{op}}^{i,\alpha}. \tag{4}$$

$\zeta_0^{i,\alpha}$ represents the heat dissipated as a result of the static power to keep the server awake, and $\zeta_p^{i,\alpha}$ represents the heat dissipation when the processing is being performed. $\zeta_0^{i,\alpha}$ is fixed, does not change, and is independent. However, $\zeta_p^{i,\alpha}$ is dynamic and is dependent on the workload. $\zeta_m^{i,\alpha}$ represents the heat dissipated by the memory that includes the energy consumed during the memory refresh operations. $\zeta_p^{i,\alpha}$ is further decomposed into $\zeta_{\text{rw}}^{i,\alpha}$ that represents the heat dissipation because of the read and write operations, and $\zeta_{op}^{i,\alpha}$ is the heat dissipation as a result of the processing performed. We model switches as normal and high-end switches. The switches used in the core layer are usually high-end switches and dissipate more heat as compared with normal switches. We assume that α and δ are normal switches and that γ are high-end switches. The heat dissipated by the normal switches, such as α and δ, is represented as ξ_n and can be calculated as

$$\xi_n^i = (\xi_0 + \xi_f + \xi_b + \xi_p)^i \tag{5}$$

where

$$\xi_b = \xi_{ig} + \xi_e \tag{6}$$

$$\xi_p = \xi_{p'} + \xi_{\text{pr}} + \xi_{\text{rw}}. \tag{7}$$

ξ_0 represents the heat dissipation of the switch as a result of static power consumption, and ξ_f represents the heat dissipation of the communication fabric used in the switch; ξ_b represents the heat dissipation of the buffer that includes ξ_{ig} and ξ_e, which represent the heat dissipation of ingress and egress processing units, respectively. ξ_p represents the heat dissipation during the processing that includes $\xi_{p'}$ and ξ_{rw}, which represent the static heat dissipation of the switch processor and when read and write operations are performed, respectively. ξ_{pr} represents the heat dissipation due to the processing performed by the switch. $\xi_{p'}$ and ξ_0 are constant. However, ξ_p and ξ_b are dynamic and depend on the workload of the switch. γ has different characteristics from α and δ. α facilitates the connection of the network with the end-node devices, and for this reason, it supports features such as port security and virtual local area networks (VLANs). δ manages or segments the traffic from the leaf nodes into VLANs and provides the information to the core layer. For the aforementioned reason, δ provides inter-VLANs routing functions to communicate. γ are the high-speed backbone of the network; thus, they have a very high forwarding rate. Moreover, they have the capability to support link aggregation to ensure adequate bandwidth and traffic routing coming from δ. Furthermore, γ have additional hardware redundancy features, such as redundant power supplies, to swap while the switch continues to operate. Because of the high workload carried out by γ, they dissipate more heat than α and δ. We represent the heat dissipation of high-end switches (core layer) as $\mathfrak{K}_\gamma$, which can be calculated using (5)–(7). However, because of the workload and hardware redundancy, the value of $\mathfrak{K}_\gamma$ must be always greater than ξ_n. In the previous discussion, we have modeled the heat dissipation of the individual nodes, as in (3) and (5), involved in the CPS. The heat dissipated by all the servers in Pod(i), which is represented as $\S_s^i$, can be calculated as

$$\S_s^i = \sum_{p=1}^{m} \sum_{x=1}^{n} \zeta_x^{i,p} \tag{8}$$

where $\zeta_x^{i,p}$ represents the heat dissipation of S_x connected to m in Pod(i). Moreover, the heat dissipation of all α and δ in Pod(i), which is represented as $\S_\partial^i$ and $\S_g^i$, respectively, can be calculated as

$$\S_\partial^i = \sum_{x=1}^{m} \xi_x^i \tag{9}$$

$$\S_g^i = \sum_{h=1}^{w} \xi_h^i \tag{10}$$

where ξ_x^i and ξ_h^i represent the heat dissipated by the access and aggregate switches in Pod(i), respectively. Similarly, the overall heat dissipated by the CPS, which is represented as ψ_c, can be calculated as

$$\psi_c = \sum_{i=1}^{k} (\S_s^i + \S_\partial^i + \S_g^i) + \sum_{j=1}^{r} (\mathfrak{K}_\gamma^j). \tag{11}$$

It is worth noting that the heat calculations performed at this point do not consider the ambient effect involved in the CPS environment. The next paragraphs will discuss the process

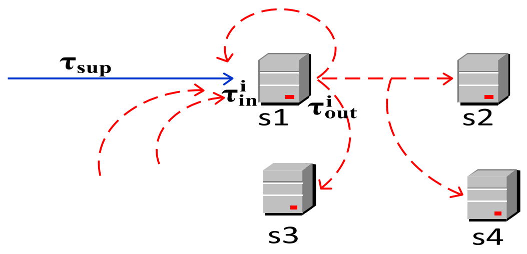


Fig. 2. Heat exchange among server nodes.

of ambient temperature and its effect on the heat dissipation of an individual component. The ambient temperature is the surrounding temperature. Fig. 2 illustrates the effect of the ambient temperature in the CPS environment. The red and blue lines in Fig. 2 depict the movement of hot and cold air, respectively. The hot air is exchanged among the racks, whereas the cooling is provided from the cooling devices, such as CRAC. Suppose there are $\aleph$ number of nodes that participate in the heat dissipation of the CPS. Two temperatures are associated with each node, i.e., the input temperature (τ_{in}^i) and the output temperature (τ_{out}^i). τ_{in}^i represents the input ambient temperature of the node that includes the heat received from other thermal nodes. As depicted in Fig. 2, the τ_{in}^i of s_1 involves the recirculation (red dotted lines) of hot air from other nodes and the cooling temperature (τ_{sup}) from CRAC (more details on CRAC modeling can be seen in [18]). The heat dissipated by any node $i \in \aleph$ will change τ_{out}^i. τ_{in}^i and τ_{out}^i represent the temperature of the surroundings and not that of the node. However, the heat dissipated by the node (π_{out}^i) can affect the values of τ_{in}^i and τ_{out}^i. The input temperature of a node (π_{in}^i) can be calculated as

$$\pi_{\text{in}}^i = \varrho^i \left(\tau_{\text{in}}^i\right) \tag{12}$$

where

$$\tau_{\text{in}}^i = \sum_{j=1}^{\aleph} \left(\pi_{\text{out}}^j\right) + \tau_{\text{sup}}. \tag{13}$$

ϱ is an air coefficient that represents the product of air density (which changes from 1.205 kg/m^3 at 20 °C to 1.067 kg/m^3 at 60 °C), the heat of air, and the flow rate of air. π_{out}^i can be calculated as

$$\pi_{\text{out}}^i = \pi_{\text{in}}^i + \beth^i \tag{14}$$

where

$$\beth_i = \varrho^i \left(\tau_{\text{out}}^i - \tau_{\text{in}}^i - \omega^*\right). \tag{15}$$

$\beth_i$ represents the heat dissipation of a node $i \in \aleph$ in proportion to the power consumed during the processing. ω^* can be replaced by any of the heat dissipation values of three nodes. For instance, if the calculating node is γ, then ω^* can be replaced with $\mathfrak{K}$. Suppose we have the current power distribution of all the servers in Pod(i), which is represented as a vector $\vec{P_i}$. The temperature profile of all the servers, which is represented as a vector $\vec{T_i}$, can be calculated based on the given power distribution. The current temperature of S_i in Pod(j) is denoted as $t_{our}^{i,j}$, which can be calculated as $t_{our}^{i,j} = \pi_{\text{in}}^i + \Delta t(c_i)$, where $\Delta t(c_i)$ represents the anticipated change in the temperature caused by

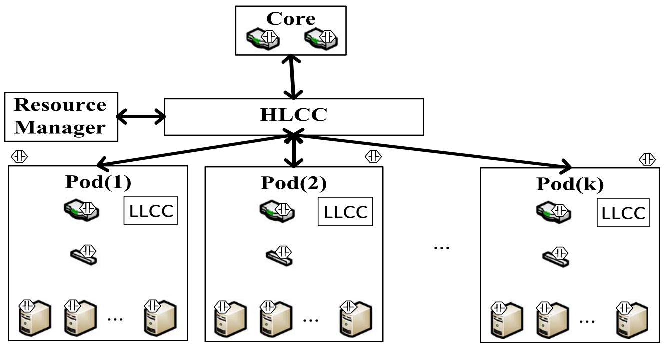


Fig. 3. HLCC and LLCC in the DC.

executing a task c_i on S. According to the abstract heat model of the DC, as discussed in previous works [25], the heat distribution and its effect on the surrounding machines can be represented as a cross-interference coefficient matrix. We follow the same model and compute the heat distribution of the servers using a matrix, which is represented as $h_{n\times m} = \{\partial_{i,j}\}$, which denotes the thermal effect of S_i on S_j and can be populated as

$$\partial_{i,j} = \tau^i_{\text{out}} \times kt \times \frac{1}{\hat{h}_j}$$

where kt is the thermal conductivity constant of the air, and $\hat{h}$ is the hop count of S_j from S_i.

IV. TACS

We propose a thermal-aware scheduling approach that uses an HLCC and an LLCC to manage and control the thermal properties of the CPS at different levels, such as the low (server) level, the high (access and aggregate switches) level, the intrapod level, and the interpod level. The goal is to eliminate hotspots and to maintain a uniform range of the thermal threshold in every pod. Whenever a new job (a job can have multiple tasks) arrives to the CPS, the tasks are allocated to the specified server based on the thermal signatures. The HLCC and the LLCC are proposed that perform the task allocation, task migration, and traffic redirection, which are based on the thermal analysis of the node or the layer. As depicted in Fig. 3, there is an LLCC in every pod that has the thermal information of all S, α, and δ. Every node in the CPS is equipped with a heat sensor that measures the temperature, and the temperature is periodically updated to the LLCC.

In the *low (server) level* (see Fig. 4), $\zeta^{i,\alpha}_s$ for all $S \in \text{Pod}(i)$ is measured and observed through sensors periodically. Whenever the value of $\zeta^j_s\ \forall j \in n$ exceeds the maximum threshold temperature of the server ($\tau^\zeta_{\max}$), the LLCC migrates some tasks from S^j to S^l, where S^j and S^l are connected to the same α. For the tasks to be successfully migrated to S^l, constraint $\zeta^l_s + \Delta T < \tau^\zeta_{\max}$ must be satisfied. ΔT represents the anticipated increase in the temperature as a result of task migration. If the task migration is not possible among the servers under α_i, then the servers belonging to $\alpha_j\ \forall j \in m \wedge j \neq i$ are considered for the migration. α_i and α_j belong to the same pod. Moreover, if there is no server available for the migration within the same pod, then interpod task migration is performed by enforcing the same constraint.

```
1:  for $i \leftarrow 1$ to $k$ do
2:      $\tau^i_\rho = §^i_s + §^i_\partial + §^i_g$      // also use in inter-pod migration
3:  end for
4:  Select $min\ (\tau^i_\rho)$
5:  Get $\varsigma^{i,\alpha}_s\ \forall\ s \in n\ \wedge \alpha \in m$
6:  Select $\varsigma^{i,\alpha}_s$, such that $\varsigma^{i,\alpha}_s < \varsigma^{i,\alpha}_y\ \forall\ y \in n\ \wedge \alpha \in m \wedge y \neq s$.
7:  Allocate $c$ to $\varsigma^{i,\alpha}_s$, $iff\ \varsigma^{i,\alpha}_s + \Delta t(c) < \tau^\varsigma_{max}$ //
8:  If $\varsigma^{i,\alpha}_s > \tau^\varsigma_{max}\ \forall\ s \in n\ \wedge \alpha \in m$, then
9:      Migrate-task c from $S_i$ to $S_j$, $iff\ \varsigma^{i,\alpha}_j + \Delta t(c) < \tau^\varsigma_{max}$
                                                  // intra-pod migration
10: end if
```

Fig. 4. Steps involved in the low (server) level.

```
1:  for $i \leftarrow 1$ to $k$ do
2:      Compute $\xi^i_n\ \forall\ i \in m\ \wedge i \in w$
3:      If $\exists\ \xi^i_n \in w$ such that $\xi^i_n > \tau^\xi_{max}$, then
4:          Redirect $nt$ from $\delta_i$ to $\delta_j$, $iff\ \xi^j_n + \Delta t(nt) < \tau^\xi_{max}$
5:      end if
6:      If $\exists\ \xi^i_n \in m$ such that $\xi^i_n > \tau^\xi_{max}$, then
7:          Migrate-task c from $S_i$ to $S_j$, $iff\ \varsigma^{i,\alpha}_j + \Delta t(c) <$
            $\tau^\varsigma_{max} \wedge \xi^j_n + \Delta t(nt) < \tau^\xi_{max}$   // intra-pod migration
8:      end if
9:  end for
```

Fig. 5. Steps involved in the high (access and aggregate) level.

In the *high (access and aggregate) level* (see Fig. 5), the focus is to avoid the hotspot at the access and aggregate layers of the CPS by redirecting the traffic from heavily loaded switches to the lightly loaded switches. Redundant paths are available in the network infrastructure of the DC that allows the redirection of traffic from one switch to another (see Fig. 1). The decisions for the redirections are made by the LLCC considering the value of ξ_n for every switch. When value ξ^i_n for α increases from $\tau^\xi_{\max}$, then task migration is performed by the LLCC in the same way as was performed in the low level. The reason for the aforesaid is the fact that there is only one path between the access and the servers. However, in the case of δ, redundant paths are available. Therefore, whenever the value of $\xi^i_n\ \forall i \in w$ exceeds the maximum threshold temperature of the switch ($\tau^\xi_{\max}$), the LLCC instructs the lower level (server) to redirect the traffic from δ_i to δ_j, which both belong to the same pod. The redirection is only allowed if $\xi^j_n + \Delta T < \tau^\xi_{\max}$. If the redirection is not possible within the same pod, then interpod task migration is performed to take some load off from the switch.

The high level and the low level are combined together to form an *intrapod control*. The goal in the intrapod is to stabilize the temperature of the pod by maintaining the thermal signatures of the server, access, and aggregate layers. Local decisions (within the same pod), such as task migration and redirection, are taken by the LLCC to stabilize the temperature. However, interpod migrations are performed with the consent of the HLCC. Whenever interpod actions have to be performed, the LLCC requests the HLCC to provide information about other pods where the tasks can be migrated. Afterward, the LLCCs of the pods can communicate with each other to accomplish the task.

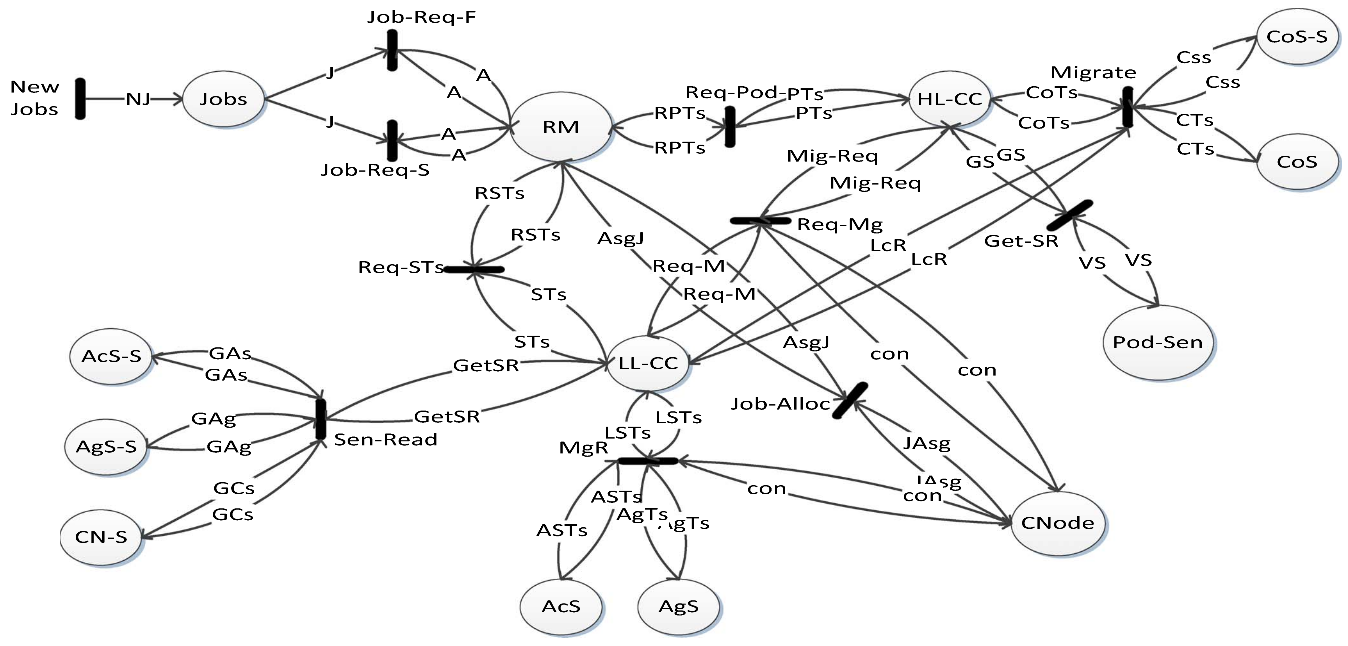


Fig. 6. HLCC and LLCC HLPN model in the DC environment.

The interpod control is focused on maintaining the unified thermal threshold value in all the pods. The thermal signatures of nodes in the CPS can evolve on the order of minutes. Moreover, the power states of servers can change as frequent as milliseconds. Therefore, the threshold temperatures are not absolute values; rather, it is a range within which the thermal signatures of the nodes and layers should lie. In the interpod control, the HLCC periodically monitors the average thermal values of each pod that it receives from sensors. Whenever the thermal signature of Pod(i) $(\tau_\rho^i = \S_s^i + \S_\partial^i + \S_g^i)$ exceeds the maximum thermal threshold value of the pod $(\tau_{\max}^\rho)$, the HLCC instructs the LLCC of Pod(i) to migrate some tasks to Pod(j) $\forall j \in k \wedge j \neq i$. The migration can be only successfully performed if $\tau_\rho^i + \Delta T < \tau_{\max}^\rho$. Moreover, the server selection and task allocation performed in the interpod control are the same as those in the low level. The HLCC only has the coarse-grained information of τ_ρ^i. The allocations of migrated tasks to servers are performed by the LLCC through the use of fine-grained servers' information.

All of the aforementioned controls work together to make sure that the CPS is operating under a specified temperature range. More detailed information, formal analysis, and behavior of the HLCC and the LLCC will be discussed in the next section using HLPNs and the Z language.

V. Verification Using HLPN, SMT-Lib, and Z3 Solver

Verification is the process of demonstrating the correctness of an underlying system [15]. Two parameters are required to verify a model of a system, i.e., specification and properties. In this paper, we use a bounded model checking [5] technique to perform the verification using SMT-Lib and the Z3 solver (For the use of SMT-Lib in the verification of the Open Shortest Path First routing protocol, see [11]). In bounded model checking, the description of any system is verified, i.e., whether any of the acceptable inputs drives the system into a state where the system always terminates after a finite number of steps.

TABLE I
Places Used in the HLCC and LLCC Model

Places	Mappings
$\varphi(job)$	$\mathbb{P}$ (*Task* × *Res*)
φ (*RM*)	$\mathbb{P}$ (*Task* × *Res-Mat* × *Th_P* × *Th_S*)
φ $(HL-CC)$	$\mathbb{P}$ (*Th_P*)
φ $(Pod-Sen)$	$\mathbb{P}$ (*Th_P*)
φ $(LL-CC)$	$\mathbb{P}$ (*Th_S* × *Th_Ac* × *Th_Ag*)
φ $(AcS-S)$	$\mathbb{P}$ (*Th_Ac*)
φ $(AgS-S)$	$\mathbb{P}$ (*Th_Ag*)
φ $(CN-S)$	$\mathbb{P}$ (*Th_S*)
φ (Ags)	$\mathbb{P}$ (*RI*)
φ (AcS)	$\mathbb{P}$ (*RI*)
φ (CoS)	$\mathbb{P}$ (*RI*)
φ $(CoS-S)$	$\mathbb{P}$ (*Th_Co*)
φ $(CNode)$	$\mathbb{P}$ (*Task* × *Res*)

A path in the Kripke structure can be stated as an infinite sequence of states represented as $\rho = S_1, S_2, S_3, \ldots$, such that $\forall i \geq 0,\ (S_i, S_{i+1}) \in R$. Model M may produce a path set $= S_1, S_2, S_1, S_2, S_3, S_3, S_3, \ldots$. To describe the property of a model, some formal language, such as CTL*, CTL, or LTL is used. (For more details about CTL*, see [13].) For a model to be correct, the states must satisfy the formulas (Definition 2) under a specific bound.

A. *Modeling HLCC and LLCC Using HLPN*

The HLPN model for the HLCC and the LLCC is shown in Fig. 6. The first step toward modeling using HLPNs is to identify the required types, places (P), and mapping (Definition 1). The mapping of P to the types is depicted in Table I. The description and operation of the controllers are discussed in the previous section, and now, we can define formulas (preconditions and postconditions) to map on transitions.

New tokens can only enter the model through the transition New Jobs. As seen in Fig. 6, no arc is incident on the aforementioned transition, which is why no precondition exists, and the rules for the transitions can be written as $R(\text{New Jobs}) = \exists j \in J | \cdot j = \emptyset$. Whenever new jobs arrive, the resource manager checks if the resources required by the job are available or not. The aforementioned authentication is performed by the transitions Job–Req–F and Job–Req–S, which are mapped to the following formulas:

$$\begin{aligned}&\boldsymbol{R}(Job{-}Req{-}F) = \forall a \in A | \exists J[2] \neq a[2] \wedge \\ &\quad \forall a[3] \in A | a[3] + \Delta t \geq Max\_Th\_P \wedge \\ &\qquad A' := A \qquad (16)\\ &\boldsymbol{R}(Job{-}Req{-}S) = \forall a \in A | \exists J[2] \in a[2] \wedge \\ &\quad \forall a[3] \in A | \exists a[3] + \Delta t < Max\_Th\_P \wedge \\ &\qquad A' := A \bigcup \{(J[1], J[2], a[3], a[4])\}. \qquad (17)\end{aligned}$$

If the resources required by the job are available in the resource matrix of the resource manager and if the thermal signature of the pod for the selected server is less than the maximum thermal threshold, then the jobs are accepted and are placed in the queue, as shown in (17). However, if the resources required by the job are not found, then the job will not be accepted. Moreover, if the cyber portion is running in full capacity, then the job will be also rejected, as in (16). The resource manager instructs the HLCC and the LLCC to provide the list of all the pods and servers that are suitable for the resource allocation. In response, the HLCC provides the thermal information of the pods to the resource manager as follows:

$$\begin{aligned}\boldsymbol{R}(Req - Pod - Ts) =& \forall rpt[3] \in RPTs \ \forall pt \in PTs | rpt[3] \\ &:= pt \wedge RPTs' := rpt[3] \bigcup \{pt\} \qquad (18)\end{aligned}$$

and the LLCC will send the list of all the servers that satisfy the constraint, i.e., $\zeta_s^l + \Delta T < \tau_{\max}^{\zeta} \ \forall l \in n \times m \times k$, as follows:

$$\begin{aligned}&\boldsymbol{R}(Req - STs)\\ &\quad = \forall rt[4] \in RSTs, \ \forall st[1] \in STs, \ \forall rt[3] \in RSTs | rt[4]\\ &\quad := \{\forall st[1] \cdot st[1] + \Delta t < Max\_Th\_S \wedge st[1] \in rt[3]\}\\ &RSTs' := RSTs \bigcup \{(rt[1], rt[2], [3], rt[4])\}. \qquad (19)\end{aligned}$$

The HLCC acquires τ_ρ^i through the heat sensors that are placed at each pod (see Fig. 4). Moreover, the LLCC acquires ζ_s and ξ_n from the heat sensors placed at every node within the pod. The HLCC and the LLCC periodically read the values from the sensors, as shown in the following, respectively:

$$\begin{aligned}&\boldsymbol{R}(Get{-}SR) = \forall g \in GS, \ \forall v \in VS | g := v\\ &GS' := GS \bigcup \{(g)\} \qquad (20)\\ &\boldsymbol{R}(Sen{-}Read) = \forall ac \in GAs, \ \forall ag \in GAg,\\ &\qquad \forall gc \in GCs, \ \forall gsr \in GetSR |\\ &gsr[1] := gc \wedge gsr[2] := ac \wedge gsr[3] := ag \wedge\\ &GetSR' := GetSR \bigcup \{(gsr[1], gsr[2], gsr[3])\}. \qquad (21)\end{aligned}$$

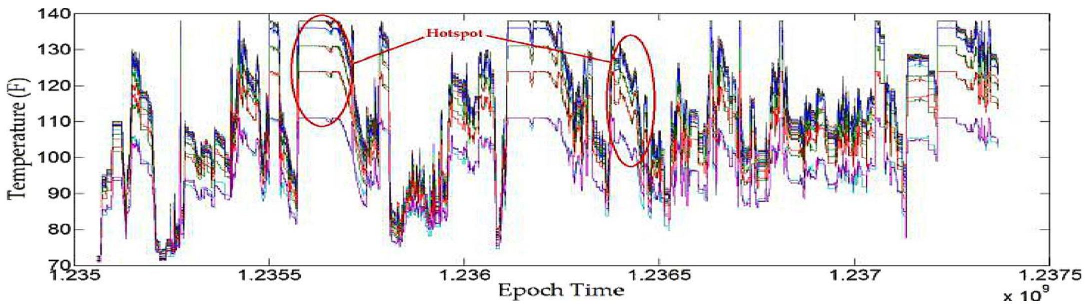
When the resource manager requests for the thermal information of the pods and the servers, the HLCC and the LLCC send the updated values read from the sensors. Transitions Get–SR and Sen–Read perform the aforementioned readings for the HLCC and the LLCC, respectively. The rules for the transitions are (20), (21), and

$$\begin{aligned}&\boldsymbol{R}(MgR)\\ &\quad = \forall act \in ASTs, \ \forall agt \in AgTs, \forall cot \in CoTs,\\ &\qquad \forall cn \in CNode, \ \forall lst \in LSTs | lst\left[1_{(i)}\right]\\ &\quad \geq Max\_Th\_S \wedge \exists lst\left[1_{(j)}\right]_{\forall j \in lst[2], j \neq i} + \Delta t\\ &\quad < Max\_Th\_S \wedge LcMg\left(cn_{(i)}, cn_{(j)}\right) \wedge lst[2_{(i)}]\\ &\quad \geq Max\_Th\_AC \wedge \exists lst\left[2_{(j)}\right]_{\forall j \in lst[2], j \neq i} + \Delta t\\ &\quad < Max\_Th\_Ac \wedge LcRd\left(lst\left[2_{(i)}\right], lst\left[2_{(j)}\right]\right) \wedge lst\left[3_{(i)}\right]\\ &\quad \geq Max\_Th\_Ag \wedge \exists lst\left[3_{(j)}\right]_{\forall j \in lst[3], j \neq i} + \Delta t\\ &\quad < Max\_Th\_Ag \wedge LcRd\left(lst\left[3_{(i)}\right], lst\left[3_{(j)}\right]\right)\\ &\qquad \wedge con\left(cn_{(i)}\right)' = con\left(cn_{(i)}\right)\left\{\left(cn_{(i)}[1], cn_{(i)}[2]\right)\right\}\\ &\qquad \wedge con\left(cn_{(j)}\right)' con\left(cn_{(j)}\right) \bigcup \left\{\left(cn_{(j)}[1], cn_{(j)}[2]\right)\right\}. \qquad (22)\end{aligned}$$

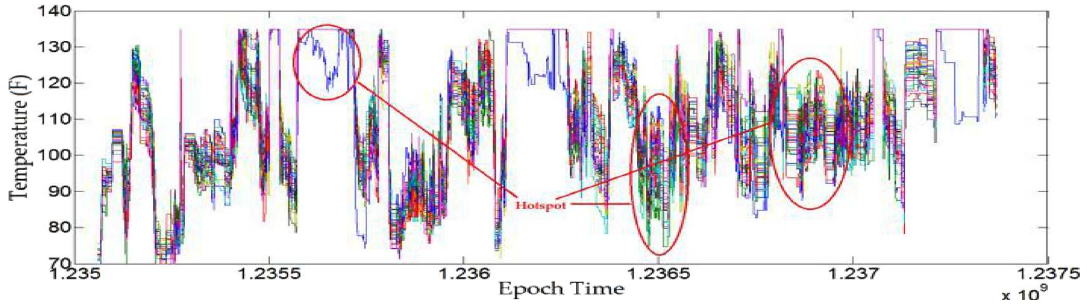
If (17) is fired, then the job is assigned to the selected server, and the resources are allocated to the task, as in (22). As stated in the previous section, to maintain a specified thermal temperature at different levels of the CPS, the HLCC and the LLCC perform task migration and traffic redirections based on the thermal signatures of the nodes. Transition MgR performs the migrations and redirection within the same pod, which are termed as LcMg and LcRd, respectively. The rules for the transition are as follows:

$$\begin{aligned}&\boldsymbol{R}(Req{-}Mg)\\ &\quad = \forall rm \in Req{-}M, \ \forall mr \in Mig{-}Req,\\ &\qquad \forall cn[1] \in con | \left(rm\left[1_{(i)}\right] + \Delta t > Max\_Th\_S\right)_{\forall i \in \text{Pod}(k)}\\ &\qquad \wedge \left(rm\left[2_{(j)}\right] + \Delta t > Max\_Th\_Ac\right)_{\forall j \in \text{Pod}(k)}\\ &\qquad \wedge \left(rm\left[3_{(z)}\right] + \Delta t > Max\_Th\_Ag\right)_{\forall z \in \text{Pod}(k)} \wedge \exists mr(y)\\ &\quad > Max\_Th\_PReq{-}InterPod{-}Mig\\ &\qquad \times \left(cn_{(i) \in \text{Pod}(k)}, cn_{(j) \in \text{Pod}(x) \neq \text{Pod}(k)}\right) \wedge con\left(cn_{(i)}\right)'\\ &\quad = con\left(cn_{(i)}\right)\left\{\left(cn_{(i)}[1], cn_{(i)}[2]\right)\right\} \wedge con\left(cn_{(j)}\right)'\\ &\quad = con\left(cn_{(j)}\right) \bigcup \left\{\left(cn_{(j)}[1], cn_{(j)}[2]\right)\right\} \qquad (23)\\ &\boldsymbol{R}(Migrate)\\ &\quad = \forall ct \in CoTs, \ \forall c \in CTs, \ \forall cs \in Css, \ \forall lr \in LcR | \exists c_{(x)}\\ &\quad > Max\_Th\_Co \wedge Rd\left(c_{(x)}, c_{(x'), x \neq x'}\right) \wedge c_{(x')} + \Delta t\\ &\quad < Max\_Th\_Co. \qquad (24)\end{aligned}$$

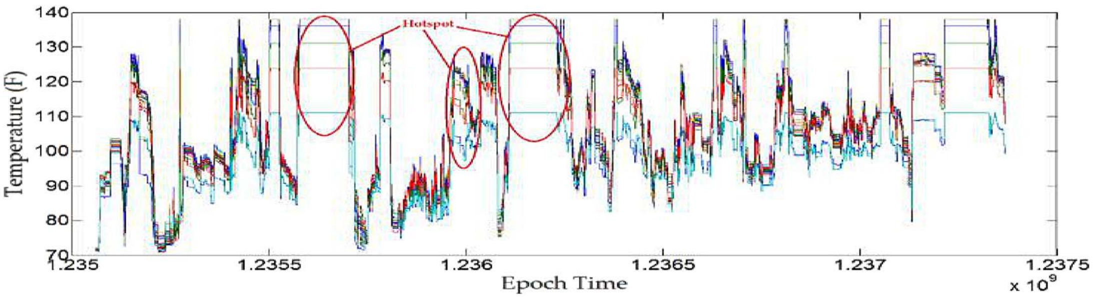
Whenever the thermal signatures of S, α, and δ are raised more than the specified maximum thermal threshold, (22) is fired. Equation (22) makes local redirection and migration by exploiting the functionalities of the LLCC. Interpod migration is achieved by the mutual communication of the HLCC and the LLCC. When migration or redirection is not possible locally, then the LLCC requests the HLCC to provide the information about the pods where the tasks can be migrated, as depicted in (23). Moreover, interpod migration is also performed when the thermal signature of γ exceeds the specified maximum thermal threshold, as illustrated in (24).



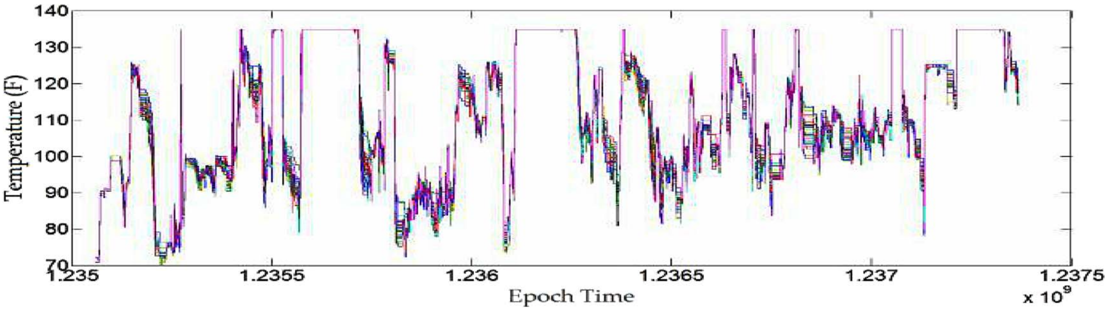
(a)



(b)



(c)



(d)

Fig. 7. Comparison of the average thermal signatures of the pods using the (a) FCFS, (b) GA-based, (c) TASA, and (d) TACS approaches.

VI. Results and Discussion

To demonstrate the effectiveness of our work in a real DC environment, we simulate the proposed strategies on a real DC workload obtained from the CCR, State University of New York at Buffalo. All jobs submitted to the CCR are logged for a period of a month. The jobs and logs from the CCR data set are used as an input for our simulation of the proposed thermal-aware strategy. The data set had 22 700 jobs (127 000 tasks) recorded in one month's time. The DC had 1056 distinct dual-core servers. A server was based on the Dell 1056 PowerEdge SC1425 processor with 3.0-GHz speed, running the x86-64 Linux operating system. The CCR DC was organized into 33 pods, and each pod had 32 servers. Moreover, we also evaluate the proposed TACS by comparing with the classical FCFS, GA-based thermal-aware scheduling [3], and thermal-aware task allocation [16] approaches. We perform the comparison among the aforementioned strategies based on the CCR data set. Before going deeper into the details of the comparison, we first briefly discuss the existing approaches. The FCFS (sometimes referred to as first-in–first-out) approach is possibly the most straightforward scheduling approach. The jobs are submitted to the scheduler, which dispatches the jobs based on the order of the jobs received. The approach in [3] follows the steps of a GA. The first step is to construct a set of feasible solutions, which is the task allocation to the servers. Then, the selected solution is mutated (randomly interchange the task allocations within the solution) and mated (randomly select pairs

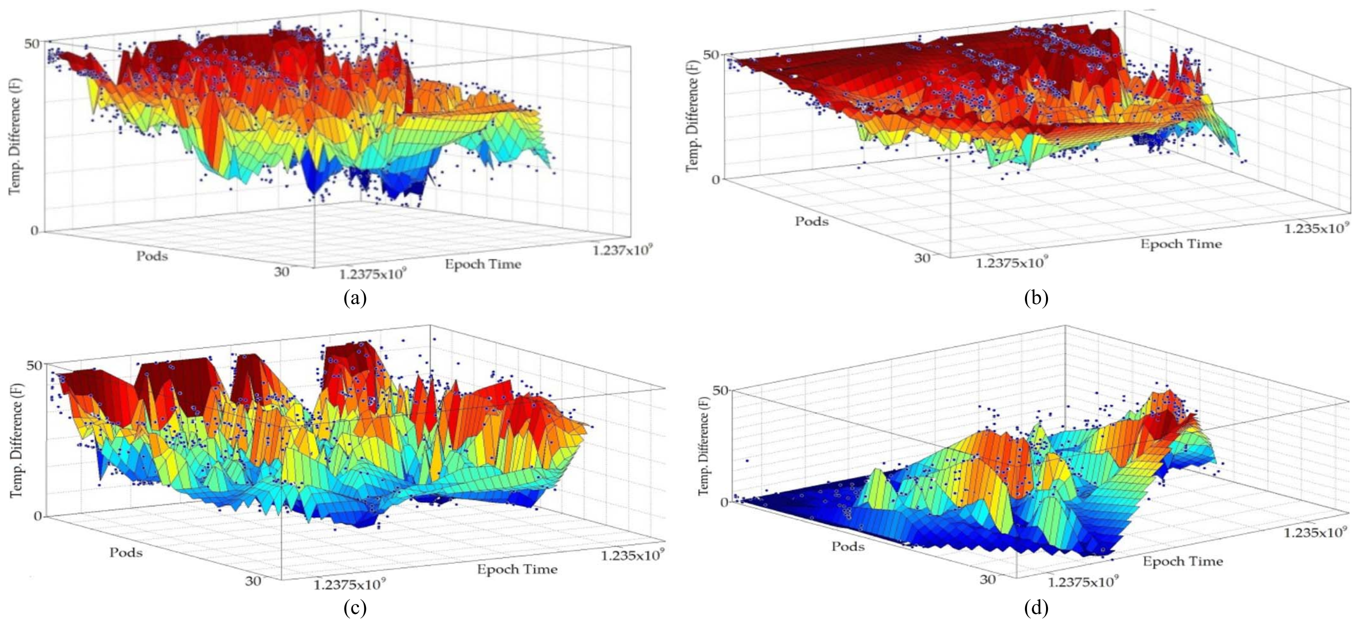


Fig. 8. Comparison of the average thermal signature difference between the highest and lowest servers. (a) FCFS. (b) GA-based approach. (c) TASA. (d) TACS.

of the solution and exchange the subset of two task assignments to get two new solutions). The fitness function, which checks the highest inlet temperature of the selected assignment, is applied to all of the solutions that are formed as a result of mating and mutation, including the original solution. Finally, the solution having the lowest inlet temperature value from the set of highest inlet temperature values, which is obtained as a result of the fitness function, is selected as a final solution. The last approach is the TASA proposed in [16], which is based on the theory of the coolest inlet that performs the assignment of the hottest jobs to the coolest servers. The TASA sorts the servers in the increasing order of the temperatures. The jobs are sorted in a similar way but in the reverse order, such that the hottest job is first in the order. The hottest job is assigned to the coolest server, and the thermal map of all the servers is updated.

Fig. 7 depicts the average thermal signatures of the pods over the period of time when the scheduling approaches are used. The epoch time stamp and average thermal signature of the pods at that particular time are plotted on the x-axis and the y-axis, respectively, in Fig. 7. It can be observed in Fig. 7 that the spread or difference between the temperatures of the servers in the trend line in Fig. 7(a)–(c) is very wide at many time stamps. The aforesaid identify the situation when the average temperature of some servers is lower than the rest of the servers in the DC. In particular, at time stamps 1.2357E+9, 1.2362E+9, and 1.2372E+9 in Fig. 7(a)–(c), the thermal signatures of some pods are very low as compared with the rest, which shows the probable presence of hotspots in the DC.

The possible reason for the occurrence of hotspots in Fig. 7(a) is the static assignment of tasks without considering the thermal status of the server. The aforesaid possibly creates a scenario when higher task temperature profiled jobs are assigned to the servers with high thermal signatures and when low-thermal-impact jobs are assigned to low-thermal-signature servers. In such a scenario, the thermal signatures of the “hot” servers will increase, causing thermal imbalance among the servers and the pods.

In Fig. 7(b), the reason for the imbalance in the thermal signatures is the random nature of the GA-based approach. The selection of the feasible solution, the mutation, and the mating process is based on randomization. If the same set of pods and servers is selected in the solutions most of the time, then the fitness function performed on the selected solution will not provide any important information that will avoid the occurrence of hotspots. Similarly, there is also a possibility that the number of tasks allocated to few pods and servers is relatively low as compared with the rest of the pods and servers in the DC. The aforementioned possibilities will allow some servers to have high thermal signatures while others have low thermal signatures, which will ultimately cause the hotspot in the DC. In Fig. 7(c), the thermal differences are lower than those in Fig. 7(a) and (b). However, there are still some time stamps, where some pods have high thermal signatures and some have low. The reason for the aforesaid is that the high-thermal-profile tasks are allocated to the coolest servers regardless of the overall thermal temperature of the pod and the recirculation effect that can cause hotspots. The aforesaid can cause a situation where the temperature of the server is low but the overall temperature of the pod in which the server lies is high. In such a situation, the overall temperature of the pod increases that can possibly create a hotspot. In the TACS, as shown in Fig. 7(d), the differences of the temperatures among the servers are low, and there are no hotspots. As stated in Sections V and VI, the selection of the pods and the servers to allocate the task is based on the thermal signatures. Moreover, the HLCC and the LLCC periodically monitor the thermal signatures of the pods and the servers, and they perform task migration or redirection to maintain a unified range of temperatures in the pods. Therefore, the trend of thermal signatures followed in Fig. 7(d) is more congested and unified as compared with the trend followed in the rest of the approaches. We plot the average difference between the hottest and coolest servers over the period of time (as shown in Fig. 8). The larger

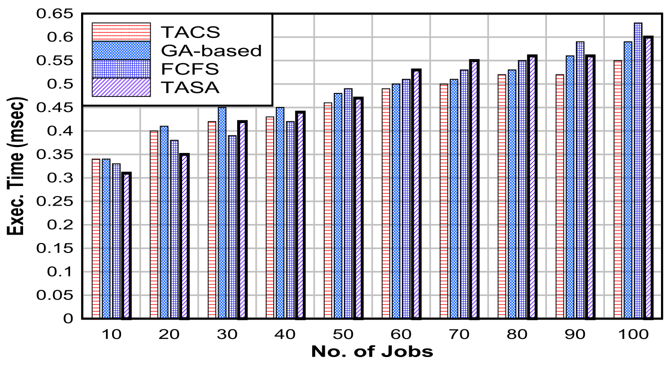


Fig. 9. Verification time comparison of the approaches.

and more frequent the differences are, the higher the thermal imbalance will be. We can see that the differences in the TACS [see Fig. 8(d)] are very low and less frequent as compared with the other approaches that indicate the thermal balance achieved by using the TACS. However, the other approaches have high differences and are occurring frequently, which indicates the thermal imbalance and occurrence of hotspots.

To verify, the HLPN models are first translated into SMT. Then, the models, along with the properties, are provided to the Z3 solver, which checks if the properties are satisfied by the models or not. It is worth noting that the goal of the verification is to demonstrate the correctness of the models, which is based on the desirable properties, such as the presence of hotspots. The results in Fig. 9 depict the time taken by the Z3 solver to check the satisfiability of the models, which is based on the stated property. The property we verify is that there must be no hotspots in the DC. The verification results reveal the absence of hotspots when the TACS is used.

The execution time serves as a bound over the verification models. The simulation and verification results reveal that our strategy is consistent and provides better results as compared with the other scheduling approaches. We reduce the possibility of hotspots in our strategy through strategic decisions performed by the HLCC and the LLCC based on the thermal signatures of the components.

VII. Conclusion

In this paper, we have modeled a DC as a CPS to capture the thermal evolution and properties presented by DC components. Moreover, we proposed a TACS that takes strategic decisions to achieve thermal uniformity within a DC. A comparative analysis was performed, which reveals the effectiveness of our strategy. Furthermore, to demonstrate the correctness of our approach, we performed formal analysis, modeling, and verification using HLPNs, SMT-Lib, and the Z3 solver. The automated verification performed using SMT-Lib and the Z3 solver authenticates the correctness of our approach as compared with other approaches, where hotspots were identified. In the future, we will analyze the effect of workload migration on the network throughput and latency. Moreover, the effect of thermal balancing toward attaining efficient power consumption will be also performed.

References

[1] R. Buyya, S. Y. Chee, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," in *Proc. IEEE HPCC*, 2008, pp. 5–13.

[2] S. U. R. Malik, S. U. Khan, and S. K. Srinivasan, "Modeling and analysis of state-of-the-art VM-based cloud management platforms," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, p. 1, Jan.–Jun. 2013.

[3] Q. Tang *et al.*, "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.

[4] J. Koomey, *Growth in Data Center Electricity use 2005 to 2010*. Berkeley, CA, USA: Analytics Press. Jul. [Online]. Available: http://www.analyticspress.com/datacenters.html

[5] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu, "Bounded model checking," *Adv. Comput.*, vol. 58, pp. 1–27, 2003.

[6] J. Shuja *et al.*, "Energy-efficient data centers," *Computing*, vol. 94, no. 12, pp. 1–22, 2012.

[7] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling 'cool': Temperature-aware workload placement in data centers," in *Proc. USENIX*, 2005, pp. 61–75.

[8] L. Ramos and R. Bianchini, "C-oracle: Predictive thermal management for data centers," in *Proc. IEEE HPCA*, 2008, pp. 111–122.

[9] L. Parolini, B. Sinopoli, B. Krogh, and W. Zhikui, "A cyber—Physical systems approach to data center modeling and control for energy efficiency," *Proc. IEEE*, vol. 100, no. 1, pp. 254–268, Jan. 2012.

[10] K. D. Kim and P. R. Kumar, "Cyber—Physical systems: A perspective at the centennial," *Proc. IEEE*, vol. 100, pp. 1287–1308, May 2012.

[11] S. U. R. Malik, S. K. Srinivasan, S. U. Khan, and L. Wang, "A methodology for OSPF routing protocol verification," in *Proc. Conf. Scalable Comput. Commun.*, Dec. 2012, pp. 1–5.

[12] H. Aydin and D. Zhu, "Reliability-aware energy management for periodic real-time tasks," *IEEE Trans. Comput.*, vol. 58, no. 10, pp. 1382–1397, Oct. 2009.

[13] M. Maidl, "The common fragment of CTL and LTL," in *Proc. Symp. Found. Comput. Sci.*, 2000, pp. 643–652.

[14] SMT-Lib, accessed Jan. 2013. [Online]. Available: http://smtlib.cs.uiowa.edu/

[15] S. Nakajima, "Model-checking verification for reliable web service," in *Proc. OOWS*, 2002, pp. 1–5.

[16] L. Wang *et al.*, "Towards thermal aware workload scheduling in a data center," in *Proc. I-SPAN*, 2009, pp. 116–122.

[17] J. Moore, J. Chase, and P. Ranganathan, "Weatherman: Automated, online and predictive thermal mapping and management for data centers," in *Proc. IEEE ICAC*, 2006, pp. 155–164.

[18] N. Kumari *et al.*, "Cooling capacity and cost metrics for energy efficient workload placement in datacenters," in *Proc. IEEE ITherm*, 2012, pp. 799–805.

[19] C. Bash, C. Patel, and R. Sharma, "Dynamic thermal management of air cooled data centers," in *Proc. IEEE ITherm*, 2006, pp. 445–452.

[20] G. Varsamopoulos, M. Jonas, J. Ferguson, J. Banerjee, and S. Gupta, "Using transient thermal models to predict cyber physical phenomena in data centers," *Sustain. Comput., Informat. Syst.*, vol. 3, no. 3, pp. 132–147, Sep. 2013.

[21] K. Bilal *et al.*, "On the characterization of the structural robustness of data center networks," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 64–77, Jan.–Jun. 2013.

[22] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[23] L. Cordeiro, B. Fischer, and J. Marques-Silva, "SMT-based bounded model checking for embedded ANSI-C software," in *Proc. IEEE ASE*, 2009, pp. 137–148.

[24] M. K. Ganai and A. Gupta, "Accelerating high-level bounded model checking," in *Proc. IEEE ICCAD*, 2006, pp. 794–801.

[25] Q. Tang, T. Mukherjee, S. K. S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," in *Proc. ICISIP*, Dec. 2006, pp. 203–208.

[26] J. Wan *et al.*, "From machine-to-machine communications towards cyber-physical systems," *Comput. Sci. Inf. Syst.*, vol. 10, no. 3, pp. 1105–1128, 2013.

Authors' photographs and biographies not available at the time of publication.