This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2024.3414671

A Caolitional Game-based Adaptive Scheduler Leveraging Task Heterogeneity For Greener Data Centers

Saeed Akbar, Ubaid Ul Akbar, Rahmat Ullah, and Zhonglong Zheng

***Abstract*—Managing power and its subsequent thermal implications is of paramount concern in modern Data Centers (DCs) management. Failure to adequately address the escalating energy use can result in excessive heat dissipation, leading to thermal imbalances and hotspots. In addition, the prolonged execution of CPU-intensive user jobs on servers operating at higher temperatures can significantly aggravate the DCs cooling efforts. Researchers advocate Thermal-aware (TA) scheduling as a promising tool to counter the said issue. However, existing state-of-the-art overlooks user jobs runtime heterogeneity, potentially causing aggravated heat dissipation when CPU-intensive tasks run on servers at elevated temperatures for longer duration. Moreover, existing works do not provide any mechanism to detect overloaded computing nodes at runtime in a TA context. Finally, existing strategies do not adapt according to the DCs dynamic thermal conditions. This paper offers a Coalitional Game-based Thermal-aware Adaptive Scheduling (CGTAS) tailored for heterogeneous DCs to minimize the cooling cost stemming from excessive heat generated during compute-intensive job execution. CGTAS intelligently differentiates incoming jobs based on their thermal profiles and CPU-time for optimal thermal outcomes. In addition, it dynamically allocates user jobs to computing nodes based on their real-time marginal thermal performance using the Core solution concept from game theory. Finally, unlike existing TA strategies, the proposed design identifies thermally overloaded computing elements using the Core and performs task migrations to optimize thermal-efficiency. Extensive simulations confirm substantial energy savings (up to 26.08%) compared to its TA substitutes, promoting sustainable and high-performance computing infrastructure in large-scale cloud DCs.**

***Index Terms*—Data Center, K-means Clustering, Coalitional Game, Core, Thermal-aware, Energy Efficiency, Cooling Cost**

I. Introduction

The power requirements and the subsequent thermal implications of Data Centers (DCs) operations are of paramount concern in modern DCs management [1]. Failure to adequately address the escalating energy use and its subsequent thermal consequences can result in excessive heat dissipation, leading to thermal imbalances and hotspots [2]. While cooling mechanisms are implemented to keep servers and other computing elements under their thermal constraints (TC), they come at a substantial energy cost, accounting from 30% and potentially as much as 50% of the total DC energy cost [3, 4, 5, 6]. Research has identified hotspots as one of the primary contributors to elevated cooling costs [7, 8]. Consequently, efficient thermal management (TM) inside DCs is essential to conserve energy within the cooling infrastructure [2]. Research community offers several strategies such as Thermal-aware (TA) scheduling, liquid cooling, and air-flow management to mitigate the said issues.

Recent studies show that TA scheduling is a promising dynamic TM tool for large-scale DCs [12, 13, 9, 14, 15]. For instance, TA approaches in [12, 8] optimize the thermal outcomes considering the servers temperature and thermal-profile (TP) of user jobs in homogeneous DCs. For heterogeneous DCs, Ref. [9] offers TA strategy considering marginal contribution of computing nodes in disturbing the DC thermal uniformity to minimize the cooling efforts. In addition to TP of user jobs and servers temperature, Ozceylan et al. [14] consider task duration in their study. However, they ignore the relative thermal performance of servers during task scheduling, which may lead to non-optimal results. In summary, existing TA models primarily focus on minimizing cooling expenses associated with elevated server temperatures, emphasizing factors such as TPs of user jobs, server inlet/outlet temperatures, and ambient conditions during task allocation and migration. However, they overlook the impact of CPU-intensive jobs run-time on thermal outcomes. This may lead to frequent rise of thermally overloaded servers in the long run. Also, existing state-of-the-art does not adapt the workload according to dynamic thermal conditions of the DC environment.

The prolonged execution of CPU-intensive jobs on servers operating at higher temperatures can significantly aggravate cooling efforts due to increased heat dissipation. Therefore, the duration of CPU-intensive tasks stands as a critical factor influencing the DCs cooling costs if not managed intelligently. Furthermore, according to [9], assigning user jobs to computing nodes based on their relative thermal performance offers significant improvement in DCs thermal-efficiency. In this context, Coalitional Game Theory (CGT) [10, 11] offers two useful solution concepts that can be used to distribute payoff/cost/benefit based on the marginal contributions of participating agents: (1) the Shapley value, and (2) the Core.

In this work, we use the Core solution from CGT as it offers more stable outcomes compared to the Shapley value [16, 11]. We devise a Coalitional Game-based Thermal-aware Adaptive Scheduler (CGTAS) that assigns incoming jobs based on the relative thermal performance of computing

Saeed Akbar and Zhonglong Zheng are with the School of Computer Science and Technology, Zhejiang Normal University, Jinhua 321004, China (email: saed.akbr@zjnu.edu.cn, zhonglong@zjnu.edu.cn).

Ubaid Ul Akbar is with the Department of Computer Science, City University of Science and Information Technology, Peshawar, Pakistan.

Rahmat Ullah is with the School of Computer Science and Electronic Engineering, University of Essex, United Kingdom.

Corresponding author: Zhonglong Zheng

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2024.3414671

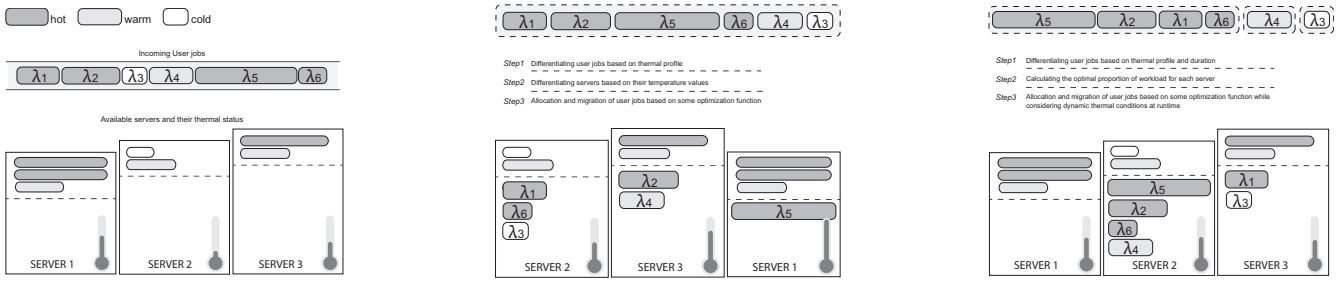

Fig. 1: Illustrating potential improvement in thermal efficiency offered by the proposed system. Computing nodes are represented using rectangles with height indicating their capacity. Rounded rectangle represents user jobs where gray, light gray, and white represent hot, warm and cold jobs. The thermal readings of computing nodes are represented using thermometer indicator. The difference in the indicators readings show thermal imbalance within the system. Lastly, running user jobs currently on servers are placed above the dotted line in each server box. (1) Figure on the left shows the current status of computing nodes before scheduling the newly arrived jobs ($\lambda_1, \lambda_2, \ldots, \lambda_6$). (2) Figure in the center illustrates thermal consequences of TA task allocation without considering CPU-time of user jobs and relative thermal performance of computing nodes, and (3) Figure on the right depicts the improvement in thermal consequences while considering CPU-time and assigning user jobs in proportion to the relative thermal performance of computing nodes

nodes while considering both the temporal aspects of user job execution and their thermal characteristics. In addition, the proposed approach identifies overloaded computing nodes and performs migrations to achieve thermal efficiency. Finally, CGTAS dynamically adapts the proportion of workload on each computing node to handle the DCs dynamic thermal conditions. Fig. 1 illustrates how considering the relative thermal performance of computing units and the duration of CPU-intensive jobs can improve the thermal performance of the system.

The proposed TA design has the following four differences compared to its TA substitutes. *Firstly*, using K-means clustering, the proposed design is able to intelligently assign long-running CPU-intensive jobs to servers that offer optimal thermal performance. *Secondly*, it uses the Core solution concept to determine a fair and optimal proportion of user jobs for each computing node based on its marginal thermal performance. *Thirdly*, it detects overloaded computing nodes using the proposed Core solution. *Finally*, using the Core , it dynamically adjusts the workload on each computing node to adapt to the DC's extremely dynamic thermal conditions. By considering CPU-time of user jobs and the relative thermal performance of computing nodes, the proposed CGTAS demonstrates superior effectiveness in achieving optimal thermal-efficiency compared to its TA counterparts.

Summary of our main contributions:

- A Core-based formulation of TA job allocation among computing nodes considering their relative thermal performance within the DC environment – it determines a fair and optimal proportion of user jobs to be assigned to each computing node based on its marginal impact on the system thermal performance.
- Devising a mechanism to detect overloaded nodes in a TA context using the Core solution and perform migration of jobs to attain optimal thermal conditions.
- Classification of user jobs based on both TP and CPU-time using K-means clustering to intelligently distribute lengthy CPU-intensive user jobs to servers that offer optimal thermal performance.
- Conducting comprehensive simulations using real-world data, demonstrating that our proposed design achieves substantial energy savings (up to an average of 26.08%) compared to existing TA alternatives.

The subsequent sections are organized as follows: Section II briefly highlights the current state-of-the-art related to TA scheduling. Sections III and IV present the system model and our proposed CGT-based task allocation strategy, respectively. Section V discusses the simulation results, and we draw conclusions in Section VI.

II. Related Work

Running CPU-intensive jobs is one of the main factors affecting the DC thermal conditions as they typically require extended periods of processing time, during which the CPU continuously operates at high utilization rate, generating significant amount of heat. The prolonged execution of CPU-intensive jobs on servers operating at higher temperatures can significantly aggravate cooling efforts due to increased heat dissipation. Therefore, the duration of CPU-intensive tasks stands as a critical factor influencing the DCs cooling costs if not managed intelligently. The impact of CPU-intensive jobs on thermal outcomes is particularly pronounced in densely packed DC environments due to the ambient temperature as multiple servers are placed in close proximity [8, 17]. In such settings, the cumulative heat generated by CPU-intensive tasks may result in the formation of thermal hotspots, areas within the DC where temperatures are significantly higher than the surrounding environment. According to literature [9], thermal hotspots pose increased cooling efforts and a risk to the integrity and reliability of the DC infrastructure, as they can lead to localized overheating and potential hardware failures. By effectively managing CPU-intensive workloads and their subsequent thermal consequences, DC operators can significantly reduce the cooling cost and ensure the reliability and performance of their computing elements [18].

TABLE I: Comparison of existing works

Ref.	Technique	Novelty	Approach	SLA	Task		Target		TC
					TP	Length	IT	Cooling	
[30]	TA scheduling to reduce the cooling cost	task temperature profile	heuristic	✓	✓	✗	✗	✓	server
[31]	TA control strategy to minimize energy consumption	task migrations to mitigate hotspots	heuristic	✗	✓	✗	✗	✓	server
[32]	Thermal, power, and colocation-aware scheduler	colocation-aware, impact of power & thermal constraints	Greedy + GA	✗	✗	✗	✓	✓	server
[8]	TA scheduler to optimize thermal balance considering ambient effect of surrounding nodes	considering ambient effect during allocation/migration	cooperative game	✗	✓	✗	✗	✓	server
[33]	TA workload and cooling manager	Joint optimization of IT and cooling system		✗	✗	✗	✓	✓	server
[34]	a two-step optimization technique to reduce the IT and non-IT power	Joint consideration of non-IT power consumption	SA + Greedy	✗	✗	✗	✓	✓	✗
[12]	TA scheduler considering impact of node failures	impact of node failures on the inlet temperatures	GA + SA	✗	✗	✗	✓	✓	✗
[9]	TA workload distribution to optimize thermal balance marginal thermal performance of nodes	workload allocation based on relative temperature of nodes	coalitional game	✓	✓	✗	✗	✓	server, rack
[13]	TA + DVFS based scheduling to optimize mean temperature margin of the chip	time-varying workload features, random inter-task communication	RL	✗	✗	✗	✓	✗	✗
[14]	TA scheduling of realtime tasks to minimize device temperature	analytical model to find trade-off between node temperature & performance	Just enough + Performance policy	✓	✗	✗	✓	✗	✗
[35]	TA scheduler assisted by deep neural network-based server prediction model	DNN, task allocation based on relative temperature, task heterogeneity	heuristic	✓	✓	✓	✗	✓	server
[15]	TA data replica placement	heat re-circulation	ACO	✗	✗	✗	✗	✓	✗
[36]	Spatio-TA scheduler to reduce sla violations, cooling and energy cost	dynamic price and temperature consideration	huristics	✓	✗	✗	✓	✓	zone
Ours	TA adaptive scheduler to avoid hotspots and reduce cooling cost	task CPU-time, detection of thermally overloaded nodes using the Core solution	Coalitional game theory, K-means	✓	✓	✓	✗	✓	server, rack

To address the thermal challenges posed by CPU-intensive jobs in a DC, dynamic TM strategies are essential. Existing TM approaches include TA designs that prioritize the allocation of CPU-intensive tasks to servers with optimal thermal conditions. TA scheduling has proven to be highly cost-effective solution for addressing thermal challenges in DCs [12, 15, 9]. A significant body of literature is dedicated to studying TA scheduling to reduce energy costs and ensure the reliable operation of DC services. Table I highlights existing TA strategies in terms of their design strategy, novelty, optimization approach, consideration of SLA violations, consideration of task TP and CPU-time (length/duration), targeted energy level, and thermal constraints (TC). Cheng et al. [19] categorize energy conservation technologies in cloud DCs into three main groups: (1) techniques focusing on energy savings in IT equipment, (2) strategies targeting the reduction of cooling expenses in DCs, and (3) approaches for joint optimization. Before diving into the details of each group of strategies, we briefly highlight the suitability of the Coalitional Game (CG) for the TA scheduling problem in our paper.

Existing literature [20, 21, 22, 23, 24] shows that Game Theory (GT) is a promising tool for complex optimization problems. GT can be applied to any field where the optimization scenario can be modeled as a strategic interaction among multiple rational agents. For instance, studies in [21, 24] model scenarios where resources are competitively allocated across multiple battlefields. Each player aims to outperform the other by maximizing their allocation's effectiveness against the opponent's strategy. The study in [21] focuses on power allocation to prevent jamming in underwater communication, emphasizing how strategic resource distribution can mitigate external threats using Colonel Blotto Game (CBG). Similarly, the study in [24] uses CBG to strategically allocate channel against aerial eavesdroppers, aiming to optimize the security of power transmissions.

On the contrary, in our work, the players (DC nodes) share a common goal of minimizing cooling energy by optimizing thermal conditions. Unlike the competitive allocation in CBG, where players work against each other, the CG in our paper fosters collaborative approach among the participants to attain globally optimal solution. The proposed game models management of computing resources holistically that is more suited to addressing complex, system-wide challenges like TM in DCs. While non-cooperative games such as the CBG provides insights into strategic resource distribution under competitive conditions, the CG is motivated by the need for cooperation among the agents, making it particularly effective in environments where joint optimization can lead to significant operational improvements. This emphasizes the suitability of CGs for addressing the multi-faceted challenges of TM in modern DCs.

A. Minimizing Power Usage of Cooling System

Cheng et al. [19] categorizes the methods for DC cooling efficiency into: (1) layout and airflow management [25, 26], (2) TA scheduling [27], and (3) other techniques including liquid cooling [28], and waste heat recovery [29]. In this paper, we focus on TA scheduling. In this context, Li et al. [12] propose a TA approach, investigating the impact of node failures on power usage. Niknia et al. [13] combine TA and DVFS strategies, modeling the task allocation in multiprocessor systems-on-chip as a SMDP. Ozceylan et al. [14] develop a TA scheduler for workloads with known resource usage and duration, exploring the trade-off between system throughput and server temperature employing a thermodynamics model. Akbar et al. [8] devise a CG-based scheduler considering the ambient temperature of computing nodes to reduce the cooling cost by minimizing thermal imbalance and hotspots. However, studies in [12, 13, 14, 8] are designed for homogeneous DCs, ignoring the thermal impact of heterogeneous resources.

In a heterogeneous DC environment, Oxley et al. [32] design three resource management mechanisms, employing a genetic algorithm (GA), a greedy algorithm, and a colocation-aware technique to reduce energy consumption without violating thermal constraints (TCs). Similarly, Akbar et al. [9] propose a Shapley value-based TA scheduler for heterogeneous DCs, aiming to allocate incoming jobs to computing nodes in proportion to their relative temperatures while considering TCs of servers and racks. However, the strategies in [14, 8, 32, 9] may lead to thermally overloaded servers in the long run due to the long running CPU-intensive workloads as the CPU-time of user jobs is ignored during task allocation. Lastly, a TA scheduling heuristic assisted by a deep neural network-based server temperature forecasting model considering user job heterogeneity is presented in [35]. The authors predict node temperature using factors such as CPU utilization rate, CPU capacity, server type, etc. However, the proposed design is heuristic-based and depends on the accuracy of the prediction model, which may not produce optimal results in environments other than it has been trained for.

B. Minimizing Energy Consumption of Servers

To reduce the energy cost of DC at the server level. For instance, Roy et al. [37] introduce SATORI, a novel strategy designed for multicore systems to equitably and efficiently allocate resources. The objective is to strike a balance between system performance and fairness among co-located user workloads. To prevent hotspots, Kumar et al. [38] propose an adaptive CPU control strategy based on Dynamic Voltage and Frequency Scaling (DVFS). This control strategy utilizes a prediction model to accurately forecast the temperature of the computing node and adjust the CPU frequency accordingly. However, the proposed approach may lead to SLA violations due to adjusting the CPU frequency. Furthermore, the objective of studies in [37, 38] is to attain energy-efficiency at the server level, ignoring the dynamic thermal conditions at the DC level due to various factors such as the ambient effect of surrounding nodes, thermally overloaded computing nodes, runtime of CPU-intensive jobs, etc.

C. Minimizing Energy Consumption of Servers

Researchers have offered various strategies to reduce the energy cost of DC at the server level. For instance, Roy et al. [37] introduce SATORI, a novel strategy designed

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2024.3414671

for multicore systems to equitably and efficiently allocate resources. The objective is to strike a balance between system performance and fairness among co-located user workloads. To prevent hotspots, Kumar et al. [38] propose an adaptive CPU control strategy based on Dynamic Voltage and Frequency Scaling (DVFS). This control strategy utilizes a prediction model to accurately forecast the temperature of the computing node and adjust the CPU frequency accordingly. However, the proposed approach may lead to SLA violations due to adjusting the CPU frequency. Furthermore, the objective of studies in [37, 38] is to attain energy-efficiency at the server level, ignoring the dynamic thermal conditions at the DC level due to various factors such as the ambient effect of surrounding nodes, thermally overloaded computing nodes, runtime of CPU-intensive jobs, etc.

D. Joint Optimization of Computing and Cooling

Recently, researchers have explored joint optimization of energy consumption at the computing and cooling system levels. MirhoseiniNejad et al. [39] investigate joint optimization in heterogeneous DCs, aiming to optimize energy consumption while considering servers TCs. Unlike conventional strategies that define a single temperature setpoint for all DC regions, their approach sets different setpoints for distinct regions. Feng et al. [34] introduce an energy-aware VM placement design to optimize power consumption across servers, cooling systems, and the network. Arroba et al. [40] propose heuristics and meta-heuristic techniques based on simulated annealing (SA) to jointly optimize cooling and computing energy.

Recently, Rostami et al. [41] introduce a novel approach to jointly manage the DC workload and cooling system, focusing on solving the typically nonlinear optimization challenges associated with thermal management, converting them into mixed integer linear programming problems that can be solved with heuristic methods. The goal is to optimize workload distribution to minimize cooling power consumption, taking into account the ambient conditions and operational status of servers. The proposed framework not only proactively adjusts to anticipated changes but also reacts to real-time server utilization and ambient temperature fluctuations, aiming to reduce overall energy costs while maintaining optimal server performance. However, the strategies in [39, 34, 40, 41] ignore the thermal impact of long running CPU-intensive jobs in large-scale DCs. Hence, the proposed approaches may lead to thermally overloaded computing nodes as they do not adjust the workload dynamically at runtime.

III. System Model and Problem Formulation

In this section, we present the DC, power consumption, and thermodynamics model followed by the problem formulation. Table II lists all the symbols with their meanings.

A. Data Center Model

DC constitutes a complex ecosystem comprising a multitude of critical components, including servers, switches, and various infrastructure elements. These components demand a substantial amount of power supply, particularly when executing

TABLE II: List of Notations

Notation	Meaning
ζ	Set of N servers indexed by i
ζ^{HS}	Set containing hotspots
ζ^{OL}	Set of overloaded servers
λ	Set of J number of user jobs indexed by j
λ^{MIG}	Set of user jobs to migrate
λ^{NEW}	Set of newly arrived jobs
λ^{CLUSTER}	Set of clusters created using K-means
π_i	Set of jobs run by a computing node i
ρ	Priority value of a user job
γ	Length of a user job
F	Finishing time of a user job
Γ	Deadline of a user job
α	Boolean indicating task assignment
P^{IDLE}	Power consumption rate at idle state
P^{ACTIVE}	Power consumption rate at active state
CRAC	Computer Room Air Conditioner
P^{CRAC}	Power consumed by the CRAC
P^{COMP}	Power consumed for computing
$\mathcal{T}^{\text{SUP}}$	Supplied temperature from CRAC
CPU^{UTIL}	CPU utilization rate
CoP	Coefficient of performance
P^{DC}	Power consumed by the data center
$a_{i\check{i}}$	Ambient effect of node i on its neighbor $\check{i}$
Q_i^{OUT}	Heat dissipated by node i
Q_i^{IN}	Heat recirculating into the node i
Q_i^{EXT}	Heat extracted from node i
$\mathcal{T}_i^{\text{IN}}$	Inlet temperature of node i
$\mathcal{T}_i^{\text{OUT}}$	Outlet temperature of node i
Δ_T	Temperature rise
$\mathcal{T}^{\text{H}}$	Temperature of the hottest node
$\mathcal{T}^{\text{C}}$	Temperature of the coolest node
$\hat{\hat{\tau}}$	Threshold temperature
$\mathcal{K}_t$	Thermal conductivity constant
ϱ	Air density
$\mathcal{H}_c$	Specific heat capacity of air
f	Air-flow rate
$\mathcal{D}_{i\check{i}}$	Distance between node i and $\check{i}$
TD	Thermal difference
TI	Thermal imbalance
$\mathcal{A}$	Set of agents/players
υ	Utility or payoff function
$\mathcal{G} = (\mathcal{A}, \upsilon)$	Game with agents $\mathcal{A}$ and payoff function υ
$\mathcal{C} \subseteq \mathcal{A}$	Coalition of agents

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2024.3414671

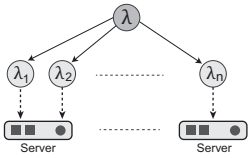

Fig. 2: User job

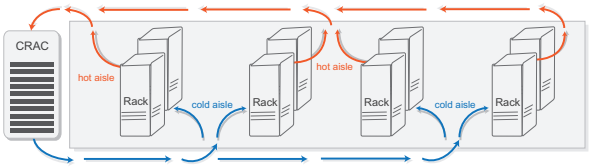

Fig. 3: Demonstration of a DC-level cooling system

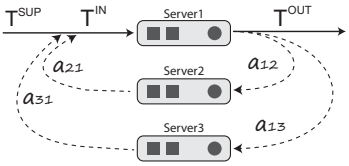

Fig. 4: Ambient effect

CPU-intensive tasks. Within the DC, servers are organized into racks, with each rack having one or more switches designed to facilitate inter-connectivity among the servers. The harmonious integration and management of these intricate components are critical to ensure uninterrupted and energy-efficient DC operation. Understanding the interplay between power consumption, thermal dynamics, and infrastructure design is essential for optimizing DCs performance while minimizing energy expenditure and environmental impact.

In the subsequent discourse, we delve deeper into the DC intricate aspects, focusing on relationship between energy, temperature, and operational efficiency in contemporary DCs. We consider a typical DC, where a job scheduler receives user requests and dispatches them to available computing nodes. Consider a heterogeneous DC with N number of servers denoted by $\zeta = \{\zeta_1, \zeta_2, \ldots, \zeta_N\}$. Moreover, a server may have more than one processing cores denoted as $\zeta_i = \{\zeta_{i1}, \zeta_2, \ldots, \zeta_{iC}\}$. Each server may have a different computing capacity and power consumption rate depending on the architecture, number of cores, and server enclosure.

B. Incoming User Jobs

Let $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_J\}$ denotes the set of incoming user jobs. As depicted in Fig. 2, each user job λ_j can be further divided into multiple tasks that can run in parallel on the same computing node or on different computing nodes. Mathematically, $\lambda_j = \{\lambda_{j1}, \lambda_{j2}, \ldots, \lambda_{jK}\}$. Moreover, $\alpha_{jk\_ic} = 1$ denotes that the task k of user job j is assigned to the core c of i^{th} server, and 0 otherwise.

C. Power Consumption Model

We consider an air-cooled DC with raised floor as shown in Fig. 3. According to literature [12], computing nodes and CRAC units are the main contributors to the DC energy cost, accounting for more than 95% of the DC total energy consumption [42]. In this work, we quantify the DC total energy consumption as the energy consumed by the computing nodes and the cooling system. Let P_i^{IDLE} and P_i^{ACTIVE} be the power consumption rate of server ζ_i in its idle and active states, respectively. Hence, Eq. (1) depicts the power consumed by ζ_i based on its CPU utilization rate.

$$P_i = P_i^{\text{IDLE}} + \text{CPU}^{\text{UTIL}} \left(P_i^{\text{ACTIVE}} - P_i^{\text{IDLE}} \right) \tag{1}$$

Let P^{COMP} denotes the total power consumed by all the computing nodes in the DC and can be expressed as:

$$P^{\text{COMP}} = \sum_{i=1}^{N} P_i \tag{2}$$

The power consumption rate of the CRAC (P^{CRAC}) directly relates to the power consumed by the computing nodes and can be computed using Eq. (3):

$$P^{\text{CRAC}} = \frac{P^{\text{COMP}}}{CoP(\mathcal{T}^{\text{SUP}})} \tag{3}$$

where $\mathcal{T}^{\text{SUP}}$ defines the supplied temperature from the CRAC. CoP quantifies the efficiency of the heat recirculation system of the DC environment and can be defined as:

$$CoP(\mathcal{T}^{\text{SUP}}) = 0.0068\ \mathcal{T}^{\text{SUP}^2} + 0.0008\ \mathcal{T}^{\text{SUP}} + 0.4580 \tag{4}$$

Now, using the above formulation, we can compute the DC total power consumption rate as:

$$P^{\text{DC}} = P^{\text{COMP}} + P^{\text{CRAC}} \tag{5}$$

$$P^{\text{DC}} = \left(1 + \frac{1}{CoP(\mathcal{T}^{\text{SUP}})} \right) P^{\text{COMP}} \tag{6}$$

D. Thermodynamics

Thermodynamics illustrates the heat dissipation of computing nodes and its effect on the surrounding nodes as shown in Fig. 4. The $\mathcal{T}^{\text{SUP}}$ from the cooling system and the heat dissipated by one node affect the inlet temperature of others. A rise in the temperature of a computing node affects the thermal profile of the neighboring servers, also known as the ambient affect. This phenomena is denoted by a cross-interference coefficient matrix [31] as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1N} \\ a_{21} & a_{22} & \ldots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N1} & a_{N2} & \ldots & a_{NN} \end{bmatrix}$$

where, $a_{i\check{\imath}}$ is the (thermal) ambient-effect of node i on its surrounding nodes denoted by $\check{\imath}$. Let $\mathcal{K}_t$ denotes the thermal-conductivity constant of air and $\mathcal{D}_{i\check{\imath}}$ is the distance of node i from $\check{\imath}$. Eq. (7) defines the ambient effect on each computing nodes as follows:

$$a_{i\check{\imath}} = \mathcal{T}_i^{\text{IN}} * \mathcal{K}_t / \mathcal{D}_{i\check{\imath}} \tag{7}$$

Let Q_i^{IN} and Q_i^{OUT} be the heat recirculating and heat dissipated by node i.

$$Q_i^{\text{OUT}} = Q_i^{\text{IN}} + \mathcal{E}_i \tag{8}$$

where $\mathcal{E}_i$ denotes the node i energy consumption. In terms of inlet and outlet temperatures, we can calculate Q_i^{OUT} and Q_i^{IN} using Eq. (9) and Eq. (10), respectively.

$$Q_i^{\text{OUT}} = \varrho f_i \mathcal{H}_c \mathcal{T}_i^{\text{OUT}} \tag{9}$$

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2024.3414671

$$Q_i^{\mathrm{IN}} = \varrho f_i \mathcal{H}_c \mathcal{T}_i^{\mathrm{IN}} \tag{10}$$

where ϱ, $\mathcal{H}_c$, and f_i defines the density, specific heat capacity and flow rate of air, respectively. Moreover, in terms of heat re-circulation, the inlet, outlet, and supplied temperature to a computing node i can be written as:

$$Q_i^{\mathrm{IN}} = \sum_{i,\check{i}=1}^{N} a_{\check{i}i} Q_{\check{i}}^{\mathrm{OUT}} + Q_i^{\mathrm{SUP}} \tag{11}$$

$$Q_i^{\mathrm{OUT}} = \sum_{i,\check{i}=1}^{N} a_{\check{i}i} \varrho f_i \mathcal{H}_c Q_{\check{i}}^{\mathrm{OUT}} + Q_i^{\mathrm{SUP}} + \mathcal{E}_i \tag{12}$$

$$Q_i^{\mathrm{SUP}} = \varrho (f_i - a_{\check{i}i} f_{\check{i}}) \mathcal{H}_c \mathcal{T}^{\mathrm{SUP}} \tag{13}$$

putting the value of Q_i^{SUP} from Eq. (13) into Eq. (11) and Eq. (12), we get:

$$Q_i^{\mathrm{IN}} = \sum_{i,\check{i}=1}^{N} a_{\check{i}i} Q_{\check{i}}^{\mathrm{OUT}} + \varrho (f_i - a_{\check{i}i} f_j) \mathcal{H}_c \mathcal{T}^{\mathrm{SUP}} \tag{14}$$

$$Q_i^{\mathrm{OUT}} = \sum_{i,\check{i}=1}^{N} a_{\check{i}i} Q_{\check{i}}^{\mathrm{OUT}} + \varrho (f_i - a_{\check{i}i} f_{\check{i}}) \mathcal{H}_c \mathcal{T}^{\mathrm{SUP}} + \mathcal{E}_i \tag{15}$$

Finally, the heat extracted can be calculated as:

$$Q_i^{\mathrm{EXT}} = \varrho f_i \mathcal{H}_c (\mathcal{T}_i^{\mathrm{OUT}} - \mathcal{T}_i^{\mathrm{IN}}) \tag{16}$$

E. Problem Formulation

Without proper thermal management, thermal-imbalances and hotspots may arise inside DCs. It is evident from the literature [8] that the energy cost during the life-time of a hotspot is higher than the energy-savings during its computations. In this work, the main objective of the proposed system is to minimize the thermal imbalance, avoid hotspots, and at the same time, keep the temperature values of servers to lowest. In the following text, we highlight the relationship between thermal uniformity and energy consumption. The thermal difference between any two nodes a and b can be defined as $\mathcal{T}_b - \mathcal{T}_a$ where $\mathcal{T}_b > \mathcal{T}_a$. We define the total thermal difference as the sum of the temperature difference of each computing node from the temperature of the hottest computing node at any time interval. Mathematically:

$$\mathrm{TD} = \sum_{i \in N} \left(\mathcal{T}^{\mathrm{H}} - \mathcal{T}_i \right) \tag{17}$$

The higher the value of TD, the lower the thermal balance inside DC environment and vice versa. However, the lower the value of TD doesn't necessarily mean the better the thermal balance achieved and vice versa. This is because the optimal thermal conditions also depend on how high the value of $\mathcal{T}^{\mathrm{H}}$ is. To understand how the energy consumed by the CRAC units increases with the increase in server temperature, Section III-D depicts the relationship between the server temperature, the supplied temperature from CRAC units, and the energy consumed by the CRAC units for the supplied temperature. The energy consumption of the CRAC units increases when the supplied temperature is lowered, and vice versa. Moreover, the supplied temperature from CRAC is lowered when the temperature of servers is high. Hence, the thermal balance is optimal if we attain lower values of $\mathcal{T}^{\mathrm{H}}$ and at the same time, the value of TD is minimized. Therefore, in this paper, we define thermal imbalance as:

$$\mathrm{TI} = \left(\mathcal{T}^{\mathrm{H}} + \mathcal{T}^{\mathrm{C}} \right) \mathrm{TD} \tag{18}$$

Based on the above equation, the thermal imbalance is minimized if the temperature value of the hottest, coolest computing node, and the value of TD are minimized at the same time. Therefore, the objective of the proposed system is to minimize the thermal imbalance inside DC environment as depicted in the following equation:

$$minimize \ \ (\mathrm{TI}) \tag{19}$$

Subject to:

$$\sum_{i,j,k=0}^{N,J,K} \alpha_{jk\_ic} = 1 \tag{20}$$

$$F_j \leq \Gamma_j, \quad \forall j \in \lambda \tag{21}$$

$$\mathcal{T}_i \leq \hat{\hat{\tau}}_i, \quad \forall i \in \zeta \tag{22}$$

According to Eq. (20), each task $k \in \lambda_j$ should be assigned to only one processing core at any time. The constraint in Eq. (21) states that every user job finishing time F_j should be less than its deadline Γ_j – enough resources must be assigned to avoid SLA violations. In other words, in addition to the task TP and CPU-time, the CGTAS also considers the deadline of user jobs during the scheduling process. Hence, all user jobs receive fair amount of resources. Finally, Eq. (22) depicts that the computing nodes temperatures should not violate their thermal constraints $\hat{\hat{\tau}}_i$.

IV. The CGTAS Design

Fig. 5 provides an overview of the proposed system highlighting the novel contributions. The proposed CGTAS has two main components: (1) The Core-based task scheduler, and (2) Job manager that clusters and sorts incoming user jobs based on their thermal and temporal characteristics. The proposed CGTAS model uses the Core [10, 16] solution concept from CGT to distribute the incoming jobs among the computing nodes based on their relative thermal performance within the DC. The proposed design has the following differences compared to existing TA designs: firstly, it considers the duration of user jobs during task allocation and migration. Secondly, it uses the Core solution concept to determine the optimal proportion of user jobs to be assigned to each computing node based on its marginal impact on the DC thermal performance. Thirdly, it detects the thermally overloaded nodes using the proposed Core solution. Finally, the proposed strategy is adaptive as it adjusts the workload on each node by migrating jobs from thermally overloaded nodes to underloaded ones.

The proposed CGTAS performs the following steps to attain better thermal uniformity and minimize the DC energy cost. Firstly, it partitions the set of incoming user jobs into clusters based on their TPs and sorts them based on their length and

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2024.3414671

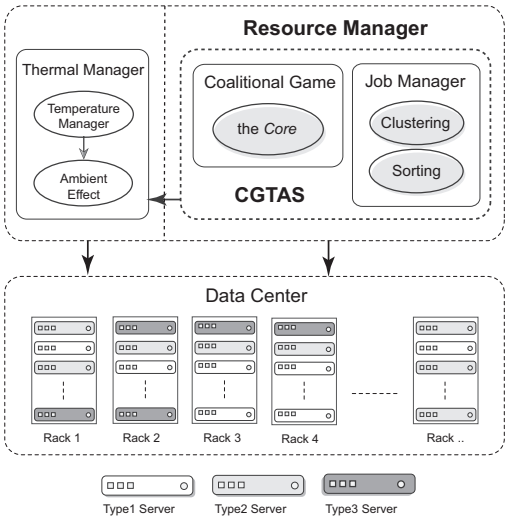

Fig. 5: Architecture of the proposed system

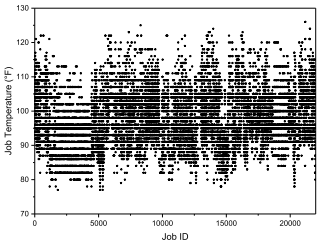

Fig. 6: Distribution of jobs based on their temperature

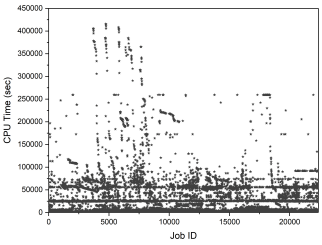

Fig. 7: Distribution of jobs based on their CPU time

Algorithm 1: K-means

Input: Data points (Temperature values), K (number of clusters)

Output: K number of clusters

1 Randomly select k number of data points as the cluster centers
2 For each data point, compute its distance from the cluster centers
3 Assign each data point to a cluster with minimum distance
4 Recalculate the cluster centers
5 Repeat step 2,3, and 4 until convergence or for specified number of iterations

TPs in each cluster. Secondly, it uses the Core solution concept to find the proportion of workload to be assigned to each computing node based on their relative thermal readings – computing nodes with lowest temperature values receive more jobs. Thirdly, using the Core solution concept, it identifies overloaded computing nodes in a TA context and considers migrations to optimize the DC thermal conditions. Finally, the second and third steps are followed at both the DC level as well as the rack level to achieve optimal thermal balance.

A. User Jobs Classification using K-means clustering

We obtain the workload trace from the "Center of Computational Research (CCR), State University of New York at Buffalo." The Buffalo dataset encompasses a substantial corpus of 22,385 user jobs observed over a 30-day period. Fig. 6 and 7 depict the distribution of jobs based on their temperature rise and CPU time, respectively. Incoming jobs can be classified based on their thermal consequences in terms of temperature rise from very hot to very cold (very hot, hot, warm, cold, very cold). In order to classify the incoming user jobs, we use K-means clustering in our study.

K-means is a machine learning algorithm used to develop a clustering model based on some input data points. The main idea is to minimize the distance between data points in each cluster and its center. There are different distance metric available such as the Euclidean and the Manhatan distance to calculate the distance between data points and cluster centers. In this paper, we use the Euclidean distance as a measure of similarity among data points and the cluster centers. Algorithm 1 depicts the main steps of the K-means algorithm. The algorithm takes data points (temperature data) and the number of clusters represented by K as input. We provide $K = 5$ for this study. The algorithm starts by randomly choosing K points as the centers of K clusters/groups. After choosing random centers, for each data point, the algorithm computes distance between the data point (temperature value) and the cluster centers. In step 3, the algorithm assigns each data point to a cluster with minimum distance. In step 4, the algorithm recalculates the center of each cluster by taking the mean value within the cluster. Finally, the algorithm repeats step 2, 3, and 4 until the it converges or the specified number of iterations are performed.

Table III shows the results of K-means clustering applied on the Buffalo dataset. After getting the clusters, the proposed approach sorts the array of clusters based on their centers. In other words, the proposed approach schedules the jobs in a cluster with highest center value first. Furthermore, since we consider the length of the incoming user jobs, the user jobs within each cluster are sorted based on their priority ρ from highest to lowest value calculated using the following formula:

$$\rho(\lambda_{jk}) = \frac{\gamma(\lambda_{jk}) \times \Delta_T(\lambda_{jk})}{\gamma(\lambda_{jk}) + \Delta_T(\lambda_{jk})} \tag{23}$$

where $\gamma(\lambda_{jk})$ and $\Delta_T(\lambda_{jk})$ denotes the length and temperature rise due to running a task λ_{jk}, respectively. Furthermore, the k-means clustering takes 0.05 seconds to build the model. However, increasing the number of parallel threads from 1 to 3 reduces the computation time from 0.05 to 0.03 seconds. Number of threads higher than 3 does not improve the performance of the k-means clustering for our dataset. Therefore, we keep the number of threads to 3.

B. Coalitional Game-based TA Task Allocation

In a coalitional game (CG), the players make coalitions to maximize their benefits. The Core provides a stable and efficient way to distribute the payoff (user jobs) of any coalition among its members (computing nodes).

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2024.3414671

Cluster	Centroid	Instances	Job Category
1	110.98	4029 (18%)	Very hot
2	102.34	6491 (29%)	Hot
3	95.24	6043 (27%)	Warm
4	89.74	4253 (19%)	Cold
5	84.00	1576 (7%)	Very Cold

TABLE III: K-means clustering results

Definition: (Coalitional game with Transferable Utility) A CG with TU is defined by a tuple $\mathcal{G} = (\mathcal{A}, \upsilon)$, where $\mathcal{A}$ is the set of N agents/players and $\upsilon : 2^{\mathcal{A}} \rightarrow R$ defines the utility function. $\upsilon(C)$ assign each coalition $C \in \mathcal{A}$ a real value payoff. Finally, $\mathcal{A}$ is a finite set. The main goal of coalition formation games is to encourage and ensure grand coalition among the agents. This can be achieved by a payoff distribution mechanism that can define fair and efficient allocation of payoff (cost or benefit) among the players of the game to motivate the players towards the grand coalition. There are two main solution concepts such as the Core and the Shapley value used to achieve the said goal. In this paper, we use the Core value as it offers stable cost/benefit distribution compared to the Shapley value [16].

In the proposed CG, the set of agents/players $\mathcal{A}$ contains all the computing nodes. The transferable utility in the game is the incoming user jobs. Finally, the CG uses Eq. (24) to allocate the incoming user jobs among the computing nodes. Since the CGTAS uses the Core solution concept, we describe the properties of the proposed game in general and the Core solution in particular in the subsequent discourse.

$$\upsilon(C) = \begin{cases} \dfrac{\lambda}{N}, & if\ \mathcal{T}^{\mathrm{H}} - \mathcal{T}^{\mathrm{C}} = 0 \\ 0, & if\ \mathcal{T}^{\mathrm{H}} - min\{\mathcal{T}_{\breve{i}} \mid \breve{i} \in C\} = 0 \\ \dfrac{\sum\ \mathcal{T}^{\mathrm{H}} - \{\mathcal{T}_{\breve{i}} \mid \breve{i} \in C\}}{\sum_{i \in N}\ \mathcal{T}^{\mathrm{H}} - \mathcal{T}_i} \times \lambda, & Otherwise \end{cases} \tag{24}$$

where $\mathcal{T}^{\mathrm{H}}$ and $\mathcal{T}_{\breve{i}}$ define the temperature of the hottest node in the DC and the node $\breve{i}$ in a coalition C, respectively. Similarly, $\mathcal{T}_i$ is the temperature of i^{th} node in ζ, and λ is the set of user jobs to be assigned to computing nodes in C.

Definition: (the Core) A payoff vector P is in the Core of the game $\mathcal{G} = (\mathcal{A}, \upsilon)$ if the following two conditions are met:
(i) $\sum_{i \in \mathcal{A}} \mathcal{P}_i = \upsilon(\mathcal{A})$, and
(ii) $\forall\ C \subseteq \mathcal{A}, \sum_{j \in C} \mathcal{P}_j \leq \upsilon(C)$
Before discussing the non-emptiness of the Core for our proposed workload distribution game, we first highlight some of the important definitions related to the CGs with TU. Introducing these concepts will help us understand the properties of our proposed game.

Definition: (Superadditive Game) $\mathcal{G} = (\mathcal{A}, \upsilon)$ is a SuperAdditive game if $\forall\ C_i, C_j \subset \mathcal{A}$ *and* $C_i \cap C_j = \emptyset$, *then* $\upsilon(C_i \cup C_j) \geq \upsilon(C_i) + \upsilon(C_j)$. Informally, the value of a grand coalition is at least equal to the sum of payoffs of all non-overlapping set of coalitions in a superadditive game. Hence, in superadditive games, the grand coalition always provides maximum benefits to its members. The proposed workload distribution game with utility function defined in Eq. (24) is SuperAdditive as for any two non-intersecting subsets C_1 and C_2 of $\mathcal{A}$, the $\upsilon(C_1 \cup C_2) = \upsilon(C_1) + \upsilon(C_2)$.

Definition: (Additive Game) $\mathcal{G} = (\mathcal{A}, \upsilon)$ is an Additive game if $\forall\ C_i, C_j \subset \mathcal{A}$ and $C_i \cap C_j = \emptyset$, then $\upsilon(C_i \cup C_j) = \upsilon(C_i) + \upsilon(C_j)$. Informally, in an additive game, the non-overlapping coalitions C_i and C_j can never affect one another. Since, all Additive games are SuperAdditive games by definition, the proposed game is also Additive in nature.

Definition: (Constant-Sum Game) $\mathcal{G} = (\mathcal{A}, \upsilon)$ is a constant-sum game if $\forall\ C \subset \mathcal{A}, \upsilon(C) + \upsilon(\mathcal{A} \backslash C) = \upsilon(\mathcal{A})$. Since the proposed workload distribution game with utility function defined in Eq. (24) is an Additive game, the expression $\upsilon(C) + \upsilon(\mathcal{A} \backslash C_i) = \upsilon(\mathcal{A})$ holds for any coalition C_i as $\mathcal{A} \backslash C_i$ and C_i are two non-overlapping subsets of $\mathcal{A}$ (Additivity property).

Definition: (Convex Game) $\mathcal{G} = (\mathcal{A}, \upsilon)$ is a convex game if $\forall\ C_i, C_j \subset \mathcal{A}, \upsilon(C_i \cup C_j) \geq \upsilon(C_i) + \upsilon(C_j) - \upsilon(C_i \cap C_j)$. To prove that the proposed game is a convex game, let us consider $\mathcal{A} = \{1, 2, 3\}$ and $C_1 = \{1, 2\}$, $C_2 = \{2, 3\}$ are two overlapping subsets of $\mathcal{A}$. Now, the $C_1 \cap C_2 = \{2\}$. The proposed game is a convex game if:

$$\forall\ C_1, C_2 \subset \mathcal{A}, \upsilon(C_1 \cup C_2) \geq \upsilon(C_1) + \upsilon(C_2) - \upsilon(C_1 \cap C_2)$$

Using the Additivity property (proved earlier), $\upsilon(C_1) = \upsilon(\{1\}) + \upsilon(\{2\})$ and $\upsilon(C_2) = \upsilon(\{2\}) + \upsilon(\{3\})$. Putting these values in Eq. IV-B, we get:

$$\upsilon(\{1, 2, 3\}) \geq \upsilon(\{1\}) + \upsilon(\{2\}) + \upsilon(\{2\}) + \upsilon(\{3\}) - \upsilon(\{2\})$$

$$\upsilon(\{1, 2, 3\}) \geq \upsilon(\{1\}) + \upsilon(\{2\}) + \upsilon(\{3\}))$$

Hence the proposed game is a Convex game.

Theorem: The workload distribution game $(\mathcal{A}, \upsilon)$ with TU is a convex game and therefore, it has a non-empty Core.

Proof: We have proved that the proposed workload distribution game is a convex game. Proving every convex game has a non-empty Core is well-studied in literature [11]. To prove that the proposed game has a non-empty Core, we use the concept of balanced game.

Definition: A vector W is a balanced collection of weights if it assigns a positive weight W_C to each $C \in \mathcal{A}$ and $\forall j \in \mathcal{A}, \sum_{C: j \in C} W_C = 1$.

Theorem: (Bondareva-Shapley) The proposed workload distribution game $\mathcal{G} = (\mathcal{A}, \upsilon)$ has a non-empty Core if:

$$\forall W_C, \sum_{C \subseteq \mathcal{A}} W_C \times \upsilon(C) \geq \upsilon(\mathcal{A})$$

Proof: Based on the above definition, the proposed game $(\mathcal{A}, \upsilon)$ where υ is the utility function defined in Eq. 24 has a non-empty Core if and only if the linear program the following equation yields exactly $\upsilon(\mathcal{A})$.

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2024.3414671

$$\begin{aligned} &Maximize \sum_{j \in \mathcal{A}} \mathcal{P}_j \\ &Subject\ to\ \forall C \in \mathcal{A} : \sum_{j \in C} \mathcal{P}_j \leq \upsilon(C) \end{aligned} \tag{25}$$

where $\mathcal{P}_j$ denotes the payoff of agent j when it chooses not to be part of any coalition. Using the strong linear program duality, the following linear program results the same output:

$$\begin{aligned} &Minimize \sum_{C \in \mathcal{A}} W_C \times \upsilon(C) \\ &Subject\ to\ \forall j \in \mathcal{A} : \sum_{C : j \in C} W_C = 1, and \\ &\forall C \subseteq \mathcal{A} : W_C \geq 0 \end{aligned} \tag{26}$$

Now, putting the value of $\upsilon(C)$ from Eq. 24 into the Eq. 26, we get $\sum_{C \subseteq \mathcal{A}} W_C \times \upsilon(C) = \upsilon(\mathcal{A})$ for every balanced W_C and for all the three scenarios depicted in Eq. 24. Therefore, the Core of the proposed game is non-empty.

C. Detecting overloaded nodes using the Core

Due to the heterogeneity of user jobs in terms of runtime, it is possible that some computing nodes get overloaded compared to others for a long period of time. In contrast to the prior investigation in [9], our proposed approach suggests a novel means of detecting overloaded computing nodes based on the Core solution from CGT. After detecting the overloaded computing nodes, the proposed approach migrates user jobs from overloaded to the most appropriate computing node to attain best thermal outcomes.

The proposed design identifies overloaded nodes by calculating the difference between the number of jobs currently running on these nodes and the number of jobs suggested by the Core based on their current temperature readings. If the number of jobs running on a computing node exceeds the Core's recommended allocation, it is deemed overloaded from a thermal perspective. For instance, consider the following scenario. A server ζ_i is operating at temperature $\mathcal{T}_i$ while running π_i workload. Assuming that the Core solution suggests $\lambda_i^{\text{Core}} < \pi_i$ number of user jobs to be assigned to ζ_i based on its temperature $\mathcal{T}_i$. Hence, the computing node is considered as overloaded, denoted by ζ_i^{OL}, in thermal-aware context. Mathematically:

$$\zeta_i^{\text{OL}} = \begin{cases} 1, & if\ \pi_i > \lambda_i^{\text{Core}} \\ 0, & Otherwise \end{cases} \tag{27}$$

Hence, when $\zeta_i^{\text{OL}} = 1$, we need to migrate $\pi_i - \lambda_i^{\text{Core}}$ number of user jobs from ζ_i to enhance the thermal performance of the system.

D. Proposed Scheduling Algorithm

In the following text, we present the proposed CGTAS algorithm based on the **Core** solution concept presented in Section IV-B. Algorithm 2 outlines the crucial steps undertaken by the CGTAS to optimize the thermal consequences inside DCs. The algorithm takes two primary inputs: the set of available servers ζ and the set of newly arrived user jobs λ^{NEW}. The objective is to determine the most advantageous assignment map, one that optimizes thermal consequences (temperature rise) of assigning the workload λ to ζ.

Algorithm 2: CGTAS

Input: $\zeta, \lambda^{\text{NEW}}$
Output: Optimal Assignment Map for Tasks λ to ζ

1 $\lambda \leftarrow \sum_i^N \pi_i \cup \lambda^{\text{NEW}}$
2 $\lambda^{\text{Core}} \leftarrow$ FindCore(λ_t, ζ)
3 $\zeta^{\text{HS}} \leftarrow$ GetHotspots(ζ)
4 $\lambda^{\text{MIG}} \leftarrow$ GetTasksToMigrate(ζ^{HS})
5 $\zeta^{\text{OL}} \leftarrow$ GetOverloadedServers$(\zeta, \lambda^{\text{Core}})$
6 $\lambda^{\text{MIG}} \leftarrow \lambda^{\text{MIG}} \cup$ GetCoolestTasksToMigrate$(\zeta^{\text{OL}}, \lambda^{\text{Core}})$
7 $\lambda \leftarrow \lambda^{\text{MIG}} \cup \lambda$
8 $\lambda^{\text{CLUSTER}} \leftarrow$ classifySort(λ)
9 $\zeta \leftarrow \zeta - \zeta^{\text{OL}} - \zeta^{\text{HS}}$
10 **foreach** $\lambda_c \in \lambda^{CLUSTER}$ **do**
11 $\Delta_T \leftarrow \{\}$
12 **while** λ_c *is not empty* **do**
13 $j \leftarrow$ first(λ_c)
14 **while** j *is not empty* **do**
15 $k \leftarrow$ first(j)
16 **while** ζ *is not empty* **do**
17 $i \leftarrow$ next(ζ)
18 $\Delta_T[i] \leftarrow$ CalcTempRise(k, i)
19 $i^* \leftarrow$ argmin(Δ_T)
20 assign(k, i^*)
21 remove k from j
22 remove j from λ_c
23 **if** $\pi_{i^*} >= \lambda_{i^*}^{Core}$ **then**
24 remove i^* from ζ

The algorithm begins by adding the number of jobs currently running on all the servers to the newly arrived jobs to calculate the total workload to be run by the system (Line 1). Then, it calculates the payoff (workload proportion) for each computing element $\zeta_i \in \zeta$ using the **Core** solution concept (Section IV-B for details) (Line 2). The workload proportion recommended by the **Core** is denoted by λ^{Core}. The algorithm then identifies all the servers exceeding their thermal constraints, denoted by ζ^{HS} at Line 3.

Subsequently, it gets suggested migrations from the hotspots, aimed at keeping them below their predefined temperature thresholds. The function GetTasksToMigrate chooses the CPU-intensive jobs running on the selected computing node for migration until the expected temperature of the computing node is below the threshold temperature value. These suggested task migrations are added to the set of migrations to be performed denoted as λ^{MIG}.

Similarly, the algorithm identifies all overloaded servers based on the proportions recommended by the **Core** (Section IV-B) at Line 5. If the number of user jobs currently running

on the selected server exceeds the suggested proportion, signifying overload, the algorithm obtains suggested migrations by calling the GetCoolestTasksToMigrate, which are again added to the set of migrations to be performed λ^{MIG} (see Line 6). It is important to note that the function GetCoolestTasksToMigrate chooses $\pi_i - \lambda_i^{\text{Core}}$ number of coolest tasks running on the selected server for migration so that the number of tasks running is less than or equal to the Core's recommended value λ_i^{Core}. λ^{MIG} is then added to the total number of jobs to be scheduled λ at Line 7.

To classify user jobs based on their TP, the algorithm invokes classifySort function at Line 8. It categorizes the λ into an array of 5 clusters (from very_hot to very_cold) denoted by λ^{CLUSTER} using the K-means clustering approach. Furthermore, the classifySort function sorts the jobs within each cluster $\lambda_c \in \lambda^{\text{CLUSTER}}$ based on their duration and TP using Eq. (23) (Section IV-A). At Line 9, it removes hotspots and overloaded servers from the set of available servers, as these servers cannot accommodate additional jobs.

Next, the algorithm iterates through each cluster in λ, prioritized based on their thermal profiles, with the aim of optimizing thermal outcomes for all feasible assignments (refer to Line 10 to 24). It initializes an empty map called 'utilities' to store the corresponding utility values for various possible assignments. The algorithm iterates through the selected user jobs until the subset is empty (see Line 12). It selects the first job j in the subset (Line 13) and assigns all the tasks $k \in j$ until no more tasks are left. To do so, it iterates through all available servers in ζ (Line 16), calculating the temperature rise for possible assignments of the k to each server i and storing the value in the Δ_T map against the selected server i (see Line 18). After computing temperature rise for all possible allocations against the selected task k, the algorithm identifies the server i^* that results in optimal thermal consequences in terms of temperature rise, assigns the task to the server i^*, and removes the selected task k from j (see Line 19 to 21).

The steps from 12 to 21 are related to optimal scheduling of tasks. This is done by traversing each task within every job arrived at the system, and the available servers. At the end of traversal, the function basically creates an allocation map that stores the thermal consequences for the selected task against each computing node. The algorithm then chooses the optimal allocation (that is the server with optimal thermal outcomes denoted by i^*) based on the thermal readings in the allocation map. These steps help identifying the impact of each task on each server and its surrounding servers within the system, then find the server with best thermal consequences among all the servers leading to optimal allocation of user jobs to the available servers. Lastly, after assigning all the tasks in selected job j to server(s), the selected job j is removed from the job cluster λ_c. Finally, if the assigned workload exceeds or equals the workload recommended by the Core, the server i^* is removed from the set of available servers ζ. This process is repeated until all the jobs in each cluster are allocated.

The complexity of the CGTAS can be divided into two main parts. First, the calculation of the exact Core for the proposed CG. Calculating the exact Core of a CG is NP-hard. However, approximating the Core is a well-studied problem – there is a plethora of approaches such as [16, 43, 44] available in the literature to efficiently calculate the Core. Moreover, it is also important to note that the complexity of calculating the core depends on the number of computing nodes and not the number of user jobs. Second, the scheduling of user jobs based on the Core value. The complexity of the second part is $J \times K \times N$, where J, K, and N denote the number of jobs, tasks in each job, and computing nodes. Hence, the computational complexity of the proposed algorithm mainly depends on the complexity of the Core solution. However, in this paper, we follow the approximation approach proposed in [16] due to the availability of source code. Therefore, the proposed approach can be easily scaled for large-scale hierarchically organized DCs with thousands of servers and millions of task requests.

TABLE IV: Server configurations

	Enclosure (CPU) (Ghz) (Cores/Chips)
S1	Asus RS720-E9/RS8 (Xeon Platinum 8180)(2.5)(56/2)
S2	Acer AR580 F2 (Xeon E5-4607)(2.2)(24/4)
S3	Dell PowerEdge R7425 (AMD EPYC 7601)(2.2)(63/2)
S4	ThinkSystem SR650 (Xeon Platinum 8176)(2.1)(56/2)
S5	HP ProLiant DL110 (Xeon Gold 6314U)(2.3)(32/1)

TABLE V: The power usage of different configurations

	0%	**10%**	**20%**	**30%**	**50%**	**70%**	**90%**	**100%**
S1	49	106	129	154	205	269	347	385
S2	109	155	170	184	211	252	324	368
S3	77	128	144	157	184	218	253	268
S4	58	119	139	159	197	242	314	357
S5	94	104	116	130	156	197	287	307

TABLE VI: The organization of servers in each setting

Racks	**Servers**	**S1**	**S2**	**S3**	**S4**	**S5**
15	300	75	25	75	100	25
20	350	75	125	25	75	50
20	400	50	75	125	25	125
25	450	75	50	150	100	75
25	500	100	100	100	100	100

V. Simulation Results and Discussion

To empirically validate the efficacy of our proposed CGTAS design, we conduct a comprehensive comparative analysis using simulations, contrasting its performance against several TA designs (TASA [30], GTARA [8], and SW [9]). The TASA prioritizes user jobs and computing units based on their TPs; hottest jobs are assigned to coolest nodes first. However, there is no mitigation strategy when servers reach their vendor-specified TCs. In addition to performing migrations when a server reaches its threshold value, the GTARA also considers

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2024.3414671

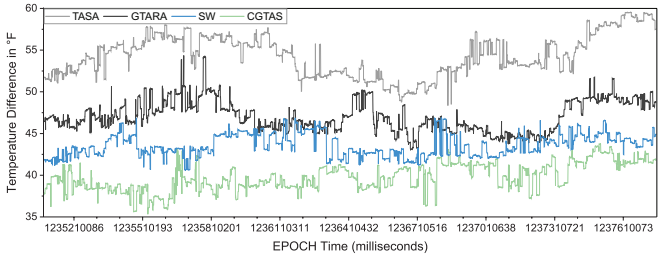

Fig. 8: Comparison of DC thermal uniformity achieved by each strategy over time

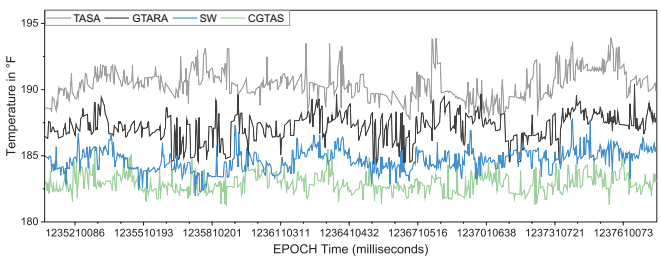

Fig. 9: Comparison of servers' maximum temperature due to each strategy over time

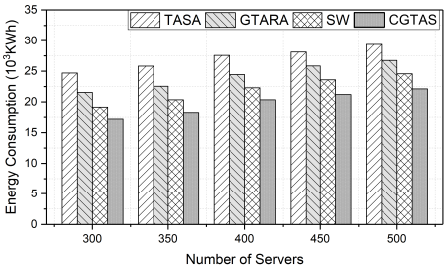

Fig. 10: Comparing total energy consumed by each scheduling design

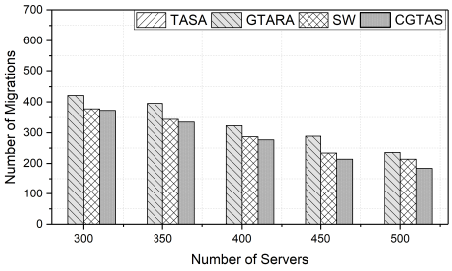

Fig. 11: Comaring the number of migrations against each approach

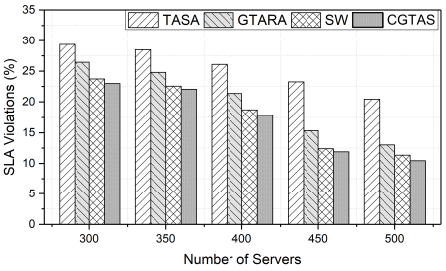

Fig. 12: Comparison of SLA violation rate due to each scheduling strategy

ambient temperature of surrounding nodes during the task scheduling process. Moreover, the GTARA uses cooperative game with Nash equilibrium solution concept to attain optimal thermal conditions. The SW strategy uses the Shapley value from CGT to avoid thermal imbalance and hotspots by assigning user jobs to computing nodes based on their marginal thermal performance inside DC. Similar to the GTARA, the SW also considers task migrations to avoid hotspots.

Our evaluation includes the following critical metrics: thermal uniformity attained over time, maximum server temperature recorded over time, energy consumption, task migrations, SLA violations, number of hotspots, maximum supplied temperature, and minimum inlet temperature. We conduct simulations using a real DC workload trace from the "Center of Computational Research (CCR), State University of New York at Buffalo". The dataset encompasses a substantial corpus of 22,385 user jobs observed over a 30-day period. Finally, we consider five server configurations in terms of computing capacity and power usage as depicted in Table IV and Table V, respectively. These settings have been adopted from benchmark results [45]. Since 2007, the benchmark results are quarterly updated every year. Lastly, servers are arranged according to settings listed in Table VI.

To compare the efficacy of CGTAS in attaining thermal efficiency, we compare thermal uniformity (TU) and the maximum server temperature reached over time against the TASA, GTARA, and SW. We calculate TU as "the temperature difference between the hottest and the coolest pod at any particular instance of time". Fig. 8 shows the TU achieved over time. The proposed CGTAS technique attains better TU as opposed to its TA counterparts due to the fact that it adapts the workload on computing infrastructure at runtime based on the dynamic thermal conditions of the DC environment. During the entire simulation, the highest value recorded for TU is 43.87, 47.14, 54.11, and 59.78 for the CGTAS, SW, GTARA, and TASA, respectively. Similarly, the lowest TU values recorded for the CGTAS, SW, GTARA, and TASA are 35.37, 40.57, 42.69, 47.86, respectively.

Fig. 9 shows the maximum server temperature values recorded over time. As depicted in Fig. 9, the proposed CGTAS technique is effective in lowering the server temperature readings due to the fact that it dynamically adapts the workload running on the computing nodes using the Core solution. Although the GTARA and the SW perform better than the TASA, they result in significantly higher server temperature values compared to the CGTAS. The highest value recorded for the maximum server temperature is 193.78, 190.32, 187.81, and 185.63 for the TASA, GTARA, SW, and CGTAS strategies, respectively. Similarly, the lowest values recorded for the TASA, GTARA, SW, and CGTAS are 187.91, 183.98, 182.12, and 181.24, respectively. In short, the proposed TA strategy attains lower server temperature values, and at the same time, effectively manages to minimize the thermal-imbalance leading to lower cooling cost.

Fig. 10 compares the CGTAS with its TA counterparts in terms of energy efficiency (total energy consumed). The total energy used is calculated as the sum of energy consumed by the computing nodes and the cooling system. It can be observed that the CGTAS techniques results in lower energy consumption compared to its TA substitutes for all the settings. Specifically, the proposed TA strategy achieves 26.08%, 17.20% and 8.91% average energy savings compared to the TASA, GTARA, and SW, respectively.

Fig. 11 depicts the total number of task migrations (the sum of both interpod and intrapod migrations) against each scheduling strategy. In our experiments, we assume that 50%

This article has been accepted for publication in IEEE Transactions on Green Communications and Networking. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TGCN.2024.3414671

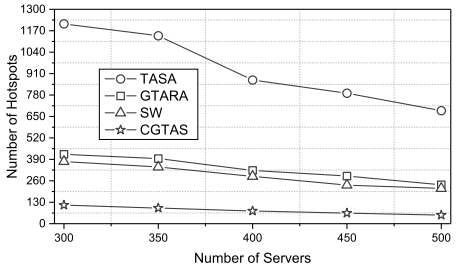

Fig. 13: Comparison in terms of number of Hotspots arised

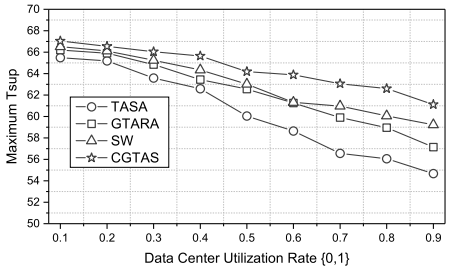

Fig. 14: Maximum supplied temperature from the CRAC

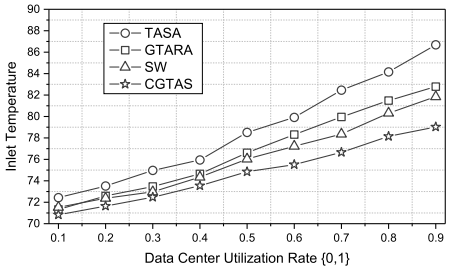

Fig. 15: Comparing minimum inlet temperature

of the CPU is utilized by the migration process. Fig. 11 does not show any migrations for the TASA strategy – the TASA allocates incoming user jobs in thermal-aware fashion and does not perform any migrations to mitigate hotspots. It can be observed that the proposed strategy performs the least number of migrations compared to any other strategy; however, the difference is not very significant when compared to the SW strategy. In other words, the proposed approach attains better thermal balance and lower server temperature values due to the fact that the CGTAS is adaptive in nature, and it adjusts the assigned load to each overloaded computing node at runtime using task migrations. Overall, energy savings attained by the CGTAS significantly outweighs the migration overhead.

Fig. 12 compares the SLA violation rate due to the TASA, GTARA, SW, and CGTAS. SLA can be defined as "the least amount of resources to be allocated to finish a job before its deadline." Fig. 12 illustrates that the CGTAS design results in lower SLA violation rate compared to the TASA, GTARA, and SW due to the fact that it effectively avoids the creation of hotspots. Also, we consider different settings by varying the number of servers from 300 to 500 nodes. In all these settings, the CGTAS strategy outperforms its counterparts.

Fig. 13 compares the number of hotspots arised due to each strategy. We consider a computing node as a hotspot when it reaches or exceeds its vendor-specified threshold temperature. In this context, the TASA approach performs worst and results in highest number of hotspots compared to any other strategy. The number of hotspots due to the GTARA is slightly higher than that of the SW strategy; however, the difference is very negligible. Finally, the proposed CGTAS performs best and effectively reduces the number of hotspots due to adaptively adjusting the workload based on the relative thermal performance of computing nodes using the Core solution from CGT.

Fig. 14 compares the maximum supplied temperature T_{SUP} from the cooling system for each scheduling strategy. It can be observed that the T_{SUP} for the TASA strategy is very low leading to extremely high energy costs. The supplied temperature for the GTARA and the SW strategy are also very low compared to the CGTAS; however, the SW strategy performs slightly better than the GTARA. Moreover, it can be observed that the difference between the supplied temperature due to CGTAS and the rest of the strategies increases with the increase in number of running servers. Overall, the proposed CGTAS design performs better than any other strategy and leads to higher T_{SUP} value for all server configurations.

Fig. 15 shows the minimum inlet temperature recorded against each scheduling strategy (TASA, GTARA, SW, and CGTAS). We compare the inlet temperature with respect to the utilization rate (between 0 and 1) of the DC where 1 indicates 100% utilization and 0 indicates 0% utilization of the whole capacity. For lower utilization rates, the difference between the inlet temperature readings for all the strategies is not very significant. However, as the utilization rate of the DC increases, the inlet temperature difference increases gradually, reaching the highest values at 90% utilization rate. The CGTAS results performs better than its counterparts and effectively manages to keep the inlet temperature higher.

In summary, the proposed coalitional game-based TA task allocation outperforms its TA substitutes. It attains better temperature values and energy savings compared to the TASA, GTARA and SW.

VI. Conclusion

This work introduces a Coalitional Game-based Ghermal-aware Adaptive Scheduler (CGTAS). The CGTAS considers both the CPU time of user jobs and their thermal profiles. It intelligently allocates lengthy CPU-intensive task to a server offering optimal thermal performance. Additionally, using the Core solution concept from CGT, the proposed TA design dynamically adapts the proportion of workload across computing nodes based on their relative temperatures in real-time. Also, it identifies the overloaded computing nodes using the Core and considers tasks migrations to lower server temperatures, avoid hotspots, and optimize thermal consequences. In other words, the CGTAS reduces the thermal difference among the computing nodes and at the same time, it effectively manages to keep the servers temperatures lower.

The CGTAS employs K-means clustering approach to optimize thermal consequences of allocating long CPU-intensive tasks. After creating clusters based on the thermal profiles of the incoming user jobs, the proposed approach sorts the user jobs in each cluster based on their duration and thermal consequences. By considering the duration of incoming user jobs during scheduling and dynamically redistributing the tasks across computing nodes based on their real-time thermal profiles using the Core solution from CGT, the CGTAS consistently demonstrates superior effectiveness in achieving optimal thermal performance compared to existing TA methods.

Extensive simulations using a real-world dataset as input to a TA simulator show that our proposed scheduling strategy

outperforms its TA counterparts. The results indicate an average energy savings of 26.08%, 17.20%, and 8.91% compared to the TASA, GTARA, and SW, respectively. In conclusion, the proposed coalitional game-based strategy enhances thermal efficiency and reduces energy costs for DC operations, promoting sustainable and high-performance computing infrastructure.

REFERENCES

[1] Jianpeng Lin et al. "Thermal Modeling and Thermal-aware Energy Saving Methods for Cloud Data Centers: A Review". In: *IEEE Transactions on Sustainable Computing* (2023), pp. 1–20.

[2] Peng Xiao et al. "A power and thermal-aware virtual machine management framework based on machine learning". In: *Cluster Computing* 24.3 (2021), pp. 2231–2248.

[3] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. "Data Center Energy Consumption Modeling: A Survey". In: *IEEE Communications Surveys & Tutorials* 18.1 (2016), pp. 732–794.

[4] Leila Ismail and Huned Materwala. "Computing Server Power Modeling in a Data Center: Survey, Taxonomy, and Performance Evaluation". In: *ACM Computing Surveys (CSUR)* 53.3 (2020), pp. 1–34.

[5] Prateek Verma, Ashish Kumar Maurya, and Rama Shankar Yadav. "A survey on energy-efficient workflow scheduling algorithms in cloud computing". In: *Software: Practice and Experience* 54.5 (2023), pp. 637–682.

[6] Jie Li et al. "Towards Energy-Efficient and Thermal-Aware Data Placement for Storage Clusters". In: *IEEE Transactions on Sustainable Computing* (2024), pp. 1–17.

[7] Luca Parolini et al. "A cyber–physical systems approach to data center modeling and control for energy efficiency". In: *Proceedings of the IEEE* 100.1 (2012), pp. 254–268.

[8] Saeed Akbar et al. "A Game-based Thermal-Aware Resource Allocation Strategy for Data Centers". In: *IEEE Transactions on Cloud Computing* 9.3 (2021), pp. 845–853.

[9] Saeed Akbar and Ruixuan Li. "A Shapley value-based thermal-efficient workload distribution in heterogeneous data centers". In: *The Journal of Supercomputing* 78.1 (2022), pp. 14419–14447.

[10] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[11] Kevin Leyton-Brown and Yoav Shoham. "Essentials of game theory: A concise multidisciplinary introduction". In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 2.1 (2008), pp. 1–88.

[12] Jie Li et al. "Towards Thermal-Aware Workload Distribution in Cloud Data Centers Based on Failure Models". In: *IEEE Transactions on Computers* 72.2 (2023), pp. 586–599.

[13] Farnaz Niknia, Vesal Hakami, and Kiamehr Rezaee. "An SMDP-based approach to thermal-aware task scheduling in NoC-based MPSoC platforms". In: *Journal of Parallel and Distributed Computing* 165.1 (2022), pp. 79–106.

[14] Baver Ozceylan et al. "Minimizing the Maximum Processor Temperature by Temperature-Aware Scheduling of Real-Time Tasks". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 30.8 (2022), pp. 1084–1097.

[15] Jie Li et al. "TADRP: Towards Thermal-Aware Data Replica Placement in Data-Intensive Data Centers". In: *IEEE Transactions on Network and Service Management* 20.4 (2023), pp. 4397–4415.

[16] Tom Yan and Ariel D Procaccia. "If you like shapley then you'll love the core". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 6. 2021, pp. 5751–5759.

[17] Muhammad Tayyab Chaudhry et al. "Thermal-aware scheduling in green data centers". In: *ACM Computing Surveys (CSUR)* 47.3 (2015), pp. 1–48.

[18] T Van Damme, C De Persis, and P Tesi. "Optimized Thermal-Aware Job Scheduling and Control of Data Centers". In: *IEEE Transactions on Control Systems Technology* 27.2 (2019), pp. 760–771. ISSN: 1063-6536.

[19] Huiwen Cheng et al. "A survey of energy-saving technologies in cloud data centers". In: *The Journal of Supercomputing* 77.11 (2021), pp. 13385–13420.

[20] José Moura and David Hutchison. "Game theory for multi-access edge computing: Survey, use cases, and future trends". In: *IEEE Communications Surveys & Tutorials* 21.1 (2018), pp. 260–288.

[21] Long Zhang et al. "Anti-Jamming Colonel Blotto Game for Underwater Acoustic Backscatter Communication". In: *IEEE Transactions on Vehicular Technology* (2024).

[22] Jiachen Yang et al. "A task scheduling algorithm considering game theory designed for energy management in cloud computing". In: *Future Generation computer systems* 105 (2020), pp. 985–992.

[23] Zheng Xiao et al. "Learning non-cooperative game for load balancing under self-interested distributed environment". In: *Applied Soft Computing* 52 (2017), pp. 376–386.

[24] Long Zhang, Yao Wang, and Zhu Han. "Safeguarding UAV-enabled wireless power transfer against aerial eavesdropper: a Colonel Blotto game". In: *IEEE Wireless Communications Letters* 11.3 (2021), pp. 503–507.

[25] Wen-Xiao Chu and Chi-Chuan Wang. "A review on airflow management in data centers". In: *Applied Energy* 240 (2019), pp. 84–119.

[26] Hongjie Lu, Zhongbin Zhang, and Liu Yang. "A review on airflow distribution and management in data center". In: *Energy and Buildings* 179.1 (2018), pp. 264–277.

[27] Sukhpal Singh Gill et al. "ThermoSim: Deep learning based framework for modeling and simulation of thermal-aware resource management for cloud computing environments". In: *Journal of Systems and Software* 166 (2020), p. 110596.

[28] Qingxia Zhang et al. "A survey on data center cooling systems: Technology, power consumption modeling and control strategy optimization". In: *Journal of Systems Architecture* 119.1 (2021), p. 102253.

[29] Assel Sakanova et al. "Multi-objective layout optimization of a generic hybrid-cooled data centre blade server". In: *Applied Thermal Engineering* 156.1 (2019), pp. 514–523.

[30] Lizhe Wang, Samee U. Khan, and Jai Dayal. "Thermal aware workload placement with task-temperature profiles in a data center". In: *Journal of Supercomputing* 61.3 (2012), pp. 780–803. ISSN: 09208542.

[31] Saif U.R. Malik et al. "Modeling and analysis of the thermal properties exhibited by cyberphysical data centers". In: *IEEE Systems Journal* 11.1 (2017), pp. 163–172. ISSN: 19379234.

[32] Mark A Oxley et al. "Rate-based thermal, power, and co-location aware resource management for heterogeneous data centers". In: *Journal of Parallel and Distributed Computing* 112.1 (2018), pp. 126–139.

[33] SeyedMorteza MirhoseiniNejad et al. "Joint data center cooling and workload management: A thermal-aware approach". In: *Future Generation Computer Systems* 104 (2020), pp. 174–186.

[34] Hao Feng, Yuhui Deng, and Jie Li. "A global-energy-aware virtual machine placement strategy for cloud data centers". In: *Journal of Systems Architecture* 116.1 (2021), p. 102048.

[35] Saeed Akbar et al. "Server temperature prediction using deep neural networks to assist thermal-aware scheduling". In: *Sustainable Computing: Informatics and Systems* 36.1 (2022), p. 100809. ISSN: 2210-5379.

[36] Ahsan Ali and Öznur Özkasap. "Spatial and thermal aware methods for efficient workload management in distributed data centers". In: *Future Generation Computer Systems* 153 (2024), pp. 360–374. ISSN: 0167-739X.

[37] Rohan Basu Roy, Tirthak Patel, and Devesh Tiwari. "Satori: efficient and fair resource partitioning by sacrificing short-term benefits for long-term gains". In: *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE. 2021, pp. 292–305.

[38] Manoj Kumar et al. "Thermal aware learning based CPU governor". In: *Concurrency and Computation: Practice and Experience* 34.11 (2022), e6862.

[39] SeyedMorteza Mirhoseininejad, Ghada Badawy, and Douglas G Down. "A data-driven, multi-setpoint model predictive thermal control system for data centers". In: *Journal of Network and Systems Management* 29.1 (2021), pp. 1–22.

[40] Patricia Arroba et al. "Heuristics and metaheuristics for dynamic management of computing and cooling energy in cloud data centers". In: *Software: Practice and Experience* 48.10 (2018), pp. 1775–1804.

[41] Somayye Rostami, Douglas G Down, and George Karakostas. "Linearized Data Center Workload and Cooling Management". In: *IEEE Transactions on Automation Science and Engineering* (2024), pp. 1–13.

[42] Weiwei Lin et al. "A taxonomy and survey of power models and power modeling for cloud servers". In: *ACM Computing Surveys (CSUR)* 53.5 (2020), pp. 1–41.

[43] Vincent Conitzer and Tuomas Sandholm. "Complexity of constructing solutions in the core based on synergies among coalitions". In: *Artificial Intelligence* 170.6-7 (2006), pp. 607–619.

[44] Andreas S Schulz and Nelson A Uhan. "Approximating the least core value and least core of cooperative games with supermodular costs". In: *Discrete Optimization* 10.2 (2013), pp. 163–180.

[45] Standard Performance Evaluation Corporation. *SPECpower_ssj 2008*. https://www.spec.org/power_ssj2008/results/. Accessed: 05-11-2023.

Saeed Akbar received his PhD from Huazhong University of Science and Technology, Wuhan, China. Prior to this, he served as a lecturer at the Department of Computer Science, Iqra National University, Pakistan. Currently, he is engaged in postdoctoral research at the School of Computer Science and Technology, Zhejiang Normal University, China. His research interests include Cloud Computing, Sustainable and High Performance Computing, Algorithmic Game Theory, Deep Learning, and Computer Vision.

Ubaid Ul Akbar has recently completed his Bachelor's degree in Computer Science from the Department of Computer Science, the City University of Science and Information Technology, Pakistan. His research interests include Object-Oriented Software Engineering, Visual (Block-based) Programming Languages, Software Project Scheduling, Resource Scheduling in Cloud and High-Performance Computing.

Rahmat Ullah is a Lecturer at the School of Computer Science and Electronic Engineering, University of Essex, United Kingdom. He previously worked as a Research & Development Associate with the Faculty of Computing, Engineering and Science, University of South Wales, U.K. Rahmat received his Master's in Software Engineering from COMSATS University, Islamabad, Pakistan, and his PhD from the School of Engineering, University of Edinburgh, U.K. His research interests include biomedical signal processing, microwave imaging, software engineering, machine learning, and digital health.

Zhonglong Zheng (Member, IEEE) received the B.S. degree in electronic engineering from the China University of Petroleum, in 1999, and the Ph.D. degree from the Department of Automation, Shanghai Jiao Tong University, in 2005. He is currently a Full Professor with the College of Computer Science and Technology, Zhejiang Normal University, Jinhua, China. His research interests include machine learning, data mining, and blockchain.