Thermal-aware Resource Management in Energy Efficient Clusters

by

Shubbhi Taneja

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 4, 2018

Keywords: Energy-efficiency, Thermal-aware, Scheduling, Approximate Computing, MapReduce, Modeling

Approved by

Xiao Qin, Chair, Professor of Computer Science and Software Engineering
Wei-Shinn Ku, Associate Professor of Computer Science and Software Engineering
Cheryl Seals, Associate Professor of Computer Science and Software Engineering
David Umphress, Professor, Computer Science and Software Engineering

Abstract

There is a pressing need for thermal management in most electronic devices today, ranging from portables to high-performance servers, and it poses barriers to the safe operations of data centers. The goal of thermal management is to reduce thermal hotspots and non-uniform on-chip temperatures that may impact the longevity of hardware. In this dissertation research, we develop a thermal-aware job scheduling strategy called *tDispatch* tailored for MapReduce applications running on Hadoop clusters. The scheduling idea of *tDispatch* is motivated by a profiling study of CPU-intensive and I/O-intensive jobs from the perspective of thermal efficiency. We show that CPU-intensive and I/O-intensive jobs exhibit various thermal and performance impacts on multicore processors and hard drives of Hadoop cluster nodes. After we quantify the thermal behaviors of Hadoop jobs on the master and data nodes of a cluster, we propose our scheduler that performs job-to-node mappings for CPU-intensive and I/O-intensive jobs. We apply our strategy to several MapReduce applications with different resource consumption profiles. Our experimental results show that tDispatch is conducive to creating opportunities to cool down multicore processors and disks in Hadoop clusters deployed in modern data centers. Our findings can be applied in other thermal-efficient job schedulers that are aware of thermal behaviors of CPU-intensive and I/O-intensive applications submitted to Hadoop clusters.

Characterizing thermal profiles of cluster nodes is an integral part of any approach that addresses thermal emergencies in a data center. As the power density of today's data centers grows, preventing thermal emergencies in data centers becomes one of the vital issues. Most existing thermal models make use of CPU utilization to estimate power consumption, which in turn facilitates outlet-temperature predictions. Such utilization-based thermal models may introduce errors in predicting power usage due to inaccurate mappings from system utilization to outlet temperatures. To address this concern in the existing models, we eliminate utilization models as a middleman from the thermal model. In this dissertation, we propose a thermal model, *tModel*, that projects outlet temperatures from inlet temperatures as well as directly

measured multicore temperatures rather than deploying a utilization model. The proposed thermal model estimates the outlet air temperature of the nodes for predicting cooling costs for cluster nodes. We validate the accuracy of our model against data gathered by thermal sensors in our cluster. Our results demonstrate that *tModel* estimates outlet temperatures of the cluster nodes with much higher accuracy over CPU-utilization based models. We further show that *tModel* is conducive to estimating the cooling cost of data centers using the predicted outlet temperatures.

High energy efficiency of applications helps in reducing the operational costs of data centers. For a wide range of applications, the overall computational cost can be significantly reduced if an exact solution is not required. Approximate computing is one such paradigm leveraging the forgiving nature of many applications to improve the energy efficiency of cluster nodes. We propose a framework called *tHadoop2* for MapReduce applications running on Hadoop clusters. To facilitate the development of *tHadoop2*, we incorporated an existing thermal-aware workload placement module called *tHadoop* into our *tHadoop2*. Our framework consists of three key components - *tHadoop*, a *thermal monitoring and profiling* module, *approximation-aware thermal manager*. We investigated the thermal behavior of a MapReduce application called Pi running on Hadoop clusters by varying the two input parameters - the number of maps and the number of sampling points per map. Our profiling results show that Pi exhibits inherent resilience in terms of the number of precision digits present in its value. It is noteworthy that this result quality varies with the application type. Other MapReduce applications can be scrutinized by exploring their characteristics and finding opportunities for acceptable inexactness in outputs. Nevertheless, the proposed framework, coupled with approaches for making tradeoffs, is generally applicable to any MapReduce application.

Acknowledgments

I am immensely grateful to many people for their contributions to this dissertation and for the time I spent in the graduate school. This dissertation would not have been completed without their invaluable guidance, support, and encouragement.

First, it has been an honor to work with Dr. Xiao Qin, my adviser. I am immensely grateful to him for his immaculate mentoring over the last four years. He has always provided honest and supportive advice in my research. He went above and beyond in writing strong letter of recommendations for me and because of that, I received many fellowships and travel awards for attending conferences. It has been a great privilege to have worked under his supervision.

I am truly blessed to be advised by my committee members, Dr. Cheryl Seals, Dr. David Umphress, and Dr. Wei-Shinn Ku, for spending their valuable time in reviewing my proposal and dissertation documents. Their timely feedback on my work and comments have been very beneficial in improving the quality of this dissertation. I would like to also thank Dr. Spencer Millican for serving on my reading committee and sharing his immense knowledge and interest in the field of real-time scheduling. Thanks to the efforts of Ms. Kelly Price in guiding me through the process of installation and set-up of my experimental testbed and sharing her system-level knowledge with me.

On a more personal level, I want to thank Dr. James Groccia, Dr. Jeffrey Smith, and Dr. Anthony Skjellum for being my mentors in teaching and scholarship. It has been rewarding and inspirational to have learned about teaching and research techniques from these passionate educators. They have been generous in providing academic and career guidance.

This dissertation would not have been completed without the counsel of my co-authors and collaborators. It has been rewarding and inspirational to have worked with a passionate group of researchers in my research lab - Yi Zhou, Yuanqi Chen, Ajit Chavan, Xiaopu Peng, Chaowei Zhang, and Xunfei Jiang. I thank each of them for making graduate school life productive and enjoyable for me. I would like to give special thanks to each one of my Auburn family

members - Amrit Abrol, Mohit Arora, Rahul Kumar, Neha Arya, Harshit Goyal, Tanuj Puri, Sarthak Kakkar, Gautam Dudeja, Rodrigo Sardinas, and Neda Topuz - for helping me with making big decisions when my parents were not around and supporting me through thick and thin.

Finally, I am also grateful to all my entire family back in India for the patience and support they provided me in the last five years. My role model and grandmother, Prakash Rani, has taught me how to face challenges and endurance in life. It is solely because of her support was I able to come to the U.S. and pursue a degree in the field of my interest. My mother, Veena Taneja, is my best friend and has always shown immense love and support in my choices. I thank my father, R.K. Taneja, who lived his dream of pursuing a technical degree through me and helping me become the first engineer in my entire family. I owe a special debt to my sister, Kritika Taneja, for offering constructive criticism on my work and performance in graduate school that kept me going. I want to bless her as she embarks on her graduate studies away from home in a few days. I cannot express enough gratitude to my best friend and soon-to-be life partner, Amrit Abrol, for enriching my life with his love and support and making every day of my life enjoyable.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivations

Data centers house a large number of computer systems and components such as power supplies, network cables, thermostats, security devices, and many more. Consequently, these components generate substantial amounts of heat and consume enormous amounts of energy for cooling purposes. According to National Renewable Energy Laboratory (NREL), a data center consumes 10-100 times more energy than a regular office [1]. Growing number of data centers are being deployed across the world in last decade. According to the EPA report, US data centers consumed approximately 2% of total U.S. energy consumption, with 24% increase in the last five years [2]. This continuous increase in energy consumption by the data centers is an effect of rapidly growing demand of computation and storage capacity. This demand has also caused an increase in the power density of data centers. As the power density of data centers grows, managing thermal emergencies in data centers becomes one of the vital issues. Gartner predicts that by 2021, more than 90% of the data centers would have to revise their thermal management strategies [3]. Thus, it is essential that we find solutions to enhance the thermal- and energy-efficiency of these data centers.

Increasing energy-efficiency in cluster systems installed in data centers can help in reducing the energy consumption, excess heat, lower cooling costs and improve reliability of server nodes in these data centers. Since energy efficiency is directly linked with power consumption, several power management techniques have been used in the recent past to deal with excessive heat buildup and cooling requirements in data centers. Two factors contributing to the power

usage in data centers are new generation technologies and an inclination towards multicore architectures. To increase a processor's speed, manufacturers are increasing the number of cores on each CPU chip. With a large number of cores in computing clusters, power consumption and heat dissipation have become a big concern for data centers. Also, the next generation architectures demand high power usage, which in turn translates into heat from processors. At the same time, evidence shows that the thermal-aware architecture techniques have a direct correlation with power-aware techniques [4]. While thermal and energy efficiency remain the most important concerns in the operation of data centers, attention has also been paid to optimizing (1) operational costs and (2) system reliability of these clusters [5].

With a dramatic increase in energy consumption of large-scale cluster computing systems, we must urgently address the energy efficiency issues. Strong evidence indicates that the cooling cost is a significant contributor of a cluster's operational cost [6] [7]. Under certain circumstances, upto half of the total energy is attributed to cooling the data centers [8]. Majority of the cooling energy is used towards keeping hardware operating temperatures below threshold (a.k.a, the redline) temperatures specified by the manufacturer of the hardware [8]. Variations in temperature accounts for more than 50% of electronic failures [9] and when a data center's temperature is over 21°C, every 10°C rise in temperature reduces hardware reliability by 50% [10]. Therefore optimal temperatures look promising in improving the lifetime of hardware deployed in data centers. Due to the rising power consumption levels and the need for enhancing the energy efficiency of data centers, the need for data center cooling is only going to grow in future. Prior studies confirmed that cutting cooling cost effectively improves the energy efficiency of data centers [11] [12]. In one such study, intriguing workload placement strategies were implemented for balancing the temperature distribution in data centers [11] [12]. Since temperature is in direct co-relation with power output over time therefore, cutting down cooling costs is of great signficance in designing data centers [4].

One way to monitor the cluster's thermal behavior is to set up temperature sensors to monitor the incoming (a.k.a, inlet) and outgoing (or outlet) air temperatures of cluster nodes in data centers. Although deploying sensors is feasible for small clusters, it becomes an expensive and tedious solution for large data centers housing thousands of server nodes. Further, sensors

are also subjected to failures, which may lead to invalid or faulty data collection. Therefore, it is important to use inexpensive models to estimate the inlet and outlet air temperature of servers in cluster systems. A thermal model aims to predict internal temperatures of different parts of data centers [13]. Thermal modeling is helpful especially when working with large data centers, because installing sensors to individual nodes would be tedious. Thus, thermal models are essential in predicting an overall thermal response of data centers. Thermal profiling lays a solid foundation for thermal models. This observation motivates us to conduct a thermal profiling study in the context of Hadoop jobs running on cluster systems. The profiling results are envisioned as thermal-models that serve as building blocks for the next-generation data centers.

Operating cluster nodes at high temperatures can lead to component failures in cluster nodes. Thus, there is a pressing need to continuously monitor the thermal behavior of data center to anticipate and proactively deal with thermal emergencies. Avoiding thermal emergencies is very important in order to achieve high availability and avoid hardware failures. Number of thermal models have been proposed in the past that calculate inlet temperatures of servers and their impact on thermal profiles and cooling cost of servers in data centers. Most of these thermal models rely on processor utilizations to predict outlet air temperatures of cluster nodes. In this dissertation, we specifically investigate the impact of MapReduce workloads on the outlet temperature of the nodes. We propose a thermal model to predict outlet air temperatures of cluster nodes arranged in a traditional rack. With our thermal model in place, we estimate the cooling costs for various Hadoop applications.

Modern day companies such as Amazon, Facebook, and Google regularly gather unprecedented amounts of data, often referred to as Big Data. As these companies expand their data centers, Hadoop increasingly becomes a popular big-data processing framework running in a distributed fashion on large clusters of commodity hardware. Hadoop accomplishes two goals, namely, massive data storage and faster processing. Hadoop works well with both structured and unstructured data. Many Hadoop clusters consist of enormous data sets that cannot be digitally acquired. Hadoop supports MapReduce programs that spread over clusters of computers to expedite the processing of big data. This parallel computing ability, moving computation to

data rather than moving data to computation, gives cost-effective Hadoop clusters an edge over other existing platforms. The popularity of Hadoop motivates us to develop a thermal-aware scheduler to improve the energy efficiency of Hadoop clusters.

The growth of Big Data and massive data sets has presented challenges in the cost and density of hardware technologies, available power for processing it and the related energy footprint. There is an array of application domains, including Big Data analytics, that can tolerate certain loss of fidelity when resources required to provide a precise answer are either expensive or unavailable [14]. Studies repeatedly show that applications consist of both critical and non-critical components [15][16]. Approximate computing works by identifying the non-critical components in applications and by compromising the accuracy in the results to a certain extent, approximate computing lays foundation for performance gains and energy savings in cluster systems. Approximation strategies are required to be determined on a per-application basis due to varying nature of applications. By applying a series of approximation heuristics, applications can trade off accuracy for more crucial resources like execution time and energy [17]. In this research work, we propose a framework running thermal-aware and approximation-enabled MapReduce applications on Hadoop cluster. MapReduce has been a popular paradigm for analyzing huge volumes of complex data on cluster systems. Thus, embedding approximation strategies in MapReduce framework can enhance the energy-efficiency of clusters running MapReduce applications.

1.2 Contributions

To address the challenges of thermal-aware resource management in energy-efficient clusters, our research investigates thermal-aware profiling of workloads and scheduling, thermal modeling of cluster nodes followed by per-application based approximation strategies that are capable of achieving high energy-efficiency and reduced cooling costs in clusters systems. The main contributions of this dissertation are the following:

- We provide detailed thermal evaluation of several Hadoop benchmarking applications. The resource access patterns are identified for performing thermal-aware job scheduling.

- We present a thermal-aware scheduling strategy, called *tDispatch*, which is conducive of reducing the temperatures of multicore processors and disks of data nodes by altering the order in which CPU-intensive and I/O-intensive jobs are executed in a batch manner. The thermal efficiency and performance of *tDispatch* is compared with those of the other two scheduler counterparts, namely, *CPUF* and *IOF*.

- We build a thermal model, called *tModel*, to estimate outlet air temperatures by characterizing the relationship among inlet and outlet air temperatures as well as multicore processor temperatures. In *tModel*, temperatures of multicore processors are modeled as a function of system configurations and applications' resources access patterns. Our *tModel* is conducive of predicting outlet air temperatures with high accuracy without relying on the power consumed by cluster nodes. Further, the average error of our model stays below 5% with all three chosen applications As a use case of our model, we extend *tModel* to predict cooling costs of a Hadoop cluster.

- We present a case study that focuses on applying approximation computation techniques to Hadoop applications. We demonstrate that approximation strategies are conducive of saving energy when exact solutions are not required, also ensuring that thermal constraints are not violated.

- We design a thermal-aware and approximation-aware framework, called *tHadoop2*. tHadoop2 orchestrates thermal-aware workload placement and applications resiliency to save energy in data centers. To facilitate the development of *tHadoop2*, we integrate an existing thermal-aware workload placement module called *tHadoop* with our *tHadoop2*. Our framework consists of three key components - *tHadoop*, a *thermal monitoring and profiling* module and a *approximation-aware thermal manager*.

1.3 Dissertation Outline

This dissertation is organized as follows.

Chapter 2 presents a brief review of existing research efforts in the area of thermal management. Special attention has been paid to the state-of-the-art techniques in thermal management, thermal modeling and job scheduling, and energy-efficiency for cluster systems.

Chapter 3 provides a background on the importance of thermal-aware job scheduling for energy efficiency, examining the impacts of CPU-intensive and I/O-intensive workloads on processors and disk. We develop a thermal-aware job scheduling strategy called *tDispatch* tailored for MapReduce applications running on Hadoop clusters. The scheduling idea of *tDispatch* is motivated by a profiling study of CPU-intensive and I/O-intensive jobs from the perspective of thermal efficiency. More specifically, we investigate the thermal behaviors of different types of workloads running on a Hadoop cluster by stress testing data nodes through extensive experiments. We demonstrate that *tDispatch* is conducive of creating opportunities to cool down multicore processors and disks in Hadoop clusters deployed in modern data centers.

Chapter 4 focuses on characterizing thermal profiles of cluster nodes - an integral part of any approach that addresses thermal emergencies in data centers. We begin our discussion by delineating a modeling framework. Then, we develop a baseline thermal model called *tModel* that projects outlet air temperatures from inlet air temperatures as well as directly measured multicore temperatures rather than deploying a CPU-utilization based model. We validate the accuracy of our model against data gathered by thermal sensors installed on our cluster nodes. Our results discussed in this chapter demonstrate that *tModel* estimates outlet air temperatures of cluster nodes with much higher accuracy over CPU-utilization based models.

Chapter 5 presents an emerging paradigm for saving energy by identifying any inherent resilience in applications. We present a case study on a Hadoop application where we apply an approximation strategy to observe reductions in execution times and power consumption. We design a thermal-aware approximate computing framework called *tHadoop2* and discuss the role of an existing module called *tHadoop* proposed in the past our research group. With our tHadoop2 in place, we reckon that approximation strategies help in reducing the execution times of MapReduce applications as well as achieving thermal-efficiency with the aid of thermal-aware data placement and thermal monitoring.

Finally, Chapter 6 concludes the dissertation by summarizing the main contributions of our research and suggests directions for future work.

Chapter 2

Related Work

With a dramatic increase in energy consumption of large-scale cluster computing systems, we must urgently address the energy efficiency issues. A variety of techniques have been proposed for improving the energy efficiency of data centers. This chapter briefly presents prior approaches that are most relevant to our research from the perspectives of thermal management, energy-efficient computing, thermal scheduling, and approximate computing.

2.1 Thermal Management

Recently, much attention has been paid to monitoring high temperatures inside data centers. One of the major reasons is that data centers now consume over 2% of the electricity in the United States as per an estimate by the Environmental Protection Agency (EPA). Various organizations are looking into ways of harnessing renewable sources of energy for cooling their data centers. Cooling consumes electricity so thermal management is needed to reduce the amount of electricity spent on cooling. The goal of thermal management is to reduce thermal hotspots and non-uniform on-chip temperatures that may impact the longevity of hardware [18]. Tang *et al.* proposed a heat recirculation model and developed an algorithm called MPIT-TA to reduce cooling costs by minimizing the peak inlet air temperature [19]. To reduce on-chip temperatures, Chaparro *et al.* investigated the correlation between processor utilization and temperature increase in processors [9]. Chaudhry *et al.* presented an array of thermal-aware scheduling policies tailored for green data centers, where heat modeling, thermal-aware monitoring and profiling are integrated [20]. Our study is different from the aforementioned research

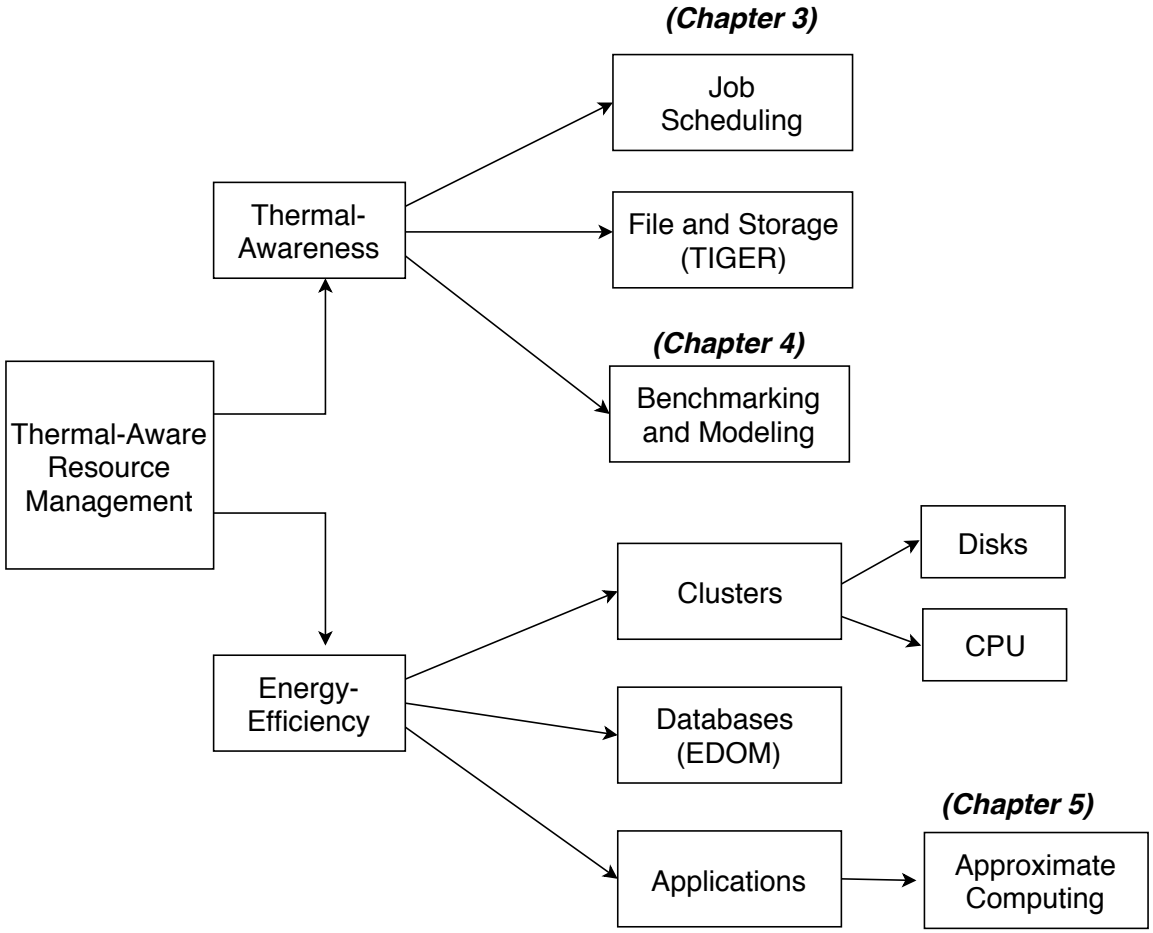


Figure 2.1: A simplified taxonomy of the approaches to the thermal management studied in this dissertation.

in essence that we focus on the thermal behaviors of both processors and disks in the realm of cluster computing.

High temperatures inside data centers not only increase the operational cost, but also penalize the performance of various components of servers. Low thermal efficiency may risk the overall safety of the data centers. In this dissertation, we pay particular attention to the effects on processors temperatures by studying applications' resource access patterns and the combined effects of incoming, outgoing, and ambient air temperatures.

In the past, thermal management techniques were only designed to address the peak temperature scenarios. Lately, much attention has been paid to processor temperatures and its impact on energy consumption of cluster nodes [18]. With a growing need for reducing the cooling costs, thermal management techniques must be employed in data centers to minimize the operational as well as the cooling costs. For example, Jiang *et al.* built a thermal model

to estimate the outgoing air temperature of a server node based on processor and disk utilizations [13]. This model estimates thermal impacts of various workloads on storage systems. Chaparro *et al.* proposed a thermally efficient frontend to reduce the on-chip temperature by (1) implementing a distributed rename and commit logic and (2) proposing a banked design on trace cache. These two techniques reduce temperatures in various components of microarchitectures [9]. While the aforementioned thermal management approaches are applicable to storage systems and microprocessors, our proposed scheme is tailored for thermal managers in Hadoop clusters.

Traditionally, reactive thermal management strategies have been employed in data centers to deal with thermal emergencies and hotspot prevention. In particular, *CFD* (i.e., Computational Fluids Dynamics) simulations that have been used to evaluate thermal performance of data centers with a given configuration. A downside of CFD simulations is the elongated time to obtain these simulation results. Since thermal emergencies should be dealt in a timely fashion to safely operate data centers, researchers have been continuously looking into improving thermal management.

Tang *et al.* developed a heat flow model that accelerates the thermal predictions as opposed to CFD to characterize hot air recirculation in data centers [21] . Tang's heat flow model relies on the use of sensors for gathering temperature data for their simulations-based study. To manage thermal emergencies without significant performance degradation, Ramoz and Bianchini created an online future temperature prediction framework by combining DVFS, request distribution, and request-admission control for Internet services [22].

In our research, we conduct experiments demonstrating that outlet air temperatures can be modeled as a function of multicore temperatures and inlet temperatures without necessarily relying on CPU utilization. Utilization-based thermal models may introduce errors due to inaccurate mappings from system utilization to outlet temperatures. Our model addresses this concern in the existing models by eliminating utilization models as a middle man.

2.2 Energy-Efficient Computing

Energy-efficiency is directly linked to thermal-awareness in cluster systems; by reducing the amount of heat in the system, energy-efficiency of cluster systems can be enhanced [5]. An increasing number of energy-saving techniques have been designed to reduce the energy costs of data centers. For example, a few common strategies adopted by modern data centers include raising the supplied air temperature, using atmospheric air directly for cooling, turning off idle-servers, and consolidating workloads to minimize the number of active nodes at a given time.

One way to improve energy efficiency is to develop thermal-aware job scheduling techniques (see, for example, [23] [24] [25] [26] [27] [28]). Beloglazov *et al.* presented a comprehensive study on the energy-efficient techniques that can be deployed in data centers and cloud environments [29]. Alsubaihi and Gaudiot developed *PETS* (i.e., Performance, Energy and Thermal-Aware Scheduler) that considers various scheduling constraints like job scheduling, core scaling, and thread allocation. PETS is able to improve execution time and energy consumption under peak power [25]. Cao *et al.* proposed a cooling strategy optimizing job-to-node mapping and reducing hotspot temperature by allocating power-hungry jobs to compute nodes [23]. By estimating the power consumption of jobs, Varsamopoulos [30] devised a cooling model applied in thermal-aware job scheduling algorithms that profiles the behavior of a CRAC systems; Computer Room Air Conditioning (CRAC) unit is a device that monitors and maintains the temperature, air distribution and humidity in a data center.

MapReduce-based computing frameworks like Yahoo!'s Hadoop or Amazon's Elastic MapReduce have been extensively used to process large-scale data and analysis. There are two very popular and successful techniques for improving the energy efficiency of Hadoop clusters. The first scheme is to allocate workload to only an appropriate number of nodes and to place the rest of the nodes in the inactive power mode. The second technique deploys the suitable nodes for a job by understanding its compute and storage features, thereby saving the energy from oversized components [31]. Acknowledging the architecture of Hadoop that allows it to transition nodes to and from active mode to inactive power mode, Leverich *et al.* designed an energy-efficient Hadoop cluster, in which a significant number of nodes are

transitioned into the low-power sleep mode while keeping active nodes fully functional [31]. Kaushik and Bhandarkar developed GreenHDFS, where a Hadoop cluster is divided into Hot and Cold zones [32]. A hot zone is a collection of nodes currently executing jobs, whereas a cold zone has nodes sitting idle in low-power mode. A zone's temperature is determined by its power consumption and the data is placed onto a suitable temperature zone by using data classification policies [32]. Our research is orthogonal to the above energy-efficient Hadoop techniques. Incorporating our thermal job scheduler into the existing energy-efficient Hadoop clusters can further improve the energy efficiency of data centers housing Hadoop clusters.

2.3 Job Scheduling

A large number of novel schedulers were proposed to optimize the energy and thermal efficiency of single machines [33][34]. Job schedulers play a vital role in improving energy efficiency and performance of large-scale clusters [35][36][37][38]. In this subsection, we summarize a list of existing job schedulers; we also compare our *tDispatch* with the existing scheduling solutions (see details in Table 2.1).

Thermal-aware job scheduling techniques aim at keeping the temperatures of all components of cluster nodes below a certain threshold while improving the energy-efficiency of the nodes at the same time. Recently, an increasing number of novel schedulers have been proposed to Hadoop clusters housed in data centers. For example, Pastorelli *et al.* designed a job scheduler called HFSP for Hadoop clusters. HFSP implements a size-based preemptive scheduling principle, which aims at reducing overall system response times and guaranteeing fairness by building job size information using virtual time and aging as metrics [39]. Zaharia *et al.* proposed a mechanism, *Delay Scheduling*, that achieves nearly 100% locality while slightly relaxing fairness. The *delay-scheduling* idea was integrated into Hadoop's fair scheduler to improve the response time of small jobs by a factor of five (5) and to double throughput in I/O-heavy workload [40]. Radheshyam *et al.* proposed a scheduling algorithm that involves task selection and task assignment to choose the best task suitable for a particular data node. Their algorithm applies heuristic and machine learning solutions to balance resources on clusters to reduce overall runtime of submitted jobs [41]. Zaharia *et al.* also developed a scheduler,

LATE, which uses estimated finish times to speculatively execute those tasks that degrade the response time [42]. LATE outperforms Hadoop's default speculative execution algorithm in real workloads on Amazon's Elastic Compute Cloud. Mashayekhy *et al.* proposed two energy-efficient MapReduce scheduling algorithms (i.e., *EMRSA*) assigning map and reduce tasks to appropriate nodes in order to minimize energy consumed in Hadoop clusters [43]. Mukherjee *et al.* present a comprehensive study on thermal scheduling algorithms by using a set of three heat models. The aim of their study is to minimize the maximum temperatures imposed by different workloads on any machine [44].

To deal with thermal emergencies and the emergence of hotspots, it is imperative to study the effects of individual applications on processor temperatures and leverage spatial job placement on various cores on a processor chip to reduce the overall heat buildup inside a node [18]. We propose a thermal-aware scheduler that dispatches Hadoop jobs based on their profiled thermal characteristics. While most of the existing Hadoop schedulers are improving overall cluster performance, our *tDispatch* is conducive to optimizing thermal efficiency and reducing cooling cost of Hadoop clusters in data centers. In particular, we strive to control the processor temperatures, which in turn contributes to saving energy as well.

Job Scheduler	**Hadoop**	**Cluster**	**Energy-efficient**	**Thermal-efficient**	**I/O-intensive**	**Fairness**
tDispatch (Ours)	✓	✓	✓	✓	✓	✓
HFSP [39]	✓	✓	✗	✗	✗	✓
Delay Scheduler [40]	✓	✓	✗	✗	✓	✓
MRSched [41]	✓	✓	✗	✗	✓	✗
LATE [42]	✓	✓	✗	✗	✗	✓
EMRSA [43] [45]	✓	✓	✓	✗	✗	✗
Green Data Centers [35]	✗	✓	✓	✓	✗	✗
CLB [36], IOCM [37]	✗	✓	✗	✗	✓	✓
MILP [33]	✗	✗	✓	✗	✗	✓
TARS [34]	✗	✗	✓	✓	✗	✗

Table 2.1: Comparisons between our *tDispatch* and the existing job schedulers.

2.4 Thermal Modeling

Like performance modeling that captures irregular resource usage patterns in clusters, thermal modeling is beneficial to various energy-optimizing tasks. Thermal models are deployed in energy-aware control algorithms for identifying cooling-efficient zones for computationally intensive workloads. A few schemes (see, for example, [46]) offer system performance analysis by modeling system behaviors. Thermal profiling was employed to examine an array of best practices like air management, optimizing the size of data centers, and utilizing free cooling by using chilled water.

A handful of thermal models tailored for thermal management in data centers have been proposed in the past decade [22], [47], [48] [46] [49] and [50]. For instance, Li *et al.* conducted a preliminary study on load distribution techniques, applying an analytical model for minimizing computing and cooling energy collectively [47]. Parolini *et al.* presented a cyber-physical system that takes advantage of thermal-aware file placement for data centers [51]. Similarly, Kaushik and Nahrstedt [52] investigated proactive, thermal-aware placement, which saves cooling costs without performance degradation.

In a few studies, processor variation was incorporated to create effective thermal models and schedulers. For example, owing to the variation in processor power efficiency, Rountree *et al.* [53] studied processor performance under power clamping; their study demonstrated that a power bound converts variation in processor power to variation in performance. Our tModel aims to predict outlet air temperatures to estimate cooling costs for CRACs in data centers. In the *tModel* research, we put thermal modeling of Hadoop clusters under a microscope. In particular, we investigate how the workload would have impacts on the temperatures of multiple cores in Hadoop applications, which in turn affect servers' outlet air temperatures.

2.5 Approximate Computing

Energy-efficiency is the biggest challenge in exascale computing. Approximate Computing (AC) has emerged as a design paradigm that enables energy savings and performance gains by incorporating modest relaxations in otherwise strict computation instructions. It is worth

mentioning that AC has gained a lot of attention in the last decade from the community of researchers, including programming languages, architecture, circuits, and application developers. Even research shows that energy-efficient applications can enhance the energy efficiency in cluster systems [5].

Recently, a few studies have shown that approximate computing techniques can be applied at four different levels, namely, application, software, hardware, and circuit [54] [55] [56] [57]. Venkataramani et al. presented a comprehensive study on the principles of approximate computing and a framework that integrates approximate computing techniques at various levels of the computing stack, namely circuit, architecture, and software [16]. Where at circuit and architecture levels, the attention is paid to reducing hardware complexity and employing approximate application-specific accelerators and programming processors respectively, at the software level, applications are partitioned into resilient and sensitive parts and such resilient parts are skipped during computations [16] [15]. A very similar framework designed by Chippa *et al.* identifies potential resilient computations in various application domains such as recognition, data mining, and search [15]. Goiri *et al.* designed a framework, ApproxHadoop, that uses sampling-based approaches to compute error bounds for popular classes of MapReduce programs. To achieve reductions in energy consumption and application execution time, ApproxHadoop considers input data sampling, task dropping, and a user-defined error-bound at a particular confidence level. Hui *et al.* designed a scheduling algorithm, *Good Enough (GE)*, that works by distributing power among the processor cores to save energy. GE is able to save energy by working in two execution modes - (1) Aggressive Energy: low power mode used for the majority of the experimentation and (2) Best Quality: used when user-specified quality falls below a predefined quality [58]. Akturk *et al.* present a comprehensive study on the accuracy metrics adopted for a wide variety of applications in the realm of approximate computing [59].

While these efforts are primarily concerned with error-resilient hardware, applications, and circuits, our primary contribution is in the analysis and characterization of MapReduce applications to find avenues for saving energy when exact results are not required. In this dissertation, we propose an approximation-aware thermal framework called*tHadoop2*. We assimilate an existing thermal-aware workload placement module called *tHadoop* with our *tHadoop2*.

tHadoop2 aids in reducing execution times in MapReduce application and improving thermal-efficiency thanks to thermal-aware data placement and continuous thermal monitoring.

Chapter 3

Thermal-aware Job Scheduling

Thermal-aware job scheduling incorporates knowledge of the thermal impact (i.e. the effect of the heat and temperature distribution) of a schedule on the cluster nodes. Thermal-aware job scheduling aims to produce schedules that yield minimal cooling needs and therefore are energy-efficient. Such schedules must not only increase energy-efficiency of systems without comprising on their computing performance.

In this chapter, we develop a thermal-aware job scheduling strategy called *tDispatch* tailored for MapReduce applications running on Hadoop clusters. The scheduling idea of *tDispatch* is motivated by a profiling study of CPU-intensive and I/O-intensive jobs from the perspective of thermal efficiency. More specifically, we investigate the thermal behaviors of these two types of jobs running on a Hadoop cluster by stress testing data nodes through extensive experiments. We show that CPU-intensive and I/O-intensive jobs exhibit various thermal and performance impacts on multicore processors and hard drives of Hadoop cluster nodes. After we quantify the thermal behaviors of Hadoop jobs on the master and data nodes of a cluster, we propose our scheduler to alternatively dispatch CPU-intensive and I/O-intensive jobs. We apply our strategy to several MapReduce applications with different resource consumption profiles. Our experimental results show that *tDispatch* is conducive of creating opportunities to cool down multicore processors and disks in Hadoop clusters deployed in modern data centers. Our findings can be applied in other thermal-efficient job schedulers that are aware of thermal behaviors of CPU-intensive and I/O-intensive applications submitted to Hadoop clusters.

The remainder of the chapter is organized as follows. Section 3.1 presents motivations for our *tDispatch* scheduler. Section 3.2 describes the experimental setups followed by the analysis

of the thermal behaviors of Hadoop applications. The description of the *tDispatch* scheduler can be found in Section 3.3. Section 3.4 shows the comparisons of *tDispatch* and the two job scheduler counterparts. Finally, Section 3.6 summarizes our results and concludes the chapter.

3.1 Motivations

Job schedulers play a vital role in improving energy efficiency and performance of large-scale clusters [60][61][62]. To understand and develop a relationship among thermal impacts of different types of jobs on a Hadoop cluster, we study the thermal behavior of master and data nodes on a homogeneous Hadoop cluster, where all the nodes share an identical configuration. After summarizing the experiment setups (see Section 3.2.1), we discuss the thermal behaviors of various MapReduce benchmarks (see Section 3.2).

Current literature focuses on the thermal efficiency of data centers [63][64], but no prior study investigated the impacts of Hadoop jobs on the overall heat buildup in a data center. In this study, we propose a thermal-aware scheduler called *tDispatch* to improve energy efficiency of Hadoop clusters and lowering cooling costs. We show that CPU-intensive and I/O-intensive jobs exhibit distinct thermal behaviors on multicore processors and hard drives of Hadoop clusters. We also demonstrate that one way to make Hadoop clusters thermal friendly is to dispatch a mix of CPU-intensive and I/O-intensive jobs to Hadoop clusters.

Our *tDispatch* job scheduler facilitates an essential aid to data centers as *tDispatch* ensures stable building plans and energy efficiency factors. The findings of this study help in designing cost-effective cooling management solutions to modern data centers.

There are four main motivations for working toward thermal-aware job scheduling in Hadoop clusters:

1. ability of Hadoop to manage big data.
2. augmenting cooling and electricity costs of data centers.
3. thermal profiles of Hadoop jobs running on clusters.
4. lack of thermal-aware schedulers tailored for both CPU-intensive and I/O-intensive jobs.

3.2 Thermal Behaviors of the Benchmarks

In this section, we study the thermal behaviors of four MapReduce benchmarks applications, namely, WordCount, Sort, PageRank, and TeraSort.

3.2.1 Experimental Testbed

Hadoop Server

In this group of experiments, we observe thermal behavior of a homogeneous cluster comprising of 15 nodes (labeled as *backend-1-0 to backend-1-16*) that are placed on a rack in our HPC room (*backend-1-4* and *backend-1-5* were not available for this experiment). The room temperature is set to 20°C. Table 3.1 describes the configuration of these nodes.

Hardware	
Computer	SuperMicro Model 825-7
CPU	Intel (R) Xeon (R) X5650@2.67GHz (cores: 24 with 2 H/W threads each)
Memory	24 GB DDR3 SDRAM at 1066 MHz
Network controller	GigaBit Ethernet and 20 GB InfiniBand
Disks	500 GB
Software	
Operating System	Cent OS 7.2 64-bit
Hadoop	Release 2.7.3
Linux Kernel	3.10.0

Table 3.1: Hardware and software configurations of the experimental testbed

MapReduce is a framework for processing parallelizable problems across a multitude of computing nodes, collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogenous hardware) [65]. Furthermore, many MapReduce applications show significant variation in resources (CPU/disks/DRAM) consumption during their execution. An I/O-intensive application spends a significant portion of the time processing I/O operations rather than computing tasks. A CPU-intensive application, in contrast, infrequently generates I/O requests while spending a vast majority of time doing computations [66]. MapReduce applications range from CPU-intensive to I/O-intensive, and the

HiBench Input Size	Wordcount	TeraSort	Sort	PageRank no of pages
Tiny	3200	3200	32000	50
Small	320000000	320000	3200000	500
Large	3200000000	32000000	320000000	500000
Huge	32000000000	320000000	3200000000	5000000
Gigantic	320000000000	3200000000	32000000000	50000000
Big Data	1600000000000	6000000000	300000000000	30000000

Table 3.2: Various HiBench inputs sizes and corresponding sizes in Megabytes (MB)

workload of each subtask is either CPU-bound or I/O-bound. To investigate the variation in the behaviors of MapReduce applications as per their characteristics (I/O-bound or CPU-bound), we run a mix of I/O-intensive and CPU-intensive Hadoop applications on a homogeneous cluster. We choose a few micro benchmarks from a popular MapReduce benchmark suite by Intel called *HiBench* [67].

We choose four popular Hadoop benchmarks, namely, WordCount, PageRank, Sort, and TeraSort and we vary the input data sizes (tiny, small, large, huge, and gigantic) for each of these four benchmarks as available with *HiBench* benchmark suite [67]. To measure processor temperatures, we use a Linux utility program called *lm-sensors*; we apply another utility called *hddtemp* to monitor internal hard drive temperatures; we make use of the *iostat* tool to measure CPU and disk utilization. CPU utilization represents the computing load across all the four cores, whereas disk utilization represents I/O load imposed by tested Hadoop applications. Further, we use *Ganglia*, a monitoring system for high-performance computing systems (e.g., clusters). We use the elapsed wall clock time as execution time. We emphasize here that our work is comprised of measuring actual execution time, CPU/disk temperatures, and utilization of the aforementioned MapReduce applications without relying on any simulation data.

The following factors inspire us to investigate thermal behaviors of homogeneous clusters without addressing the air re-circulation and node placement issues.

- In this study, we focus on homogeneous clusters, because nodes in a brand new cluster are typically homogeneous in nature. It is worth mentioning that our thermal profiling research can be extended to heterogeneous clusters.

- For the purpose of measurements, the evaluated nodes are placed in a traditional rack. The node placement in a rack, of course, can affect air re-circulation in a data center; air re-circulation is beyond the scope of this study. Our thermal profiling results can be incorporated into a data center's thermal model, where air re-circulation is taken into account.

In the following sections, we will analyze the thermal behavior of our cluster running Hadoop with the aid of graphs obtained from *Ganglia*.

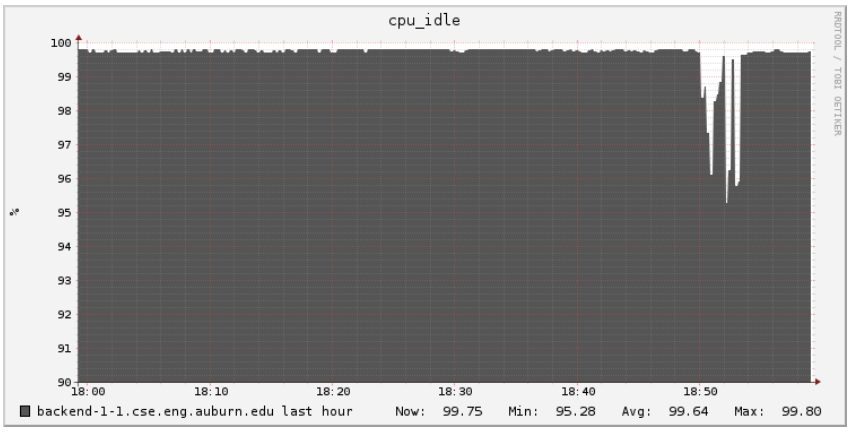
3.2.2 WordCount

The tested WordCount is a MapReduce application running on a Hadoop cluster, where text files are loaded and the frequency of each word in the files is counted. The input to Wordcount is text file(s), and output is also a text file containing words followed by number of times they occur. By the nature of it, WordCount is a CPU-bound application as it spends most of its time in computing.

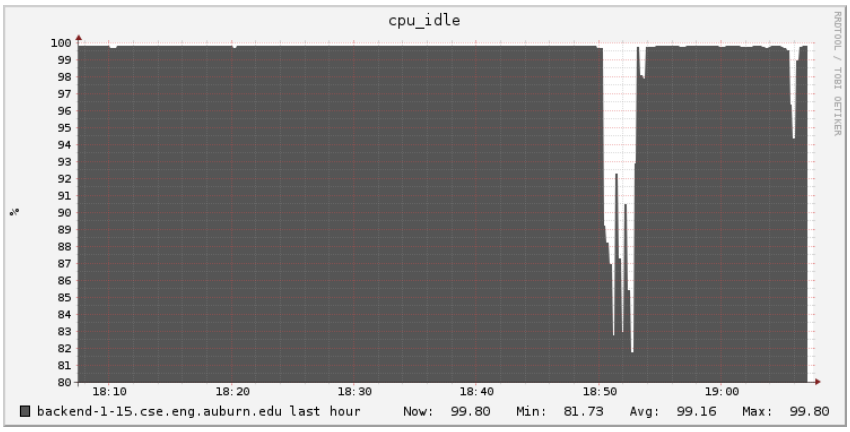
The purpose of using WordCount is to study the thermal properties of the processors of the Hadoop cluster's nodes by keeping CPU extremely busy. We make WordCount process a huge-sized data (i.e., 32 GB) available through *HiBench* in order to make the experiment run for a long duration of time, which helps us to better understand of the cluster's behavior.

Fig. 3.1(a) shows the CPU utilization of all the cores residing in the master node. We observe that CPU utilization of the master node change insignificantly; the disk temperature stays almost as a constant (not presented here). This thermal trend is expected, because the master node is simply responsible for dispatching MapReduce jobs to the data nodes without dealing with any CPU-bound or I/O-bound tasks. For example, Fig. 3.2(a) confirms that the CPU temperature of the master node is extremely low.

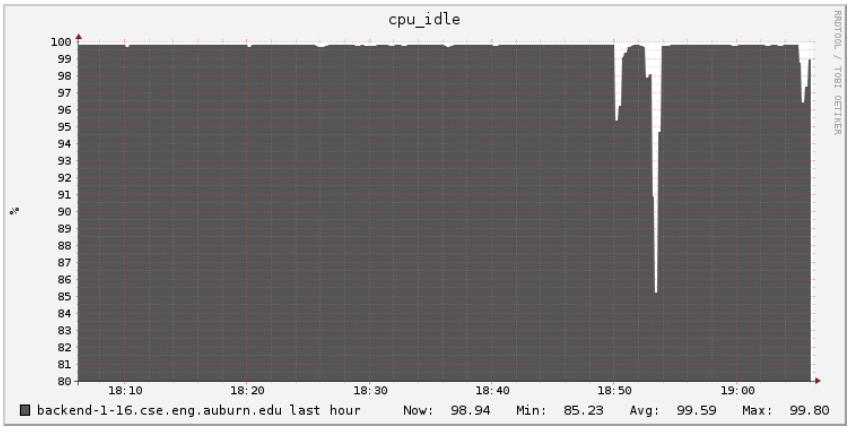
We measure and collect thermal trends of the data node(s) that ran the Wordcount. Due to space limitation and the similar trends of all the data nodes, we only present the thermal results of the two nodes in Figs. 3.2(b) and 3.2(c), each of which depicts the temperatures of the twelve cores as functions of time. Figs. 3.2(b) and 3.2(c) show that the temperatures of all the cores increase gradually. Not surprisingly, the data nodes' CPU temperatures are much higher than



(a) CPU Utilization (%) of the *master* node



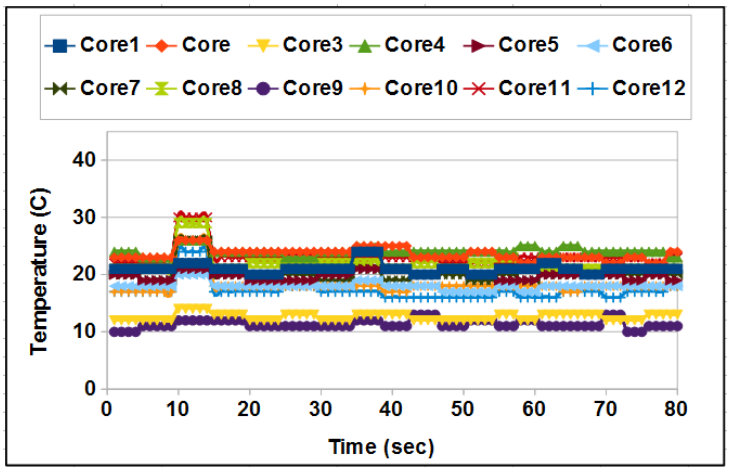
(b) CPU Utilization (%) of a *data* node



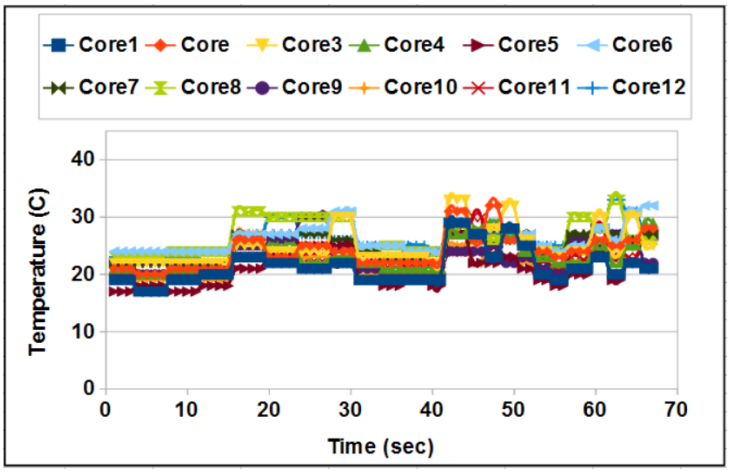
(c) CPU Utilization (%) of another *data* node

Figure 3.1: Figures show the CPU utilization (%) of WordCount application running on data nodes in our cluster.

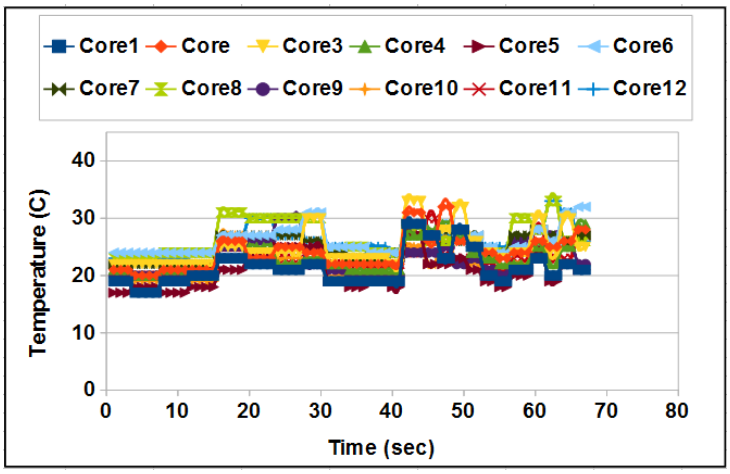
those of the master node (see also Fig. 3.2(a)), because most of the work is allocated to the data nodes. Moreover, we noticed in our results that there is not much deviation in CPU core temperatures inside the same chip. The average CPU temperature of all the multiple cores in



(a) CPU temperature of all cores on the *master* node



(b) CPU temperature of all cores on a *data* node



(c) CPU temperature of all cores on another *data* node

Figure 3.2: Average CPU temperatures of the master and two data nodes collected during the execution of Wordcount application.

one data node (see Fig. 3.2(b)) is 28.5°C. The CPU temperatures start diminishing towards the end of the experiments on all the data nodes.

We observed that compared to the processor temperatures, disk temperatures are insensitive to the WordCount workload. More specifically, the disk temperatures rise very slowly as the experiment proceeded. It takes around 15-20 minutes for the disks to heat up by 3°C on the slave nodes. Since the disks are not as thermally sensitive to workload as processors, no cooling down phase of the disks is observed immediately at the end of the experiments. For the same reason, we do not present the disk temperature here and thus, focusing only on CPU temperatures.

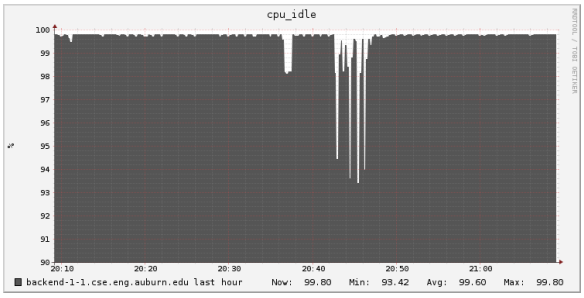
Figs. 3.1(a), 3.1(b), and 3.1(c) show the CPU utilization patterns on master and data nodes of our cluster running WordCount. We observe that for a slight increase in CPU utilization on slave nodes, the CPU temperature shoots up significantly (see Figs. 3.2(b) and 3.2(c)).

3.2.3 TeraSort

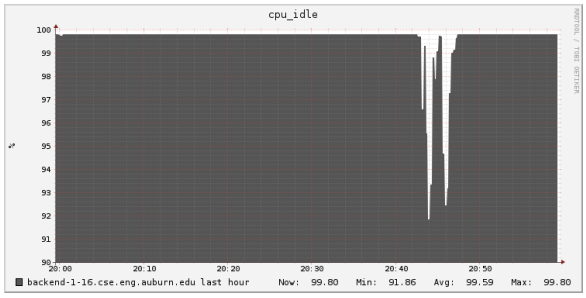
We make use of TeraSort - a MapReduce benchmark - to measure the execution time spent in sorting terabytes of data on a Hadoop cluster. The performance of TeraSort depends on the amount of available resources on the cluster. Unlike WordCount, TeraSort spends a significant amount of time performing input-output operations; we classify TeraSort as an I/O-intensive application.

We make TeraSort process a large-sized data available with HiBench benchmark suite (i.e., 32 GB). Unsurprisingly, we observe similar trends in CPU temperature among all the data nodes and; therefore, we select the data node which was deployed to run TeraSort and the results are shown in this subsection. We have represented CPU utilization and temperature trends here.

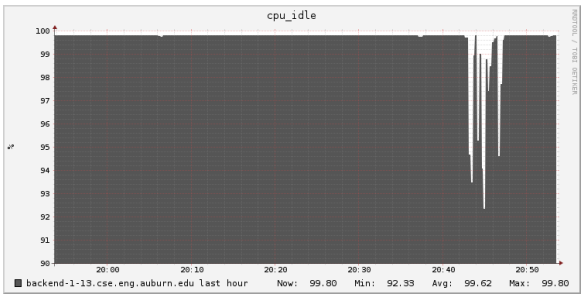
The input data of TeraSort is partitioned equally among the available cores; therefore, all the cores on each node exhibit identical heat patterns. We observe interesting heat patterns during the execution of TeraSort. We present the CPU utilization patterns for the master node (see fig. 3.4(a)) and two representative data nodes (see fig. 3.4(b) and fig. 3.4(b)). We observe that



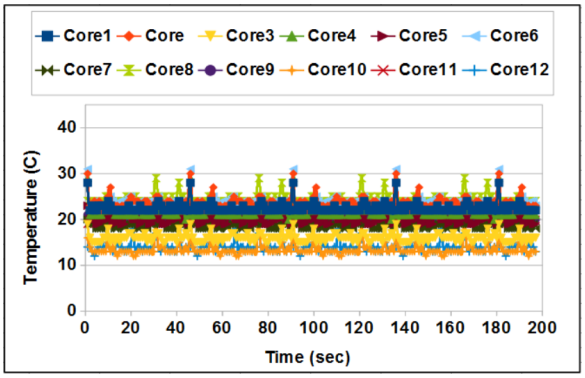
(a) CPU Utilization (%) of the *master* node



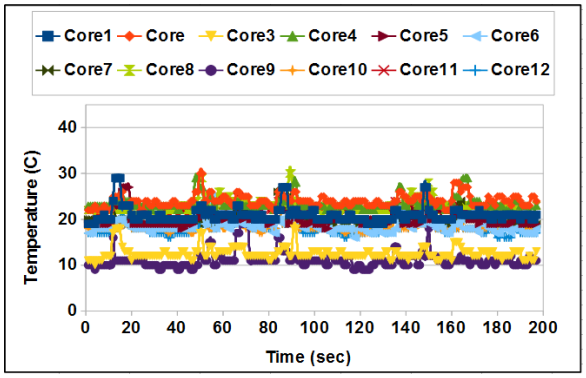
(b) CPU Utilization (%) of a *data* node



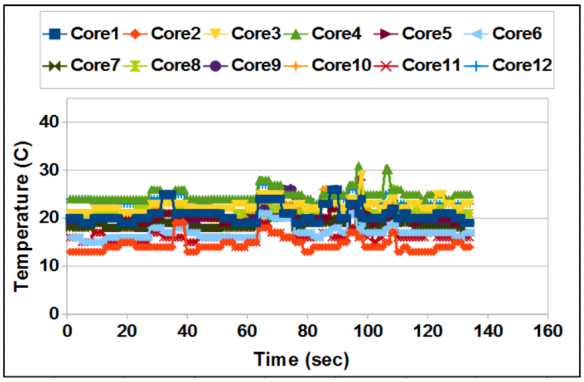
(c) CPU Utilization (%) of another data node



(d) CPU Core temperatures of the master node

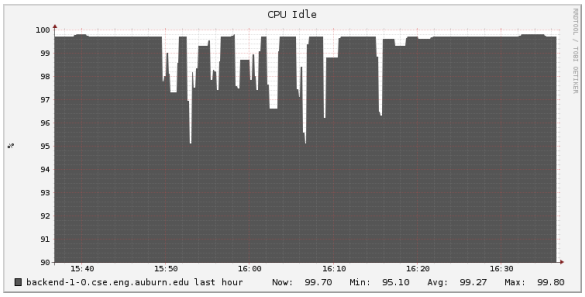


(e) CPU Core temperatures of a data node

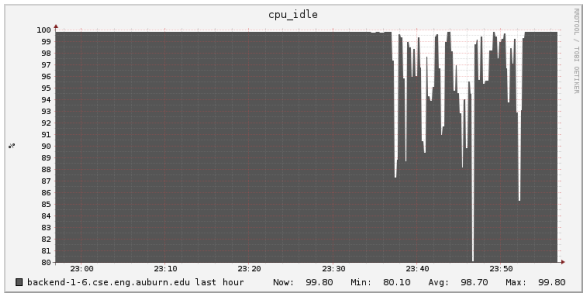


(f) CPU Core temperatures of another data node

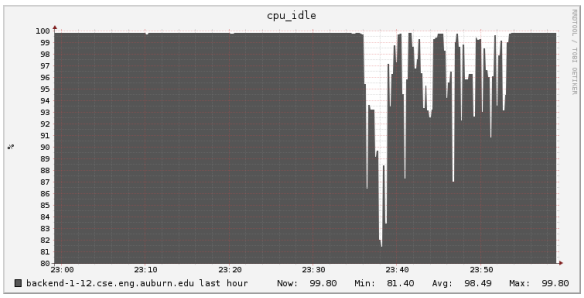
Figure 3.3: Average CPU utilization and temperatures of all twelve CPU cores of the master and two data nodes of the Hadoop cluster running TeraSort benchmark.



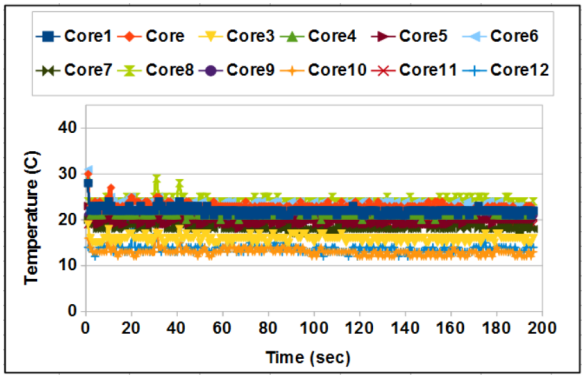
(a) CPU utilization (%) of the master node



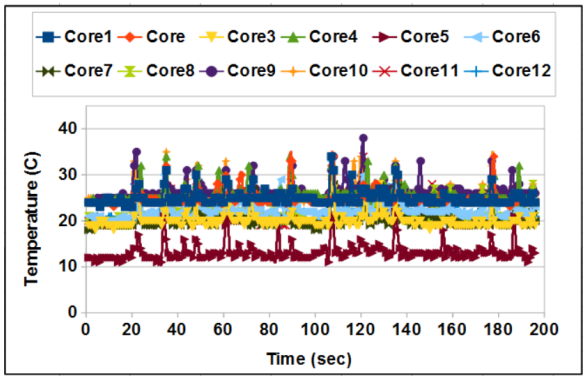
(b) CPU utilization (%) of a data node



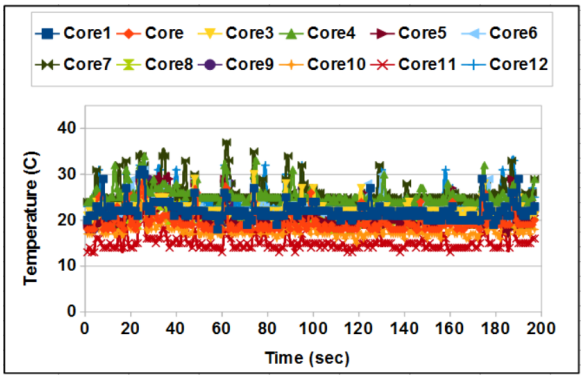
(c) CPU utilization (%) of another data node



(d) CPU temperature of the master node



(e) CPU temperature of a data node



(f) CPU temperature of another data node

Figure 3.4: Average CPU utilization and temperatures of all cores of the master and two data nodes of the Hadoop cluster running *PageRank*

the CPU utilization is significantly low for disk-intensive jobs. Fig. 3.3(d) shows the temperature of the twelve cores and on the master node. Because the master node allocates a majority of the work to the data nodes, the CPU utilization remains at a low level on the master. As such, the corresponding temperatures are also remain low. As the data nodes starts processing the job assigned to them by the master node, the disk temperatures on the slave nodes start rising due to the extensive disk usage. The CPU temperature rises occasionally but remains steady for most of the time (see Figs. 3.3(e) and 3.3(f)).

Similar trends for *Sort* are observed but in the interest of space, the figures are not represented here. The CPU utilization as well as the CPU temperatures stay very minimal with running huge-sized Sort.

3.2.4 PageRank

Now we test PageRank, which is a open source implementation of page-rank algorithm and is used in web search engines that calculate the ranks of web pages according to the numbers of references links [67]. This is helpful in finding the most relevant information for the searched keyword(s). Page ranking in Hadoop involves parsing, calculating, and ordering.

PageRank spends considerable amount of time on iterating small jobs and most of these jobs are CPU bound, with some memory and disk I/O consumption. The goal of this group of experiments is to study PageRank's behaviors. Compared to its counterparts (i.e., WordCount, Sort, and TeraSort), PageRank has the longest execution time given the same size of input data. Thus, it is an ideal candidate to examine the thermal behaviors of the cluster in a long interval. We set the PageRank's input data size to 32 GB (which is categorized as *huge* as per *HiBench*'s default configuration) and run PageRank on this data set for ten time to understand the behavior clearly. Similar to the previous experiments, we only present the results of the three data nodes; disk utilization trends are omitted in this subsection.

Figs. 3.4(a), 3.4(b), and 3.4(c) show the CPU utilization of the master and two data nodes. Fig. 3.4(d) shows the temperatures of all 12 cores residing on the master node. The CPU thermal trends are expected, because the role of the master node is to dispatch all the tasks to the data nodes.

Figs. 3.4(e) and 3.4(f) demonstrate the behavior of PageRank on the data nodes, where we observe an interesting pattern of the CPU temperatures. The CPU temperatures start rising as the experiment progresses and remain considerable high throughout the experiment. The maximum CPU core temperature on one of the data nodes is 39°C (see Fig. 3.4(e)) while the maximum disk temperature on another data node is 38°C (see Fig. 3.4(f)).

3.3 A Thermal-aware Job Scheduler

In this section, we propose *tDispatch* - a thermal-aware job scheduler aiming to reduce the CPU and disk temperatures of Hadoop clusters.

During the process of designing *tDispatch*, we answer the following two questions:

- How should we classify Hadoop jobs into CPU-intensive and I/O-intensive categories? (see Section 3.3.1)
- How do we make use of these two job categories to improve thermal management of Hadoop clusters? (see Section 3.3.2)

3.3.1 Job Profiling

We run and profile WordCount and TeraSort on a Hadoop cluster. We only run one job at a time and collect the CPU and disk temperatures of the cluster nodes.

Using the results of our prior experiments, we identify the access patterns of MapReduce applications by taking resource utilization of subtasks into account. Since each subtask issues either computation or I/O requests, we can categorize these benchmarks in one of the two categories: CPU-Intensive or I/O-intensive.

- *CPU-intensive Jobs*: A submitted application is called a CPU-intensive job, if its CPU load dominates the overall workload of the job. A CPU-intensive job spends more time in computation than processing disk or network I/Os.
- *I/O-intensive Jobs*: We refer to a submitted application as an I/O-intensive job when disk or network I/O activities dominate the job's load. Unlike CPU-intensive jobs, I/O-intensive ones spend a significant amount of time processing disk for network I/Os. In

this study, we pay particular attention to disk-intensive jobs. Nevertheless, the scheduler is also applicable to network-intensive jobs.

We classify Hadoop jobs into the CPU-intensive and I/O-intensive ones, because CPU and disk temperatures are highly sensitive to CPU and disk utilization, respectively. I/O-intensive jobs have the property of performing only a small amount of computation before performing I/O. Such jobs typically do not use up their entire CPU allocation. CPU-intensive jobs, on the other hand, use their entire CPU allocation without performing any blocking I/O operations. In what follows, we show how to rely on profiling information to determine if a job is a CPU-intensive or I/O-intensive one. The *tDispatch* makes use of this information to dispatch jobs in a thermal-friendly way. We also show that CPU-intensive jobs have significant thermal impacts on multicore processors, whereas I/O-intensive jobs noticeably increase disk temperatures of Hadoop clusters.

3.3.2 Job Scheduler Design

In this part of the study, we design a thermal-aware job scheduling strategy aimed to keep CPU and disk temperatures of cluster nodes as low as possible. We design *tDispatch* - a job scheduler tailored for big-data applications running on large-scale Hadoop clusters deployed in data centers. Hadoop's fair scheduling policy assigns a fair share of CPU resources to all running jobs with minimum starvation. *tDispatch* addresses the downside of the existing job schedulers that overlook thermal efficiency, the improvement of which has become a critical concern for data centers.

3.3.3 Two Modules

Our job scheduling mechanism consists of two key components, namely, *tQueue* and *tDispatch*(Fig. 3.5). The *tQueue* module is responsible for maintaining two dedicated queues - one for CPU-intensive jobs (i.e., Q^c) and one for I/O-intensive jobs (i.e., Q^d). The master node receives incoming jobs through clients. The received jobs are appended to one of the two queues based on their types. If a job is CPU-intensive, it is queued up into Q^c; otherwise, the I/O-intensive job is queued up in Q^d.

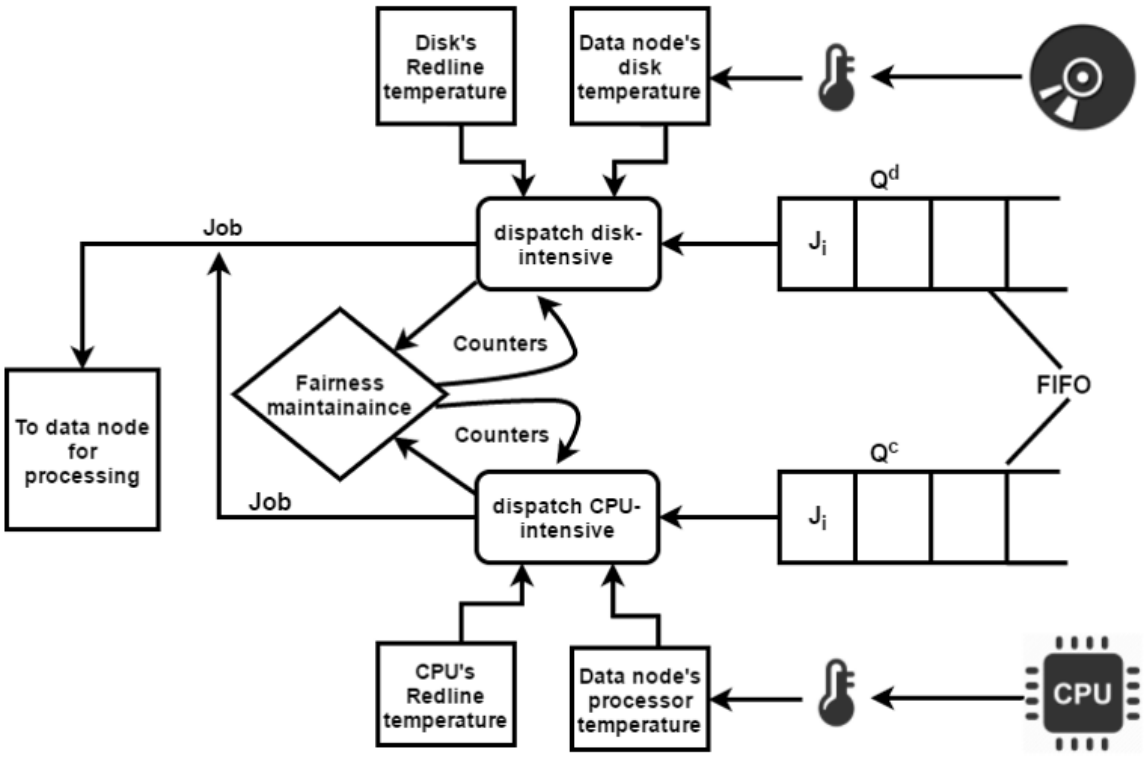


Figure 3.5: Flow chart diagram of two modules, *tQueue* and *tDispatch*, of our proposed scheduler.

Algorithm 1 presents the pseudocode of *tQueue*. It is noteworthy that *tQueue* has high extensibility, because *tQueue* can be readily extended by managing additional queues for memory-intensive jobs or jobs with other types of characteristics.

Algorithm 1: tQueue. **Input:** Job J_i; **Output:** Q^c, Q^d.

```
1: if J_i.type = I/O − intensive then
2:     Q^d.enqueue(J_i);
3: else
4:     Q^c.enqueue(J_i); /*job j is CPU-intensive*/
5: end if
```

We design *tDispatch* (see Algorithm 2) to allocate resources of a Hadoop cluster to one of the jobs in queues Q^c and Q^d, which are managed by *tQueue*. After a running job finishes its execution, *tDispatch* thermal-efficiently picks another candidate job from Q^c and Q^d to run on the cluster. In our design, the interface between *tQueue* and *tDispatch* are Q^c and Q^d.

3.3.4 Fairness and Starvation Avoidance

One novelty of our design is that *tDispatch* allows CPU and disk temperatures to cool by dispatching two types of jobs - I/O-intensive and CPU-intensive respectively. To avoid starvation and guarantee fairness between these two types of jobs, we ensure that the number of consecutive runs of the same type of jobs is always below an upper bound (see S in Lines 6 and 15 of Algorithm 2). In doing so, no job will be starved in the queues. This fairness-maintaining idea is implemented by introducing two counters W_c and W_d (see Line 1 of Algorithm 2), which keep track of the number of times one type of jobs' execution has been postponed. For example, if the scheduler repeatedly dispatches CPU-intensive jobs from queue Q_c to reduce disk temperatures, then counter W_d is increased by 1 after each CPU-intensive job is dispatched (see Lines 6-9 of Algorithm 2). When counter W_d reaches the upperbound S, an I/O-intensive job must be dispatched regardless of disk temperatures (see Lines 10-13 of Algorithm 2). In this case, counter W_d is reset to 0 after the I/O-intensive job starts its execution, thereby offering additional opportunities to continue cooling down disks.

3.3.5 Thermal Awareness

During the process of choosing the best job candidate from the queue Q_c and Q_d, algorithm *tDispatch* checks if the average CPU and disk temperatures of the Hadoop cluster are well below redline temperatures, which are recommended by manufacturers to safely operate devices. Here we denote R_c and R_d as the threshold temperatures of CPUs and disks. When the cluster's average CPU temperature exceeds the threshold R_c, *tDispatch* starts running I/O-intensive jobs rather than CPU-intensive ones to lower CPU temperatures.

tDispatch makes an effort to keep CPU and disk temperatures below thresholds R_c and R_d. *tDispatch* achieves this goal by taking advantage of the fact that a CPU-intensive jobs tend to further raise the CPU temperatures whereas an I/O-intensive job inevitably augment the disk temperature. Therefore, when the CPU temperature is nearly equal to temperature threshold R_c, *tDispatch* postpones the execution of any CPU-intensive job by dispatching I/O-intensive

Algorithm 2: tDispatch. **Input:** Q^c, Q^d.

Initialize $W_c \leftarrow 0$, $W_d \leftarrow 0$
while a job finishes its execution **do**
  if $i.t_c < R_c$ **then**
    if $i.t_d < R_d$ **then**
      dispatch a job from Q^c and Q^d using FIFO;
    else if $i.t_d > R_d$ and $W_d < S$ **then**
      dispatch a job from Q^c to node i;
      $W_c \leftarrow 0$;
      $W_d \leftarrow W_d + 1$;
    else
      dispatch a job from Q^d to node i;
      $W_d \leftarrow 0$
    end if
  else if $i.t_c > R_c$ **then**
    if $W_c < S$ **then**
      dispatch a job from Q^d;
      $W_d \leftarrow 0$;
      $W_c \leftarrow W_c + 1$;
    else
      dispatch a job from Q^c;
      $W_c \leftarrow 0$;
    end if
  else
    dispatch a job from Q^c and Q^d using FIFO;
  end if
end while

jobs in queue Q_d. Consequently, *tDispatch* keeps the overall CPU temperature low (see Lines 14-20 of Algorithm 2).

Similarly, *tDispatch* attempts to reduce overall disk temperatures (see Lines 6-9 of Algorithm 2). Specifically, if the cluster's average disk temperature goes beyond threshold R_d, then CPU-intensive jobs rather than I/O-intensive ones are dispatched to the cluster.

3.3.6 Two Extreme Scenarios

If both average CPU and disk temperatures are below threshold temperatures R_c and R_d, then our *tDispatch* scheduler follows the *FIFO* policy to pick a job with the earliest arrival time from queues Q_c and Q_d, and dispatches the chosen job to the Hadoop cluster for execution (see Lines 3-5 of Algorithm 2).

Under heavy workload conditions, both CPU and disk temperatures are likely to exceed threshold R_c and R_d. In this extreme case, *tDispatch* applies *FIFO* to schedule jobs in queues Q_c and Q_d (see Line 24 of Algorithm 2).

3.3.7 Extensibility

Since our cluster is homogeneous in nature, the CPU and disk threshold temperatures (i.e., R_c and R_d) are identical across all the nodes. Nevertheless, our algorithm can be extended to handle the heterogeneous case where multiple nodes have various CPU and disk redline temperatures. When it comes to heterogeneous clusters, we replace threshold temperatures R_c and R_d with two threshold vectors $\hat{R}_c$ and $\hat{R}_d$, where different nodes have distinct threshold temperatures maintained in the two vectors.

In the aforementioned two extreme scenarios (i.e., very light or heavy load, see Section 3.3.6), *tDispatch* applies the *FIFO* policy. Our algorithm *tDispatch* is extensible in a way that *FIFO* can be readily substituted by any other policy that optimizes overall system performance. For example, if we adopt *tDispatch* to deal with real-time jobs, a good scheduling policy to replace *FIFO* is the earliest deadline first or *EDF*.

3.4 Results and Discussions

In this section, we evaluate the thermal efficiency and performance of our proposed thermal-aware job scheduling strategy (i.e., *tDispatch*). The goal of our experiments is to demonstrate that the *tDispatch* scheduler is capable of improving thermal efficiency of Hadoop clusters by the virtue of its thermal awareness.

3.4.1 CPU-intensive and I/O-intensive Jobs

We are focusing on two popular types of Hadoop jobs, namely, CPU-intensive and I/O-intensive jobs. The tested Hadoop jobs were implemented using the MapReduce programming model, where each MapReduce application comprises of a large number of mapper and reducer tasks running on clusters. These mappers and reducers may be either CPU-bound, I/O-bound, or network bound. CPU-bound tasks tend to increase processor temperatures, whereas I/O-bound jobs noticeably affect disk temperatures. We choose WordCount and PageRank as the CPU-intensive jobs; we also run TeraSort and Sort as representatives of I/O-intensive jobs.

For fair comparison, we set the size of data for each job, regardless of its type, to be *HiBench*-large. The duration of these jobs is short and this short execution time resembles real-world cloud computing environments where a time-sharing policy is enforced.

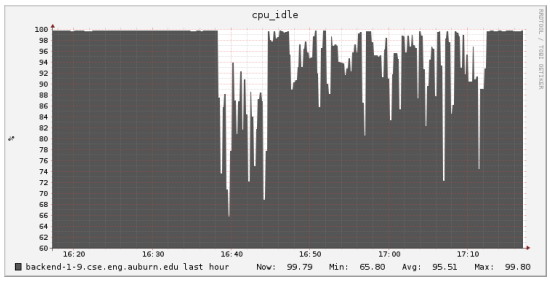
With our scheduler in place, system administrators can easily maintain the overall temperature of a Hadoop cluster below a specified redline temperature by lowering the supplied power of a cooling system's. Hence, our thermal-aware scheduler offers an energy-saving capability through improving the cooling system's energy efficiency.

3.4.2 Baseline Schedulers

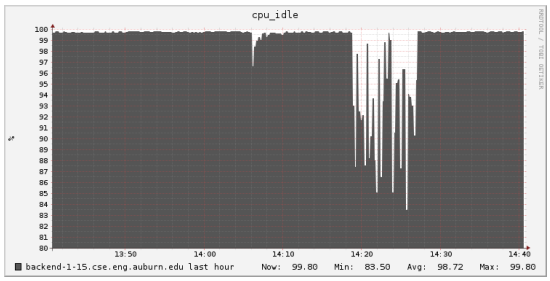
To demonstrate the thermal awareness of *tDispatch*, we compare our *tDispatch* with two baseline schedulers. The first one called *IOF* (i.e., I/O-intensive job first) gives high priority to I/O-intensive jobs; the second scheduler named *CPUF* (i.e., CPU-intensive job first) chooses

CPU-intensive jobs to be executed prior to any I/O-intensive ones. These two schedulers favor different types of Hadoop jobs submitted in a queue of the evaluated Hadoop cluster. We compare the thermal effectiveness of our *tDispatch* with those of IOF and CPUF.

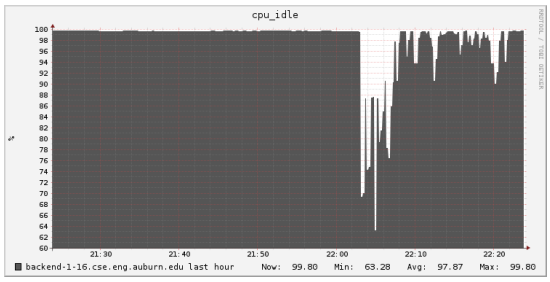
In addition to evaluating the thermal behavior of *tDispatch*, we investigate whether favoring CPU-intensive jobs is more thermal friendly than favoring I/O-intensive jobs on Hadoop clusters or vice versa. In one of our future studies, we will compare *tDispatch* other candidate schedulers including fair scheduler and FIFO.



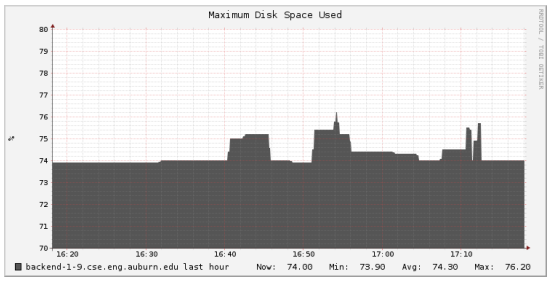
(a) CPU utilization (%) of a data node with CPUF



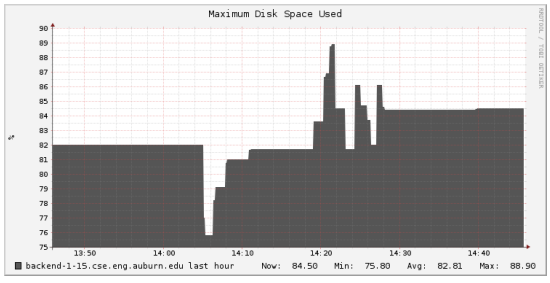
(b) CPU utilization (%) of a data node with IOF



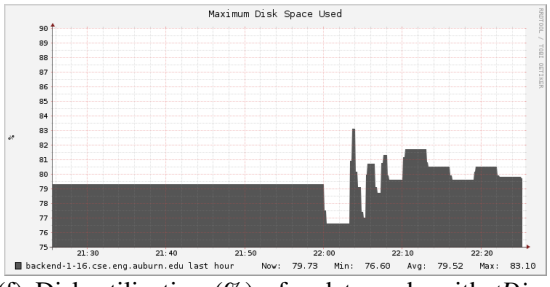
(c) CPU utilization (%) of a data node with *tDispatch*



(d) Disk utilization (%) of a data node with CPUF



(e) Disk utilization (%) of a data node with IOF



(f) Disk utilization (%) of a data node with *tDispatch*

Figure 3.6: Average CPU and disk utilizations of the master and data nodes on a Hadoop cluster running Terasort, WordCount, PageRank, and Sort with three different schedulers: CPUF, IOF, and *tDispatch*.

3.4.3 Experiment Methodology

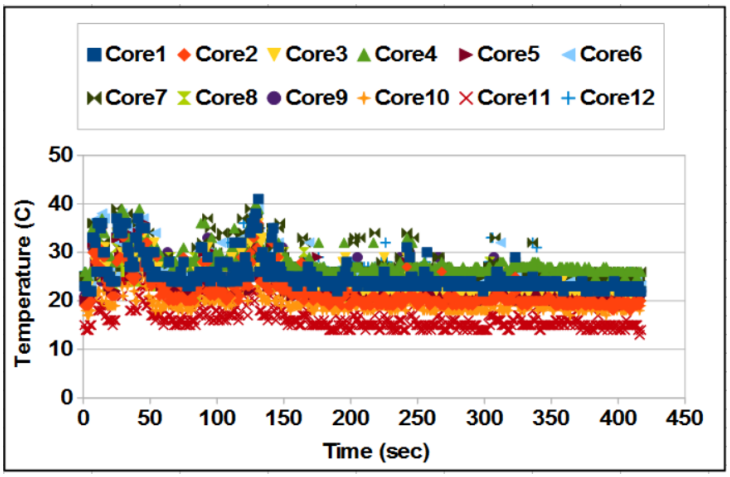
In each test, we submit a total of 20 *HiBench-large* sized CPU- and I/O-intensive jobs to our Hadoop cluster, where a dispatch queue (a.k.a., batch queue) maintains all the submitted jobs. In our experiments, two CPU-intensive jobs are *WordCount* and *PageRank*; two I/O-intensive jobs including *Terasort* and *Sort*. After all the jobs are submitted to the batch queue, our scheduler picks an appropriate job to be launched on the Hadoop cluster. When a job completes its execution, the scheduler chooses the next most appropriate job to be executed on the cluster nodes.

We keep track of the CPU and disk temperatures of the master and data nodes in the Hadoop cluster. The temperature measurements allow us to quantitatively assess the thermal-efficiency improvement offered by *tDispatch* from the perspective of both CPU and disk resources.

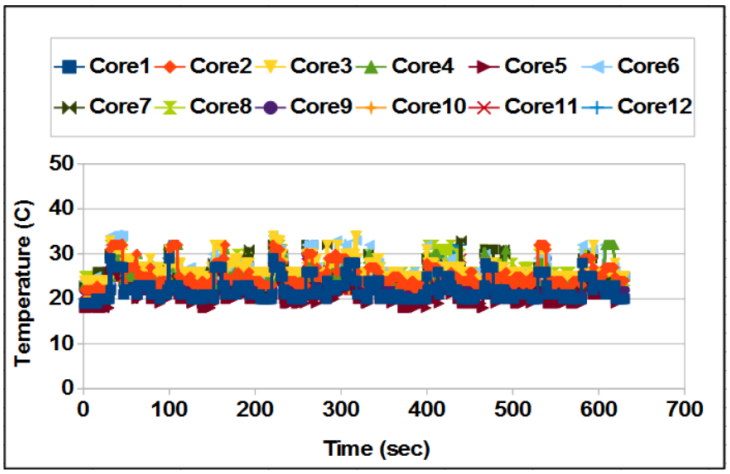
In addition to collecting CPU and disk temperatures, we monitor CPU and disk utilization during the course of running the submitted jobs. In the interest of space, we did not present disk temperatures since disks take a considerable amount of time to heat up and thus do not show interesting trends. We intend to show that *tDispatch* is conducive of reducing temperatures of system components without making any adverse impact on the performance of Hadoop clusters.

3.4.4 Master Node

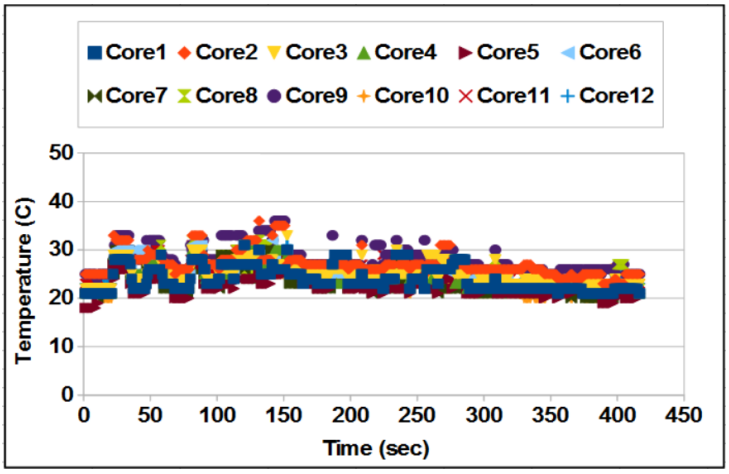
In the first group of experiments, we measure the CPU and disk temperatures of the master node governed by the three schedulers. We observe that the schedulers have little impact on the thermal efficiency of the master node. Comparing Fig. 3.7 with Figs. 3.1(a) and 3.3(a) found in Section 3.2, we conclude that both Hadoop jobs and the scheduler make no noticeable thermal impact on the master node. Because the master node is not a thermal-efficiency bottleneck, the results suggest that we should pay attention to improve the thermal efficiency of data nodes rather than a master node.



(a) CPU core temperature of a data node with CPUF



(b) CPU core temperatures of a data node with IOF



(c) CPU core temperatures of a data node with *tDispatch*

Figure 3.7: Average CPU temperatures of the master and data nodes node on a Hadoop cluster running Terasort, WordCount, PageRank, and Sort with different schedulers: PUF, IOF, and *tDispatch*.

3.4.5 Data Nodes

In this group of experiments, we compare the thermal impacts of the three tested schedulers on the Hadoop cluster from the perspective of data nodes. All the data nodes share a similar thermal-efficiency trend and; therefore, we pick one data node as a representative to demonstrate the thermal behaviors of all the data nodes.

Figs. 3.6(a), 3.6(b), and 3.6(c) show the CPU and disk utilization of the nodes when the jobs are running with these three scheduling policies. We observe that the average CPU utilization for the representative nodes with both CPUF and IOF remain considerably high during the execution of the jobs; *tDispatch* manages to reduce the overall average of CPU utilization (see 3.6(c)). CPU utilization is in direct correlation with the power consumption; *tDispatch* is conducive of saving energy. We observe that the disk utilization is considerably reduced with *tDispatch* (Fig:3.6(f)) as opposed to CPUF and IOF (figs. 3.6(d) and 3.6(e)).

Figs. 3.7(a), 3.7(b), and 3.7(c) keep track of the temperatures of all twelve cores in the data node for our three chosen schedulers. We measure the average CPU temperature of each node during the course of the experiments. Instead of calculating the average temperatures across all the data nodes, we focus on the CPU utilization and the peak temperature reached since the three experiments last for a different duration in time. Instead of using average disk utilization, we observe the rise in disk utilization for all the three cases.

Our *tDispatch* is able to bring down the peak temperature of these nodes. The disk temperatures in all the three scheduler cases remain flat mainly because of the relatively low I/O load and short observation time period. Nevertheless, *tDispatch* noticeably reduces the average disk temperatures of CPUF and IOF approximately by 1°C.

3.5 Reduction in Peak Temperature and Energy-efficiency

When it comes to CPU thermal efficiency, *tDispatch* is superior to CPUF and IOF as it can prevent hotspots by demonstrating the maximum difference in temperature that any core had from the average temperature at each iteration is lower and optimizing the peak temperature to reduce the overall power consumption. For example, the peak CPU temperature of the

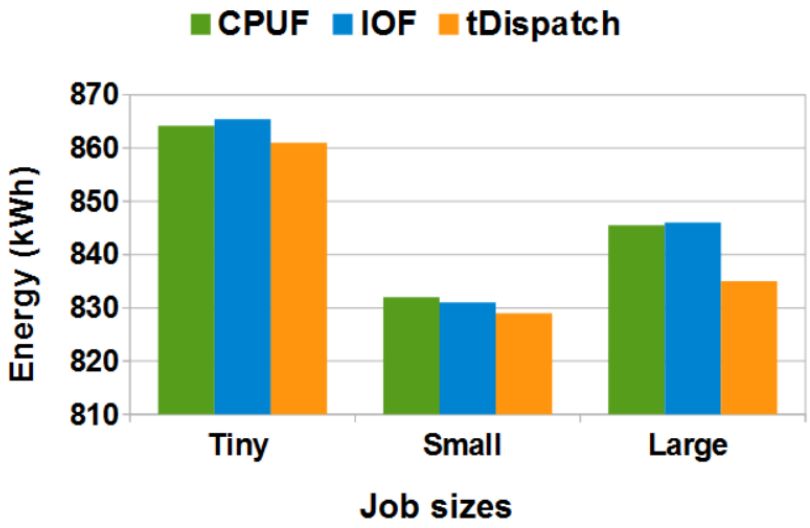


Figure 3.8: Energy consumption of various Hadoop applications running with CPUF, IOF, and *tDispatch* schedulers and three different input sizes. The duration of three experiments are 3 minutes, 6 minutes, and 20 minutes.

representative node is governed by *tDispatch* is 36.2°C (see fig. 3.7(a)), whereas the average CPU temperatures of the same node through CPUF and IOF are 41°C and 34.7°C, respectively.

To estimate the energy efficiency of our scheduler, we used network power distribution unit (PDU) installed in our HPC room to collect the energy consumption by our nodes. An HPC application's power and energy consumption can vary during the lifetime of the application due to the nature of application itself. We use the mean power to compute energy consumption for four different jobs with each of our schedulers. We ran three sets of experiments by varying the input data size for our four chosen applications; the experiments finished at almost the same time and this helps us with a fair comparison among all three schedulers in each size category. Figure 3.8 shows the results of our experiments.

We notice that the energy consumption of very small jobs (labeled as *tiny* in HiBench) is relatively high when compared with other two job sizes. Out of the three cases, the energy consumption is minimal with small size input for the four applications. Regardless of the job size, *tDispatch* is able to mitigate the energy consumption for these jobs. From the energy consumption, we can derive the cooling cost of the data centers running these jobs.

3.6 Summary

Recognizing that improving thermal efficiency of clusters helps in reducing the operational cost of data centers, we developed a thermal-aware job scheduling strategy called *tDispatch* for MapReduce applications running on Hadoop clusters. To facilitate the development of *tDispatch*, we conducted a profiling study of CPU-intensive and I/O-intensive jobs from the perspective of thermal efficiency. We investigated the thermal behaviors of CPU-intensive and I/O-intensive applications running on Hadoop clusters by stress testing data nodes through extensive experiments. Our profiling results show that CPU-intensive and I/O-intensive jobs impose distinctive thermal and performance impacts on multicore processors and hard drives of Hadoop clusters.

After we offered detailed analysis on the thermal profiles of the Hadoop applications, we elaborated the design of the *tDispatch* scheduler, which judiciously dispatches CPU-intensive and I/O-intensive jobs to Hadoop clusters in a way to minimize CPU and disk temperatures.

Our scheduler consists of two key components - *tQueue* and *tDispatch*. The *tQueue* module is responsible for maintaining a CPU-intensive-job and I/O-intensive-job queues. We implemented *tDispatch* to allocate resources of a Hadoop cluster to one of the jobs in these two queues. After a running job finishes its execution, *tDispatch* thermal-efficiently picks another candidate job from the job queues to run on the cluster. Our scheduler alternatively chooses CPU-intensive and I/O-intensive jobs from the two queues to run on a Hadoop cluster, offering ample opportunities for multicore processors and disks to cool down.

We conducted an empirical study to compare the thermal and energy performance of *tDispatch* with two scheduler counterparts, namely, CPUF and IOF. Specifically, we measured the CPU and disk temperatures of the master node and data nodes governed by the three schedulers. At all times, we strive to control the processor temperature as it is directly related to the power consumption in nodes and therefore, reducing the processors temperatures shows reduction in energy consumption by our nodes. Our findings suggest that the schedulers make little impact on the thermal efficiency of the master node. More importantly, our experimental results show

that *tDispatch* is superior to CPUF and IOF in terms of improving thermal efficiency of data nodes.

Chapter 4

Thermal-aware Benchmarking and Modeling

Thermal-aware designs are important to maximize thermal uniformity without degrading the performance of computing servers. This chapter is focused on characterizing thermal profiles of cluster nodes - an integral part of many approaches that address thermal emergencies in data centers. Most existing thermal models make use of CPU utilization to estimate power consumption, which in turn facilitates outlet-temperature predictions. Such utilization-based thermal models may introduce errors due to inaccurate mappings from system utilization to outlet temperatures. To address this concern, we eliminate utilization models as a middleman from the existing thermal models. In this chapter, we propose a thermal model, *tModel*, that projects outlet temperatures from inlet temperatures as well as directly measured multicore temperatures rather than deploying a utilization model. In the first phase of this work, we perform extensive experimentation by varying applications types, their input data sizes, and cluster size. Simultaneously, we collect inlet, outlet, and multicore temperatures of cluster nodes running these diverse Big data applications. The proposed thermal model estimates the outlet air temperature of the nodes to predict cooling costs. We validate the accuracy of our model against data gathered by thermal sensors in our cluster. Our results discussed in this chapter demonstrate that *tModel* estimates outlet temperatures of the cluster nodes with much higher accuracy over CPU-utilization based models. We further show that *tModel* is conducive of estimating the cooling cost of data centers using the predicted outlet temperatures.

We start this chapter by delineating a modeling framework (Sec. 4.2). Then, we develop a baseline thermal model to estimate outlet temperatures of Hadoop clusters (Sec. 4.2.1). The rest of this chapter is organized as follows. Section 4.3 introduces our proposed model's design

and an essential modeling parameter, K. Section 4.4 provides the thermal profiling evaluations. Finally, Section 4.5 concludes our study by summarizing our findings.

4.1 Towards Thermal Modeling

Thermal profile of a data center has a significant impact on its cooling cost. As mentioned earlier, thermal models are used to characterize the thermal profile of data centers. Number of thermal models have been proposed in the past that calculate the inlet temperatures of the nodes and their impact on the thermal profile and cooling cost of data centers. Most of the thermal models rely on processor utilizations to predict outlet temperatures of cluster nodes. In this study, we specifically investigate the impact of MapReduce workloads on the outlet temperature of the nodes. We propose a thermal model to predict outlet air temperatures of the nodes housed in a data center. With our thermal model in place and workload characterization, we estimate the cooling costs for various Hadoop applications.

There is a growing demand to build thermal-prediction models for high-performance clusters. We design a systematic approach to developing thermal models for Hadoop applications deployed in data centers. The thermal model proposed in this study immediately benefits usage cases. First, outlet temperatures of server blades are applied to estimate cooling costs [26], which depends on outlet temperatures and heat dissipated by the servers. Second, temperatures captured in our thermal model paves a way to investigate heat re-circulation model [68] [21], which is affected by cross-interference coefficients.

There are three contributions in this work. First, we conduct a thermal profiling study of a cluster on which various Hadoop applications are investigated. When the applications process various data size under different number of nodes, we profile our server nodes to monitor multicore CPU temperatures. Second, we build a thermal model to estimate temperatures using inlet temperatures and CPU core temperatures. Third, as a use case of our model, we extend the model to predict the cooling cost of a Hadoop cluster.

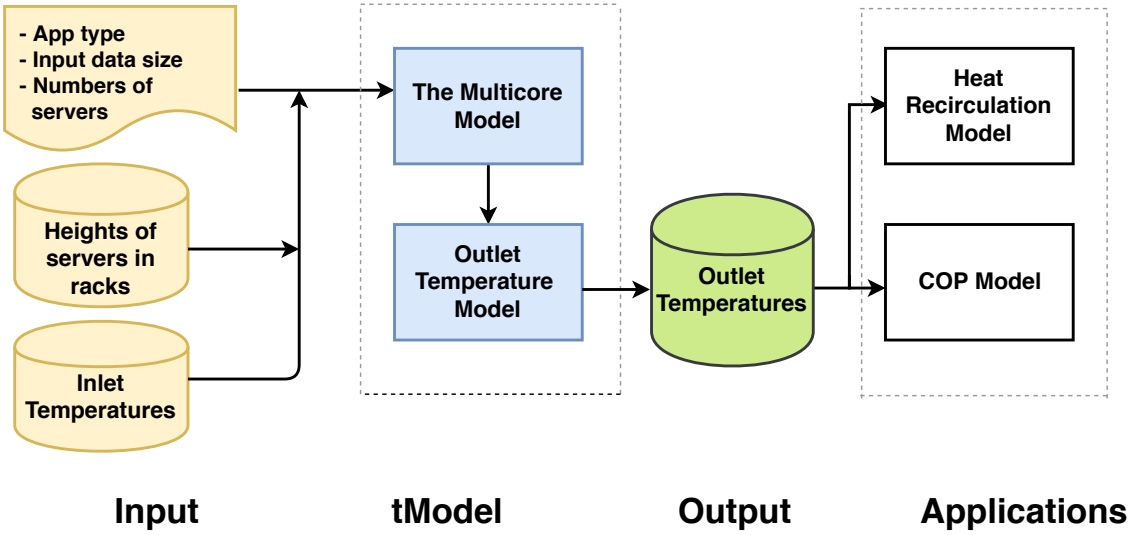


Figure 4.1: The system framework of our thermal models.

4.2 The Modeling Framework

Fig. 4.1 outlines our thermal-modeling framework, which consists of two key components: the outlet-temperature model and the multicore model.

The input data of *tModel* includes application profiles (e.g., application type, data size), heights of servers in racks, and inlet temperatures. It is worth noting that all these input parameters are directly fed into the multicore model which generates temperatures of multicore processors (Sec. 4.3.2). The multicore-processor temperatures are assimilated into the outlet temperature model (Sec. 4.3) to estimate outlet temperatures of any given computing node. Thus, the outlet-temperature model builds up the correlations between inlet and outlet temperatures by incorporating multicore temperatures residing in computing nodes.

Recall that tModel relies on inlet temperatures monitored by sensors, which tend be expensive for large-scale data centers. To reduce such hefty costs, we only install temperature sensors on a few selected blade servers (a.k.a., nodes) equidistant from each other in the rack (e.g., 4 out of 12); one sensor each on the topmost and bottom-most node and two sensors in between these nodes at different heights. We also may use the inlet temperatures measured on one rack to speculate inlet temperatures of nodes mounted on another rack.

Outlet temperatures produced by tModel benefits applications like cooling cost models (i.e., the COP model [11]) and heat recirculation models. The COP model computes cooling costs by taking into account outlet temperatures offered by the outlet-temperature model in tModel. The main strengths of the tModel framework are three-fold. First, temperatures of multicore processors are modeled as a function of system configurations and applications' resource access patterns. Second, a thermal model characterizes the relationship among inlet and outlet temperatures as well as multicore processor temperatures. Third, tModel opens an avenue for data center designers to investigate heat recirculation and cooling costs.

4.2.1 A Baseline Thermal Model

To predict outlet temperatures for nodes in a Hadoop cluster, we construct a baseline thermal model speculating outlet temperatures derived from inlet temperatures, multicore temperatures, input data size, and height of servers placed in racks. In a data center equipped with cooling systems, the inlet temperatures of the servers are affected by the supplied room temperature mixed with recirculated hot air. Given a set of nodes, we denote T_i^{in} as the inlet temperature of the ith node. It is worth noting that T_i^{in} can be directly and accurately monitored by a temperature sensor. Alternatively, T_i^{in} can be estimated using monitored temperatures from neighboring servers.

Our baseline thermal model is inspired by the *T** thermal model proposed by Kaushik *et al.* [52]. The T*-model predicts servers' outlet temperatures to estimate the cooling cost for data centers.

Let $T_{i,t}^{out}$ be the outlet temperature of the ith server at a given time *t*. $T_{i,t}^{out}$ depends on (a) server i's power consumption $P_{i,t}$ at time t, (b) server temperature $T_{i,t}$, and (c) heat exchange rate θ_i. Thus, we have

$$T_{i,t}^{out} = T_{i,t} - \frac{P_{i,t}}{\theta_i}. \tag{4.1}$$

For simplicity, we consider in this study statistical temperature measures rather than temperature in a specific time. After removing t from (4.1), we obtain

$$T_i^{out} = T_i - \frac{P_i}{\theta_i}, \tag{4.2}$$

where P_i is server i's average power consumption in a given monitoring period.

According to the law of energy conservation and the fact that dissipation of power is mostly in the form of heat, the relation between power consumption P_i of a server node i and its inlet/outlet temperatures is given as [26]

$$P_i = \rho f_i C_p (T_i^{out} - T_i^{in}) \tag{4.3}$$

where, C_p is the specific heat of the air, f_i is the flow rate of the air, and ρ is the air density.

Substituting (4.3) for the power consumption on the right-hand side of (4.2), we obtain the outlet temperature T_i^{out} of server i as

$$T_i^{out} = \frac{\theta_i * T_i + \rho f_i C_p * T_i^{in}}{\theta_i + \rho f_i C_p} \tag{4.4}$$

Let $\rho f_i C_p$ be α_i; outlet temperature modeled in (4.4) can be rewritten as

$$T_i^{out} = \frac{\frac{\theta_i}{\alpha_i} * T_i + T_i^{in}}{\frac{\theta_i}{\alpha_i} + 1} \tag{4.5}$$

To further simplify the above expression, we denote ratio θ_i/α_i as β_i. The outlet temperature model (4.5) can be expressed in form of parameters θ and α. Thus, we have

$$T_i^{out} = \frac{\beta_i * T_i + T_i^{in}}{\beta_i + 1} \tag{4.6}$$

The above thermal model (4.6) indicates that server i's outlet temperature can be derived using its inlet temperature T_i^{in} as well as system parameter β_i, which is affected by the server's workload as well as airflow (i.e., heat recirculation). Given a server (e.g., the ith server) in a

cluster, we can estimate the server's outlet temperatures from the corresponding inlet temperature and its critical parameter β_i. This observation motivates us to focus on a way of modeling parameter β_i as

$$\beta_i = \frac{T_i^{out} - T_i^{in}}{T_i - T_i^{out}} \tag{4.7}$$

We apply the non-linear regression model to investigate the β parameter in (4.7), which is a driving force behind the baseline thermal model. To model parameter β for each server running a Hadoop application, we conduct a profiling analysis to study the correlations between β and thermal data (i.e., inlet/outlet temperatures, multicore temperatures) when a relatively small data is processed by a target application. One hypothesis is that parameter β, affected by application access patterns and server locations (e.g., height), is independent of datasize for data-intensive applications.

4.3 tModel: Modeling Outlet Temperatures

In this section, we develop a thermal model that aims at predicting outlet temperatures by considering the impacts of inlet air temperature and CPU core temperatures on the outlet temperatures (Sec. 4.3.1). Next, we discuss the role of our multicore model in predicting outlet air temperatures (Sec. 4.3.2). We also demonstrate a sample usage of *tModel* in Sec. 4.3.3).

4.3.1 A Simple Yet Efficient Model

The model proposed in Sec. 4.2.1 relies on power consumption and other factors. In this part of the study, we intend to construct a simple yet effective thermal model to predict outlet temperatures without explicitly using power consumption. We show in Sec. 4.4.3 that the simplified model described in this section is more accurate than the complicated baseline model.

The goal of our simplified thermal model is to predict outlet temperatures for Hadoop cluster nodes from (a) inlet temperatures and (b) multicore temperatures without relying on any other parameters. Most existing thermal models make use of CPU utilization to estimate power consumption, which in turn facilitates outlet-temperature predictions. Such utilization-based thermal models may introduce errors due to inaccurate mappings from system utilization

to outlet temperatures. Rather than deploying a utilization model, our tModel projects outlet temperatures from inlet temperatures as well as directly measured multicore temperatures by leveraging on-processor sensors. Again, to cut cost of maintaining an excessive number of sensors monitoring inlet temperatures, we mount sensors on a small percentage of nodes; inlet temperatures of the other nodes can be modeled from their neighboring nodes due to the fact that the heat measured at each sensor is the combination of heat generated by the workload and the heat from the ambient air at the server's inlet air grill. Thereby, we also eliminate another middleman, the ambient temperature.

The outlet temperature T_i^{out} of the ith node is modeled from its inlet temperature T_i^{in} and the average multicore temperature of the node. Thus, we express temperature T_i^{out} as

$$T_i^{out} = T_i^{in} + K_i \times \frac{\sum_{j=1}^{m} T_{ij}^{core}}{m}. \tag{4.8}$$

where T_{ij} is the j core in node i, and $\frac{\sum_{j=1}^{m} T_{ij}}{m}$ is the average core temperature in node i. Here, n denotes the total number of active nodes in the cluster; m is the number of cores in each node. Let T_i^{core} be the average core temperature. Thus, we have

$$T_i^{core} = \frac{\sum_{j=1}^{m} T_{ij}^{core}}{m}. \tag{4.9}$$

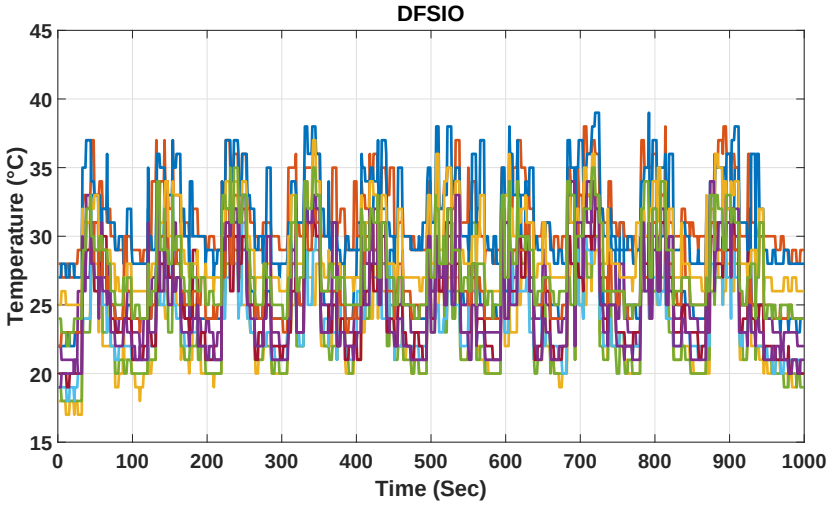
Applying (4.9) to (4.8), we simply temperature T_i^{out} as

$$T_i^{out} = T_i^{in} + K_i \times T_i^{core}. \tag{4.10}$$

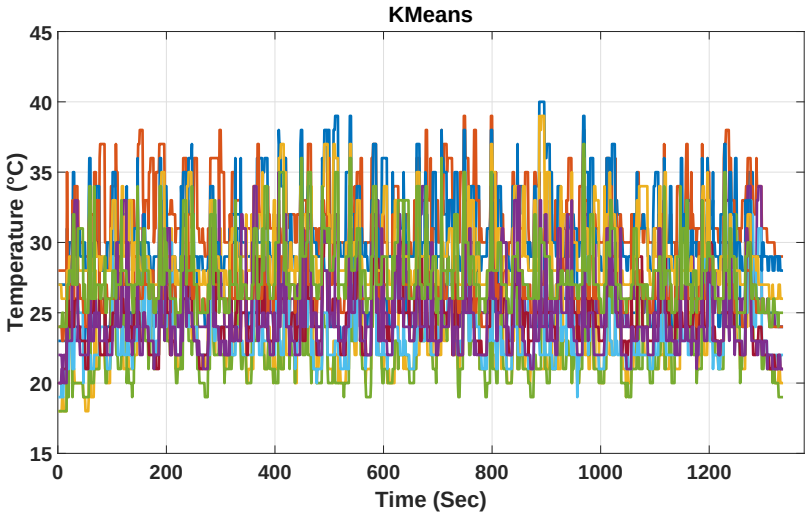
K is a modeling parameter that is determined through benchmarking. This *K* is employed to predict outlet temperature for determining the overall cooling costs for data centers. Section 4.4.3 discusses the methodology for obtaining *K* for various benchmarking applications.

4.3.2 The Multicore Model

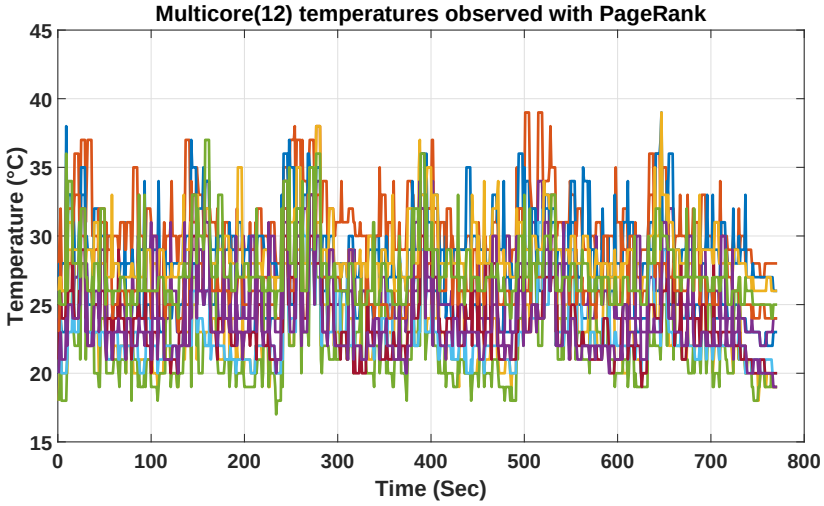
Our proposed thermal model, tModel, constitutes two essential components - a multicore model and an outlet temperature model. The multicore model is responsible for measuring the average



(a)



(b)



(c)

Figure 4.2: CPU core temperatures for three benchmarking applications - DFSIO, KMeans, and PageRank - running on one server node.

core temperature for a running job. For predicting core temperatures, our multicore model takes into account application types, inlet temperatures, heights of computing nodes in a rack, input data size, and number of active data nodes. Recall that we rule out ambient temperatures as a middleman, because the inlet temperatures of the nodes can be modeled from their neighboring nodes. We factor in server heights, which may be used to project inlet air temperatures in case when there is no sensor mounted in front of the servers.

We update the multicore model to include workload information as a proxy for the amount of heat injected into a Hadoop system, because various applications may exhibit different resource access patterns (e.g., CPU-intensive, I/O-intensive, memory-intensive, or other types) and owing to their nature. The thermal impacts of different applications on various server components would be diverse as well. It is indispensable to consider data sizes along with the application behaviors as large data sizes would have a direct impact on CPU and disk temperatures. Disks take a considerable amount of time to heat up, and therefore disk temperatures are not addressed in tModel.

Let T_i^{in} be the inlet air temperature of ith node, D be the input size of application A. Multicore temperature T^{core} denotes the average temperature of all cores of ith node; temperature T^{core} depends on T_i^{in}, D, and A. Thus, the multicore temperature model can be expressed as:

$$T_i^{core}(D, A) = f(T_i^{in}, D, A). \tag{4.11}$$

Our multicore model is motivated by the resource-access patterns of big-data applications, which often exhibit iterative characteristics. To capture CPU thermal behavior of parallel and big-data applications, we run representative applications on a single server node. Based on the benchmarking results, we notice some common characteristics in the thermal behavior of the chosen applications. The thermal behavior of multicore processors running an application can be modeled by a series of working sets, each of which is comprised of multiple intervals sharing a very similar behavior (Fig. 4.2). We also refer to this multicore model as the *working-set* model. It is worth noting that different applications may have an array of working sets of varied lengths.

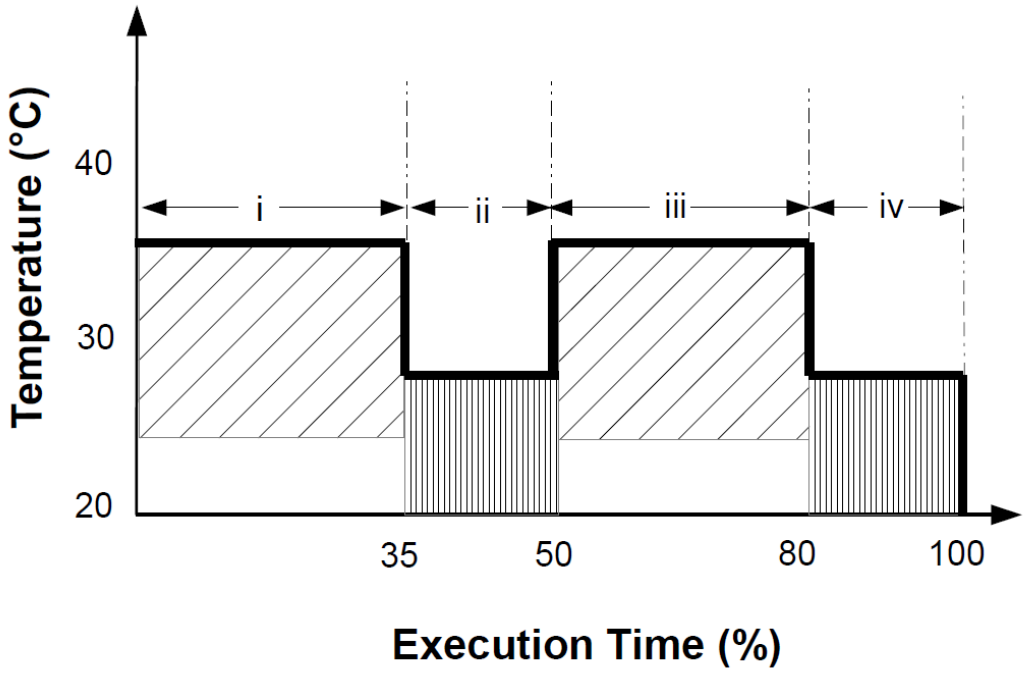


Figure 4.3: Working sets in the multicore model capture thermal patterns the DFSIO application running on a Hadoop cluster.

We make use of the working-set model to represent an application's thermal signature. Suppose an application has a total of m working sets, which are denoted as a m-dimensional vector $\vec{\phi} = [\phi_1, \phi_2, ..., \phi_m]$. The ith working set phi_i is a vector $\phi_i = (\pi_i, \lambda_i, \mu_i, \gamma_i)$, where π_i is the number of phases repeatedly executed in the working set, $lamdba_i$ and μ_i are the maximum and minimum core temperatures in the working set, and γ_i is the length of each phase.

The multicore model captures the phase signatures of each working set to present the average temperature of the working set, where temperature patterns repeat during the running time of an application. The average CPU temperatures obtained from the multicore model are fed into the outlet temperature model to estimate outlet air temperature.

The multicore model elaborated in this section is focused on one application at a time. Nevertheless, the model is applicable for multiple applications currently running on a time-sharing environment. In such a case, we may build an aggregated multicore model by combining the working sets of individual Hadoop applications with probability distribution functions, thereby resembling the mixture of the applications.

4.3.3 A Sample Usage

Now we show how to apply the multicore model (Sec. 4.3.2) through a sample usage, where the well-known *COP* model (a.k.a., the Coefficient Of Performance model [11]) is incorporated. COP is defined as the ratio of heat removed to the energy cost of the cooling system for heat removal. A large COP value implies that a data center's energy efficiency is high and vice versa.

In this usage case, we make use of our model coupled with the COP model to estimate the cooling cost of a data center. After our tModel sheds light on outlet temperatures, we plug the temperature into the following equation (Eq. 4.12) to obtain a value of the COP parameter.

$$COP(T) \ = \ 0.0068 * T^2 + 0.0008 * T + 0.458. \tag{4.12}$$

Let P_{AC} be cooling cost in terms of power consumed by the cooling system. We calculate cooling power P_{AC} from (1) the CRAC's supply temperature T and (2) power consumption P_C of the computing infrastructure. Thus, we have

$$P_{AC} = \frac{P_C}{COP(T)} \tag{4.13}$$

where the cooling cost is inversely proportional to the COP value. It is worth noting that P_{AC} is a metric quantifying the data center's energy efficiency.

4.4 Experimental Evaluation

We implemented *tModel* on a testbed of one traditional rack consisting of 14 SuperMicro Model 825-7 servers with Intel (R) Xeon (R) X5650@2.67GHz processors (Fig. 4.4). The computer rack is located in Auburn University's High-Performance Computing Lab. The cooling unit supplies cool air from the ceiling and the room temperature is set to 63F. We allocate 16 GB RAM for Hadoop leaving 8 GB for operating system processors. Each map is allowed 4 GB RAM and each reducer is allocated a minimum of 8 GB RAM. This way, Hadoop can run up to 4 mappers and 2 reducers per node. To measure processor temperatures, we use a Linux utility

program called *lm-sensors* [69]. We deploy APC environment sensors placed to the back and front of the cluster nodes. We measure execution time in terms of elapsed wall clock time. It is worth mentioning that throughout this study, we measure actual execution times, CPU temperatures, inlet and outlet temperatures of the aforementioned MapReduce applications without relying on any simulation data. Further, our evidence shows that it takes approximately 30 minutes to heat up a disk under heavy I/O workload conditions. To minimize impacts of one experiment on its subsequent one, we ensured that the minimal interval between two subsequent experiments is at least one hour.

(a) (b)

Figure 4.4: Environmental sensors attached to the front and back of cluster nodes placed in a traditional rack.

In the following section, we evaluate the effectiveness of the proposed thermal model, *tModel*, in accurately estimating the outlet temperatures of the servers. The purpose of profiling is to understand the trends among inlet and multicore temperatures which can be used in calculating K used in our modeling equation. This K is employed to predict outlet temperature for determining the overall cooling costs for data centers.

Application	DFSIO	PageRank	KMeans
Parameters	No. of files read, file size MB, No. of files written, file size MB	Pages, No. of iterations, block, block width	No. of clusters, dimensions, No. of samples, samples per input, maximum iterations, k, converge distance
Small	32, 10, 32, 10	5000, 3, 0, 16	5, 20, 3000000, 600000, 5, 10, 0.5
Large	64, 10, 64,10	50000, 3, 0, 16	5, 20, 20000000, 4000000, 5, 10, 0.5
Huge	256, 100, 256, 100	5000000,3,0,16	5, 20, 100000000, 20000000,5 10, 0.5
Gigantic	2048, 1000, 2048, 1000	50000000, 3, 0, 16	5, 20, 2*108, 4*106, 5, 10, 0.5

Table 4.1: Summary of input parameters for three benchmarks, DFSIO, PageRank, and KMeans in HiBench, including their data size configurations and input parameters.

4.4.1 Profiling

To facilitate the development of our *tModel*, we benchmark three Big Data applications, namely *DFSIO*, *KMeans*, and *PageRank*. KMeans is a well-known clustering algorithm for knowledge discovery and data mining. KMeans performs clustering on a given set of points by running algorithms to minimize Euclidean distances among the points in every group. Since for each input, the nearest centroid has to be calculated and the minimum distance between a data point to the cluster centroid needs to be searched, KMeans is computation-heavy in nature. DFSIO, on the other hand, attempts to measure HDFS' capacity for reading and writing bulk data. It has been used to measure the average I/O and throughput of a cluster, and therefore is classified as I/O intensive one. PageRank is comprised of both CPU-intensive and I/O-intensive phases owing to its iterative nature. PageRank consists of a large number of computations and spends some I/O-waiting CPU time due to large input data sizes. The MapReduce implementation and various input data sizes for these applications can be found in the *HiBench* benchmarking suite [67]. Developed by the Intel Labs, *HiBench* provides a collection of real-world Hadoop applications. HiBench enables a clear understanding of benchmark characteristics by running an actual Hadoop-implemented code of the applications on our cluster as opposed to other trace-based workloads. Table 4.1 summarizes the data size configurations and input parameters for three chosen benchmarks, DFSIO, PageRank, and KMeans.

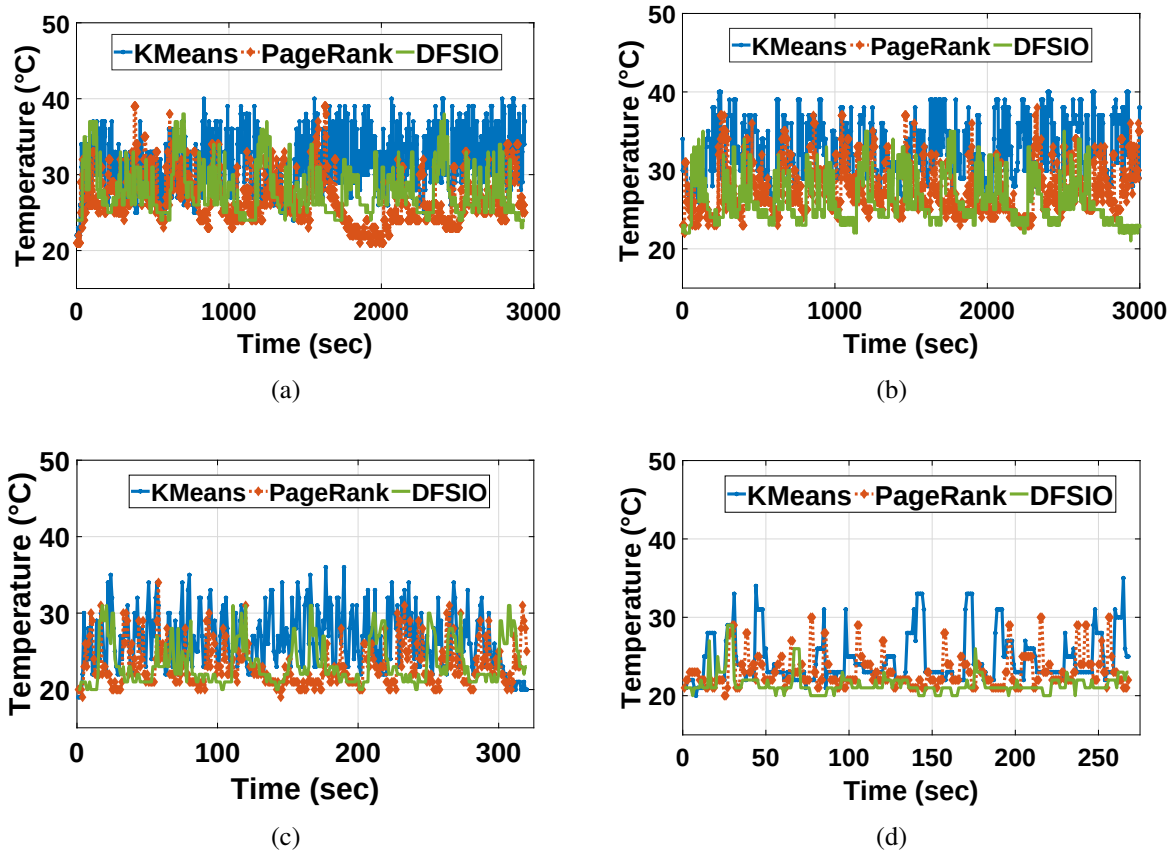


Figure 4.5: Average multicore processor temperatures of one worker node executing KMeans, PageRank, and DFSIO benchmarking applications on a cluster comprising of 4, 6, 8, and 12 nodes respectively.

We conduct extensive experiments in three phases to estimate an accurate value of K proposed in our model. In Phase I, we submit a large-sized input for these benchmarks running on our cluster and observe the thermal behavior of our cluster nodes. In Phase II, we pick a representative benchmark to process different input data sizes available in the *HiBench* benchmark suite. In Phase III, keeping the same representative benchmark and the input data size, we vary the number of running nodes (i.e., 4, 6, 8, and 12) to further enhance of understanding of the variability in K of the applications on the cluster.

In each phase, we measure CPU core temperatures, inlet and outlet air temperatures of the master and data nodes in the Hadoop cluster every 10 seconds. The real-time thermal data allows us to (1) quantitatively assess our thermal model and (2) perform validations with a high degree of confidence. Fig. 4.5 shows the multicore processor temperatures of one representative cluster node running KMeans, DFSIO, and PageRank applications. The four sub-figures in

Fig. 4.5 show thermal results for four different cluster sizes made up of (a) 4 nodes, (b) 6 nodes, (c) 8 nodes and (d) 12 nodes. Thanks to the fair sharing of data by Hadoop default scheduler among all the cores, all the cores share the same temperature trends; and therefore, we choose to present only one out of 12 cores in these figures.

It must be noted that the execution time needed for processing a given data set with different number of nodes takes different time; we present the results keeping the execution of shortest job in each of the three cases. It is worth noting that the peak CPU temperature for KMeans is much higher than its counterparts - PageRank and DFSIO - owing to the nature of KMeans as being a CPU-intensive one. The trend is similar given a fixed input data size and different number of nodes in a cluster. This type of profiling helps our multicore model in predicting an average multicore temperature, which is fed into the outlet-temperature module of *tModel* (Fig. 4.1).

4.4.2 Issues with Power-based Models

To deal with thermal emergencies, a growing number of predictive techniques have been developed in the past. These techniques utilize CPU utilization and power consumption of server nodes in order to estimate outlet air temperatures. Our approach, on the other hand, eliminates these two parameters, which are applied to predict outlet temperatures with the insight of multicore temperatures. In this subsection, we provide insights on the background of existing power-based models, which are summarized as follows. Further, the baseline thermal model (a.k.a, β-model) discussed in Section 4.2.1 has also been inspired by the following idea.

Heath *et al.* discussed the law of conservation of energy and Newton's law of cooling in their study [70]. According to concepts in physics, the temperature of an object is an indicator of the object's internal energy; the object's temperature is affected by heat exchanging between itself and it's environment. Cluster nodes draw power from power distribution units and convert the power into heat. This heat enables the functioning of the node's various components including processors, RAM, storage devices, graphic cards, motherboard, fans and networking interconnects. Processors consume the most power, followed by main memory. As per thermodynamics, the heat (Q) produced by these components correspond is equivalent to their energy

consumption, which is defined as:

$$Q_{component} = P(\mu) \times t. \tag{4.14}$$

where t is time interval and $P(\mu)$ represents the average power consumed by the component over a given interval of time; this power is a function of its utilization.

Heath *et al.* formulated a linear relationship between component utilization and power consumption to estimate the real power consumed by a component of a server. This relationship entails the power consumption of a component at idle state (P_{base} = 0%) and fully utilized state (P_{max} = 100%); these two hardware-specific parameters are generally provided by component manufacturers.

$$P(\mu) = P_{base} + \mu \times (P_{max} - P_{base}). \tag{4.15}$$

In specific, for processors equipped with a total of n cores, the average power consumption is estimated using the following formula [71]:

$$P_n = (P_{max} - P_{idle}) \times \frac{n}{100} + P_{idle}. \tag{4.16}$$

It is arguably true that estimating the power consumption of processors using utilization information may not be adequate for modern processors due to three reasons. First, the power estimated in Eq. 4.16 with the aid of CPU utilization introduces prediction errors at a rate of 5%-7% [71]. Second, power is consumed not only by the processors, but also by memory, disk, and motherboard in non-trivial amounts. It is unfair and inaccurate to focus on CPUs while overlooking other components (e.g., memory and disks) that are significant consumers of power. Third, server utilization may change instantaneously, but it takes time for temperature distribution to adjust. Hence, involving multicore temperatures in our thermal model tends to be more accurate than the existing power-based thermal models. We show a quantitative comparison between our model and the existing β-model (Sec. 4.2.1) in the next section (Sec. 4.4.3).

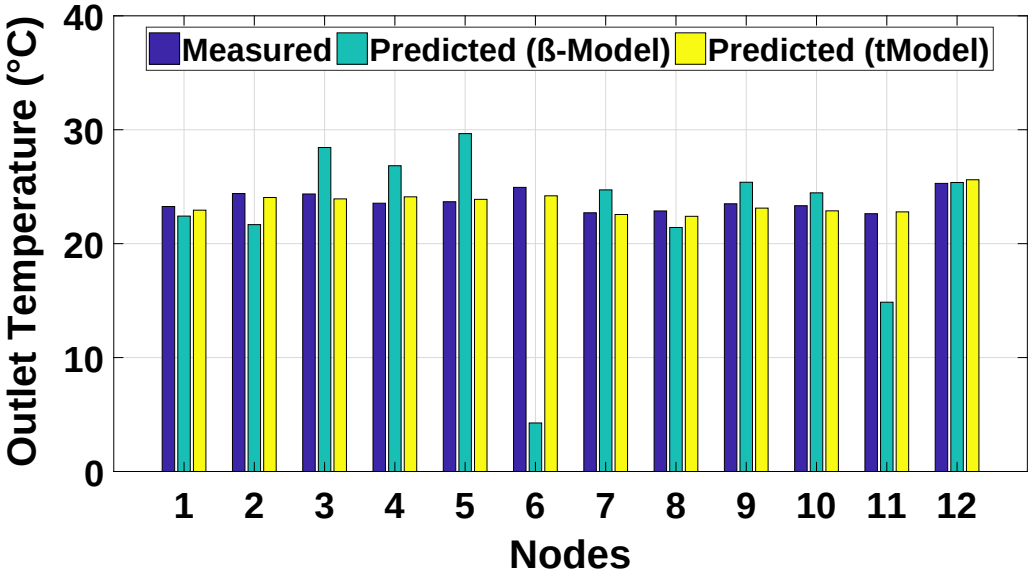


Figure 4.6: Performance comparisons between tModel and its alternative β-model in terms of outlet-temperature prediction. The *measured* temperature is obtained using physical sensors attached to each node in the rack.s The benchmark used in this group of experiments is *KMeans*.

4.4.3 Improving Accuracy

Through our preliminary profiling experiments (Sec. 4.4.1), we observe and analyze the value of K_i introduced earlier in Section 4.3 using various applications and data sizes. This estimation is important in terms of costs of temperature sensors. The tModel has to be tailored for each application, because applications may have diverse behaviors and access patterns. For the same reason, our system framework (Fig. 4.1) provides the application type as an input to the multicore model. In this section, we used one of our representative applications, KMeans, to fine-tune and validate the tModel. It is worth noting that our modeling methodology is very generic. We believe that a diverse set of applications, ranging from CPU-intensive to I/O-intensive to memory-intensive, can be tuned in the same way. Because application type is an important determinant of core temperatures in tModel, different type of application would have to be profiled separately. Our findings suggest that outlet temperatures vary with application types.

We begin by profiling every core of the processors installed in each server of our cluster. Then, we employ a non-linear regression model to estimate the value of K_i, which drives the estimation of outlet temperatures. For profiling K_i initially, we use outlet temperatures to train

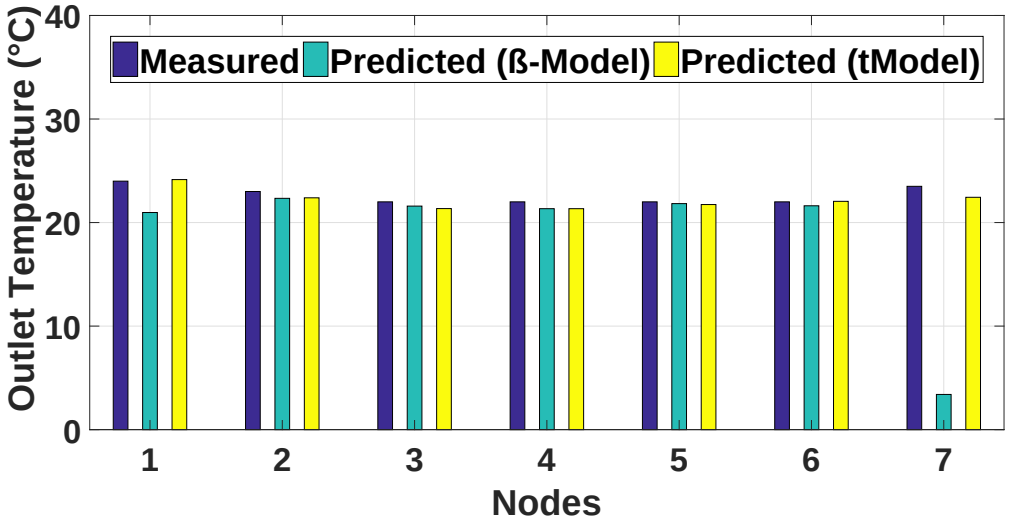


Figure 4.7: Performance comparisons between tModel and its alternative β-model in terms of outlet-temperature prediction. The benchmark used in this group of experiments is *PageRank*.

the model (i.e., obtain the K_i value). Next, we make use of the K_i value to predict the future outlet temperatures.

Fig. 4.6 shows the comparisons of our *tModel* and the existing β-model (Sec. 4.2.1). Fig. 4.6 reveals outlet temperatures predicted by the two models as well as the measured ones for model validation purpose. We use the average power consumption and K values generated by running one application to predict the outlet temperatures for another application without changing the input data sizes. The experimental results indicate that *tModel* is conducive to estimating outlet temperatures with a higher accuracy than the β model for KMeans, a CPU-intensive application. In another experiment, we collected profiling data in terms of *K*- and β-values on twelve worker nodes and used these values to predict outlet temperatures of PageRank and DFSIO running on a smaller number of the total worker nodes. Our *tModel* performs with higher accuracy than β-model; although β-model performs prediction of outlet temperatures much better with PageRank and DFSIO when compared to its performance in previous case of KMeans.

The outlet temperatures predicted by tModel are highly accurate when compared with the temperatures recorded by the installed sensors (Fig. 4.6 and Fig. 4.7). The training parameter, *K*, needs to be trained for different applications for the first execution of a novel application and later, can be used for predicting outlet temperatures in the most accurate manner without

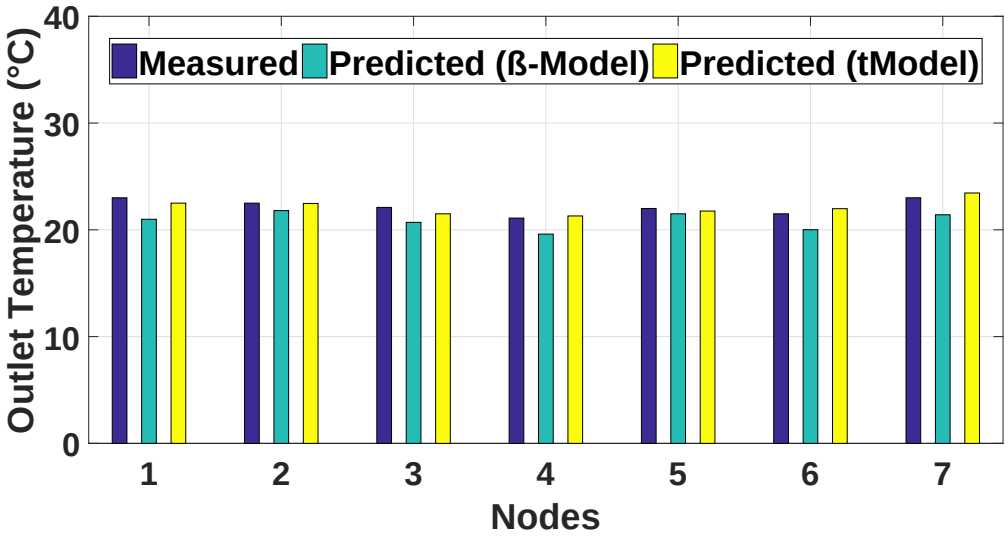


Figure 4.8: Performance comparisons between tModel and its alternative β-model in terms of outlet-temperature prediction. The benchmark used in this group of experiments is *DFSIO*.

relying on temperature sensors. The reason for more accuracy of β-model in PageRank than KMeans owes to the nature of the applications; KMeans is a CPU-intensive application and its thermal impacts on CPU and disk are much more intense than PageRank, which has both CPU-intensive and I/O-intensive phases (Fig. 4.5). tModel benefits CPU-intensive applications more than I/O-intensive applications (Fig. 4.8) since CPU-intensive applications increase the CPU-temperatures at a much faster rate than an I/O-intensive one; I/O-intensive application performs disk I/O frequently and thus, minimizing their impacts on CPU temperatures. We must acknowledge that CPU temperatures are fed into the multicore model for the estimating of outlet temperatures and I/O-intensive applications impose minimal thermal impacts on CPU temperatures.

The experimental results shown in the previous subsection indicate that outlet temperatures are affected by the inlet as well as the multicore temperatures and these vary greatly among nodes. Our *tModel* is conducive of predicting outlet temperatures with high accuracy without relying on the power consumed by the running nodes. Further, the average error of our model stays below 5% with all three chosen applications. On the other hand, the β-model model fails in accurately predicting outlet temperatures in majority of the cases.

4.5 Summary

We proposed a thermal model called *tModel* aiming to predict outlet temperatures of servers in Hadoop clusters running Big Data applications. tModel eliminates a pressing need to estimate power consumed by Hadoop nodes prior to speculating their outlet temperatures. tModel makes use of a vital parameter K_i to resemble correlation between outlet and inlet temperatures of the ith node. Parameter K_i is obtained through profiling in the sampling phase. We apply the procured values for each individual core *i* to predict the thermal behaviors of Big Data applications processing various data sizes. tModel is more accurate than traditional models that rely on power consumption and CPU utilization, because modeling core temperatures from CPU utilization is likely to introduce prediction errors.

Our thermal model offers two compelling benefits. First, *tModel* makes it possible to cut back thermal monitoring cost by immensely reducing the number of physical sensors needed for large-scale clusters. Monitoring temperatures is a vital issue for safely operating data centers; however, it is prohibitively expensive to acquire and set up a large number of sensors in a large data center. Second, *tModel* enables data center designers to evaluate thermal management strategies during the center design phase.

We made use of KMeans, DFSIO, and PageRank as three benchmarks to validate our proposed tModel. Apart from investigating the KMeans, DFSIO, and PageRank benchmarks, we intend to study tModel in the context of streaming applications. Prior to such a model validation process, we will profile the thermal behaviors of an array of streaming benchmarks running on a Spark cluster.

We also compared our tModel with an existing β model, which is more complicated than tModel. The experimental results demonstrate that tModel improves the accuracy of β-model with an average of 3% in KMeans application. This trend was similar with other applications and data sizes. Further, CPU-intensive applications benefit more from tModel than I/O-intensive applications.

Chapter 5

Energy-Efficiency in MapReduce using Approximate Computing

There is an ongoing search for finding energy-efficient solutions in multi-core computing platforms. Approximate computing is one such solution leveraging the forgiving nature of applications to improve the energy efficiency at different layers of the computing platform ranging from applications to hardware. In this part of the dissertation, we are interested in understanding the benefits of approximate computing in the realm of Apache Hadoop [72] and its applications.

A few mechanisms for introducing approximation in programming models include sampling input data, skipping selective computations, relaxing synchronization, and user-defined quality-levels [55] [16]. We believe that it is straightforward to apply the aforementioned mechanisms to conserve energy in Hadoop clusters as well. The emerging trend of approximate computing motivates us to systematically investigate thermal profiling of approximate computing strategies in this dissertation research. In particular, we design a thermal-aware approximate computing framework called *tHadoop2*, which is an extension of *tHadoop* proposed by Chavan *et al.* [73] in the year 2017.

This chapter is organized as follows. We summarize some preliminaries in Section 5.1. We propose a framework to apply approximation strategies in Hadoop applications which is discussed in Section 5.2. A case study on a MapReduce application is presented in Section 5.3. Finally, we conclude this chapter in Section 5.4.

5.1 Preliminaries

There is an array of application domains that can tolerate loss of accuracy when the resources required to provide a precise answer are either expensive or unavailable. Owing to varying characteristics of applications, approximation strategies are determined on a per-application basis when computing resources are limited [74] [59]. For instance, applications in domains like machine learning, image recognition, social-network based recommendations, sensor data analysis already assimilate imprecision in their algorithm designs [17]. Approximate Computing has potential to benefit a wide range of applications and frameworks, including but not limited to, data analytics, scientific computing, multimedia and signal processing, machine learning, and MapReduce programming [54].

The U.S. Department of Energy (DOE) has identified energy-efficiency as one of the biggest challenges of the exascale computing era [75]. The DOE has also set a goal of 20 MW for exascale (10^{18} flops) supercomputers. Improving resource utilization and energy efficiency at all levels of a system hierarchy is imperative to achieve the goals of exascale computing. Further, research has established that energy-efficient applications have the potential to enhance the energy efficiency in cluster systems [5].

Some of the state-of-the-art works in resource management for thermal-aware cluster computing are categorized in Table 5.1. TIGER performs thermal-aware file placement using the disk utilization of worker nodes and heat-circulation among them [76]. *tDispatch* is a thermal-aware job scheduler that can be deployed on top of Hadoop's default scheduler and it uses applications' characteristics and resource usage pattern for its scheduling decisions [77]. Another thermal-aware job scheduler, *tHadoop*, reduces the power consumption of nodes by capping the number of tasks that can be executed on nodes. It achieves the same by minimizing the heat re-circulation using a cross interference matrix and calculates an appropriate number of tasks that must be run on a given cluster node [73]. Approximate computing techniques exploit the inherent resilience in applications to realize improvements in energy consumption and their execution time without paying attention to thermal impacts on system resources [16] [15].

In this part of the dissertation, we seek a novel approach for enhancing the energy-efficiency of Hadoop clusters through approximate computing techniques and thermal-awareness of the cluster nodes. The next section delineates the design of a thermal-aware computing framework called *tHadoop2*, which incorporates approximate computing to conserve energy in Hadoop clusters.

Technologies	**MapReduce**	**Error Bounds**	**File Placement**	**Thermal-awareness**	**Tune number of tasks**	**Scalability**
Approximate Computing [55] [15]	✗	✓	✗	✗	✗	✗
TIGER [73]	✗	✗	✓	✓	✗	✗
tDispatch [77]	✓	✗	✗	✓	✓	✗
tHadoop [73]	✓	✗	✓	✓	✗	✗
tHadoop2 (ours)	✓	✓	✓	✓	✓	✓

Table 5.1: Comparisons between our *tHadoop2* and the existing energy-efficient techniques.

5.2 Framework Design

We start this section by explaining the *tHadoop2* system framework (see Section 5.2.1), which offers a high-level design from the perspective of approximation in Hadoop applications. Next, we shed light on the responsibilities of various modules in *tHadoop2* (see Section 5.2.2).

5.2.1 The tHadoop2 Platform

In this section, we propose *tHadoop2*, a thermal-aware framework that incorporates approximation techniques for MapReduce applications. In what follows, we outline the interfacing between *tHadoop* and our proposed framework, *tHadoop2*, followed by the descriptions of the various modules in *tHadoop2*.

Fig. 5.1 shows the design of our framework. *tHadoop2* is an incremental version of *tHadoop* - a thermal-aware scheduling framework minimizing the heat re-circulation through

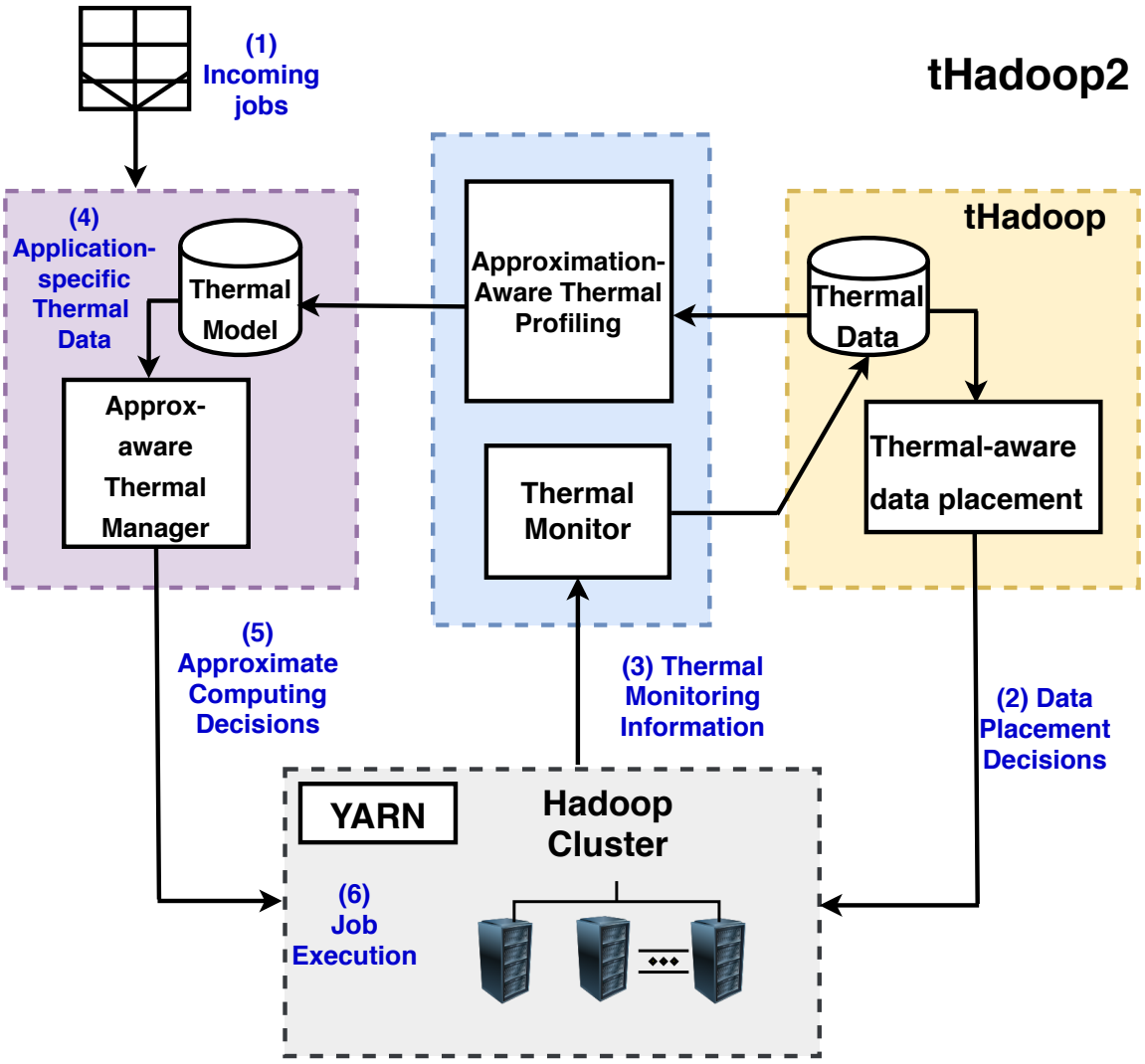


Figure 5.1: The *tHadoop2* framework demonstrating execution flow of a MapReduce job.

controlling the power consumption of nodes. For deciding thermal-aware workload placement, *tHadoop* constraints the number of tasks that are allowed to run on cluster nodes [73]. Instead of determining which tasks to schedule on nodes, *tHadoop* decides the number of tasks to schedule on cluster nodes.

The incoming jobs are fed into *tHadoop* which performs thermal-aware data placement by spreading the jobs' data across worker nodes. It is worth noting that *tHadoop* determines the number of tasks that can be executed on a node to conserve energy. The goals of thermal-aware task placement is to minimize peak incoming air temperatures resulting in reduced cooling costs [29]. For the purpose of data placement, it employs a heat-recirculation model to understand the heat flow among nodes and contribution of individual nodes towards heat buildup.

Once the data is placed across the worker nodes, *tHadoop2* ensures that the thermal stability of the nodes and surrounding environment through a thermal monitoring module. The sole responsibility of thermal monitoring module in *tHadoop2* is to collect real-time thermal data from the sensors installed on the nodes, ensuring that thermal constraints are not violated. It is essential that incoming air temperatures for nodes are kept low to prevent the formation of hotspots which can lead to harware failure in the long run [44]. If the required cooling is not delivered to these hotspots, the performance can be throttled down to reduce power [78]. Also, cooling energy reduction is best guaranteed when thermal-aware job scheduling is coordinated with thermal data obtained via Computer Room Air Conditioners (CRACs) [8]. This is the primary goal of incorporating *tHadoop* in *tHadoop2* framework.

5.2.2 The tHadoop2 Design

tHadoop2 is a thermal-aware resource management system orchestrating thermal-aware workload placement and applications resiliency to save energy in data centers. Thermal-aware workload placement is inevitable because in its absence, increasing temperatures may affect circuit reliability and life of hardware in a long term resulting in high cooling costs and energy inefficiency [78]. For this reason, *tHadoop2* depicted in Figure 5.1 seamlessly integrates two modules, namely approximate computing (see item 4 in Figure 5.1) and thermal-aware data placement (see *tHadoop* in Figure 5.1)

The *tHadoop2* consists of the following three modules:

- The thermal profiling and monitoring module (see the light blue box in Figure 5.1).
- The approximation-aware thermal manager or ATM (see item 4 in Figure 5.1).
- The tHadoop subsystem (see *tHadoop* in Figure 5.1).

The above three modules in the proposed framework communicate with the environment sensors through the thermal monitor module, which maintains rich thermal data in the thermal database residing in the *tHadoop* subsystem.

The main strengths of the *tHadoop2* framework are three-fold. First, temperatures of cluster nodes are modeled as a function of system configurations and applications' resource access

patterns. Second, a thermal model gathers per-application thermal data and cluster's temperatures including cross-interference, processor temperatures, and inlet/outlet air temperatures to assists ATM with thermal-aware scheduling decisions. The constructed models are managed in the ATM module. Third, *tHadoop2* opens an avenue for data center designers and application developers to investigate heat recirculation and cooling costs along with applications' resiliency.

In what follows, we outline the contributions of the *tHadoop* system in the *tHadoop2* framework, followed by the descriptions of the other primary modules in *tHadoop2*. The functionalities of the three core modules of *tHadoop2* are summarized in the list below:

- **tHadoop**. *tHadoop* [73] is responsible for capturing the spatial cross-interference - a phenomenon with which the heat generated by running jobs on a server not only raises a server's temperature but the temperatures of surrounding nodes due to air recirculation. *tHadoop* constitutes two important components, (1) a cross-interference matrix and (2) a scheduler. The cross-inferences matrix stores the fraction of outlet heat of each node contributing towards the inlet heat of every other node. The scheduler accesses the cross-interference matrix to assess a node's contribution towards heat re-circulation and modifies the number of tasks assigned to the Hadoop TaskTracker(s) running on the node.

- **Thermal Monitoring and Profiling.** This module initiates an automated thermal profiling process that captures nodes' inlet, outlet, ambient, and CPU core temperatures in real time. The acquired thermal data is then placed in the database of *tHadoop*. With the assistance of the gathered thermal data and identified resilience in applications, the profiling module initiates the modeling procedure to construct thermal models maintained by the approximation-aware thermal manager (see the item below). The thermal models are infused into the approximate-aware thermal manager to make judicious thermal-aware scheduling decisions. Please note that approximation-computing strategies are applied on a per-application basis, meaning that the monitoring and profiling module is obligated

to build a model for individual applications. The approximation-aware profiler may employ a variety of techniques ranging from scaling input data to using a predetermined quality metric to produce good-enough results. Performing proactive thermal management by fetching temperature measurements directly from on-chip sensors is ideal for constructing thermal models. Such thermal models help in making scheduling decisions by predicting future temperatures.

- **Approximation-Aware Thermal Manager (ATM).** ATM is a thermal manager that performs static scheduling of applications to meet predefined thermal goals. ATM is conducive of making job scheduling decisions while keeping thermal constraints in check. These scheduling decisions are forwarded to the YARN, which in turn requests for a set of containers expressing the amount of computer resources required for launching each container, as well as locality constraints for the containers [65]. Once the containers are allocated, a Hadoop job can be executed on cluster nodes in a thermal-conscious manner.

5.3 Empirical Study Results

In this section, we present a case study focusing on correlations between performance and accuracy when we evaluate the Pi estimator as an approximate computing application on a Hadoop cluster. More importantly, we show the thermal behavior and energy efficiency of the Pi estimator when we attempt to tune the computing accuracy.

After presenting the experimental testbed in Section 5.3.1, we outline the Pi evaluation method in Section 5.3.2. The single-node profiling and multiple-node profiling results can be found in Section 5.3.3 and Section 5.3.4, respectively.

5.3.1 Experimental Testbed

Before presenting the results of the Pi estimator, let us briefly summarize the testbed configuration as well as the setups of the three experiments.

Our testbed consists of one traditional rack consisting of 14 SuperMicro Model 825-7 servers with Intel Xeon X5650@2.67GHz processors; out of these 14 nodes, we use a subset of

these servers act as master and workers. The computer rack is located in Auburn University's High-Performance Computing Lab. The cooling unit supplies cool air from the ceiling and the room temperature is fixed at 63F. A Linux utility program called *lm-sensors* [69] is used to collect processor temperatures during the execution of Hadoop applications. The execution time for the experiments is measured in terms of elapsed wall clock time. To minimize thermal impacts of one test on its subsequent one, we ensured that the minimal interval between two subsequent tests is at least one hour. This strategy allows the testbed to cool down prior to the next round testing.

We conduct three sets of experiments by varying the number of worker nodes. In every experiment, we chose two configurable parameters, namely, number of maps and number of samples per map.

5.3.2 The Pi Evaluation Method

We observe the thermal behaviors of a popular Hadoop application, Pi estimator, which is used to estimate the mathematical value of π (i.e., 3.142857142857143...). We pay attention to the Pi estimator as a case study owing to the following facts. First, the Pi estimator can endure an approximated output value. Second, it is straightforward to observe and assess the accuracy of the computing results.

There are a few distinctive ways of implementing the Pi estimator. Among all the candidate approaches, we choose to test the Monte Carlo (MC) implementation of Pi. We don't imply by any means that the MC method is more optimal than other implementations of the Pi estimator. We sense that the other algorithms of the Pi estimator are likely to share a similar patterns as the MC method investigated in this study.

The MapReduce implementation of the Pi estimator employs Monte Carlo (MC) simulation for generating random samples to estimate the result. The MC method works by randomly generating points in a unit square plane where each point can be identified as either being inside or outside of the circle. Further, each sample is a 2-dimensional point (x,y) in a *Halton sequence H(i)* [79].

The value of π can be estimated by calculating the ratio of the number of points in the circle (i.e., $numInside$) versus in the square (i.e., $numTotal$); the area of unit square is 1 and the area of the inscribed circle is π/4. Then, Pi's estimated value is $4 * (numInside/numTotal)$. Each mapper is responsible for generating points in a unit square and counting the points inside/outside of the inscribed circle of the square; each generated point (x, y) falls in the range $[-1, 1]$. Every reducer, on the other hand, accumulates points from the mappers, where each qualifying point, satisfying $\sqrt{(x^2 + y^2)} < 1$, is used in the calculation of Pi.

Let's summarize the functionality of the mappers and reducers below.

- **Mapper**: Each mapper generates points in a unit square and then count points inside/outside of the inscribed circle of the square.
- **Reducer**: Each reducer accumulates inside/outside results from the mappers.
- **Estimating Pi**: The value of π can be estimated by calculating the ratio of the number of points in the circle (i.e., $numInside$) versus those in the square (i.e., $numTotal$).

When it comes to the Pi calculation, we can specify two parameters, namely, (1) sampling points and (2) the number of mappers, to estimate the value of Pi. Intuitively, a high precision can be obtained by specifying a large number of points, more iterations, or a combination of both. A variety of inputs must be considered for assessing the relationship between accuracy and energy efficiency and thus, we use an array of sampling points ranging from 100,000 to 1,000,000,000, where an iteration of each sampling point has 50 samples serving as a small input [59]. Similarly, we use the same range of sampling points, and one iteration each with 100 samples as a large input.

5.3.3 Single-Node Profiling

In this section, we focus on the profiling results collected from a single-node Hadoop server. In the next section (see Section 5.3.4), we will test the Pi estimator on a multiple-node Hadoop cluster.

The output of the Hadoop-based Pi estimator is an approximated value of π with varying precision. Before finding an appropriate approximation technique for Pi, we conduct a group

of experiments and record the number of precision digits, execution time, and average CPU temperatures of the worker nodes running these experiments.

Tables 5.2 and 5.3 show a detailed analysis of Pi running 50 and 100 mappers respectively, on a single server node. It is worth noting that the execution time increases in proportion with the number of sampling points. Our goal is to determine that optimal point beyond which there is no significant improvement in the quality of results and a tradeoff between computing quality and thermal constraints can be observed.

Observation 1: A General Trend Table 5.2 shows the correlation between sampling points and the estimated value of π. The results summarized in Table 5.2 suggests that increasing the number of sampling points is a way of improving the precision (i.e., an increased number of digits). For instance, in order to achieve one extra digit in the π result, we have to increase the number of samples by a factor of 10. Recall that these π values are generated using the Monte Carlo method discussed in Section 5.3.2. The error% is calculated using the experimental value (see column 2 in Tables 5.2 and 5.3) and the theoretical value of π (i.e., 3.142857142857143...). With very minute error, the Monte Carlo method is able to produce quite accurate value of π. Although the number of digits in the value of π increases with an increasing number of sampling points per map, there are diminishing improvements in terms of error% beyond a certain point (see row 3 in Tables 5.2 and 5.3). This trend is peculiar in Monte Carlo simulations since the results generated from running the Monte Carlo simulations follow a similar trend as $1/sqrt(x)$ plot.

Observation 2: Samples/Map vs. Execution Time. Although the number of digits in π goes up by adding more number of sampling points, such a precision improvement comes at an expense of elongated execution time. Evidence from our experimental results indicates that execution time is directly and strongly linked to power consumption of the nodes; thus, the longer an application executes on server nodes, the more electricity it consumes. The execution time gradually increases until the number of sampling points become 100,000,000. Beyond this point, the execution time becomes fivefold with both 50 and 100 mappers (see rows 4 and 5 in Tables 5.2 and 5.3). We speculate that the execution time increases multifold due to the limited availability of system resources on the cluster nodes for the Hadoop mappers and

reducers. Keeping thermal constraints, we are specifically interested in determining such an optimal point beyond which the execution time for an application increases multifold without any significant improvements in the quality of the results. For instance, in Fig. 5.4, the ideal number of sampling points for the calculation of Pi is 10^6. At this optimal point, the error% in the value of Pi as well as execution time is minimal.

Observation 3: Value/Error% vs. Energy Consumption. Let us pay attention to the correlation between number of sampling points and the energy consumed by the cluster node running the Pi estimator. Since the total amount of energy consumed in a cluster system over a given time interval is a product of average power consumed in the same time interval, energy and power are directly proportional to each other. Under a constant power supply like in *power-capped* systems, the energy consumption will increase linearly with time. In our example, the energy consumption with 1 billion sampling points will be the maximum and minimum with 100,000 samples owing to their execution times.

The implication behind this observation is that we can conserve energy consumption by immediately terminating the π calculation once the application produces a reasonably good result in terms of precision. Unsurprisingly, such an early termination leads to a reduction in overall execution time of the application as well as the energy consumption.

Observation 4: Thermal Trends An unexpected finding is that there is no interplay between the number of sampling points and processor temperatures. Recall that the number of mappers determines the number of threads launched by the π estimator for computing the value of π. The number of mappers is fixed to 50 and 100, respectively (see Tables 5.2 and 5.3). Further, the average processor temperature with 50 mappers is much less than that with 100 mappers (see the last column of Tables 5.2 and 5.3). We infer that since the number of mappers and the number of threads remain unchanged, there are no significant variations in the processors temperatures within each group (i.e, 50 mappers and 100 mappers). Minor fluctuations in the temperature measurements are due to the varying execution times for each set of experiments in Tables 5.2 and 5.3.

Sampling points	Calculated π	Error (%)	Execution Time (sec)	Avg. CPU Temp. (°C)
100000	3.1415728	0.04086	64.52	29.99
1000000	3.14159448	0.04017	64.502	31.26
10000000	3.141593128	0.04021	64.512	30.62
100000000	3.1415927992	0.04023	114.693	30.79
1000000000	3.14159266896	0.04023	552.98	31.43

Table 5.2: Performance and accuracy of Pi under various number of sampling points. The number of mappers is fixed to 50.

Samples per map	Calculated π	Error (%)	Execution Time (sec)	Avg. CPU Temp. (°C)
100000	3.1415844	0.0405	114.65	31.33
1000000	3.14159256	0.04024	114.672	31.83
10000000	3.141592736	0.04023	119.128	32.82
100000000	3.1415926492	0.04023	214.03	31.91
1000000000	3.1415926568	0.04023	1090.482	31.84

Table 5.3: Performance of Pi running on a single server node running 100 mappers. The number of samples per map are varied for every experiment and the performance and accuracy is recorded.

5.3.4 Scaling Out

To further enhance our understanding of Pi and its thermal effects on cluster nodes, we conduct a group of experiments on a cluster comprising of one master node and two worker nodes. Adding more resources to a cluster does not necessary improve performance and therefore, scaling out our existing cluster will help us determine the resources that are just enough to produce a quality output for the tested applications. Since we have acquired a fair idea on the execution time of Pi with a variety of sampling points and number of mappers, we restrict the number of mappers to only 25, 50, and 100; the number of sampling points ranges between 100,000,000 (i.e., 10^8) and 1,000,000,000 ((or 10^9)). Figures. 5.2 and 5.3 show the average processor temperature and energy consumption of two data nodes running 25, 50, and 100 mappers with 10^8 and 10^9 sampling points per map.

We observe similar trends in terms of average processor temperatures and energy consumption on both data nodes thanks to Hadoop's fair load balancing. It is interesting to note that irrespective of the sampling points and execution time, the average processor temperature is minimum when the number of mappers is set to 50. This observation motivates us to look

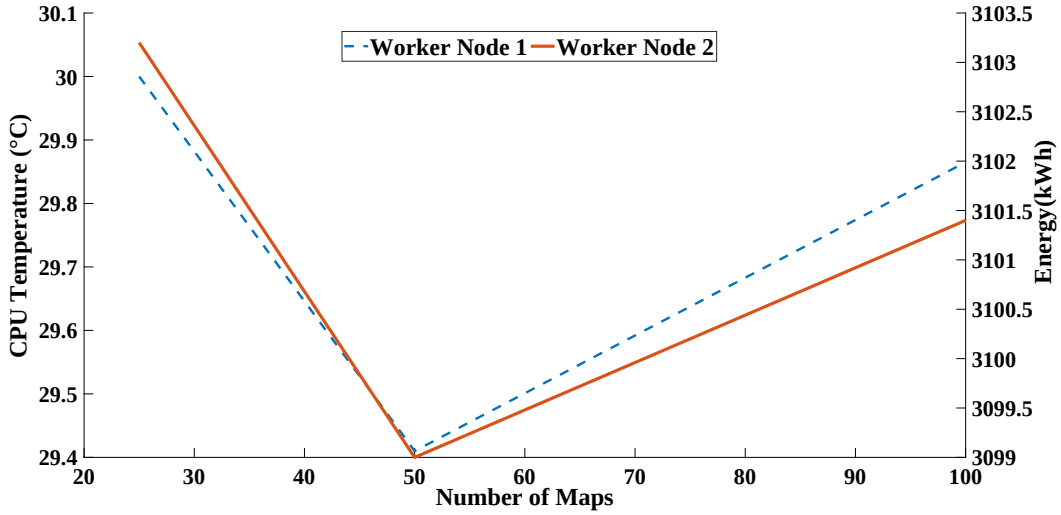


Figure 5.2: Average CPU core temperatures and energy consumption of two server nodes running 25, 50, and 100 mappers with 100,000,000 sampling points. The execution times for running Pi with 25, 50, and 100 mappers are 66, 67, and 130 seconds, respectively.

into minimizing thermal impacts on the nodes by tuning the number of mappers launched with the applications. We also believe that the number of nodes chosen to run an application can be tuned to further conserve energy. The deviation in average CPU temperatures is larger in case of 10^9 sampling points than 10^8 sampling points due to the elongated duration of the experimental time. In the first case (see Fig. 5.2, the experiments do not run long enough to heat up the processor cores to the same extent as in the second case (see Fig. 5.3). Further, for a given number of mappers, there is a narrow gap in the processor temperatures between the two data nodes; this behavior prevails due to several complex factors like heat redistribution and height of server nodes in the rack.

5.3.5 Estimating Accuracy

For a wide range of applications, the overall computational cost can be significantly reduced if an exact solution is not required. In the previous section (see Section 5.3.3, we perform a detailed analysis of a Hadoop application, Pi, that employs Monte Carlo algorithm to determine the precision digits of π. In this section, we pay special attention to applying an approximation strategy called early termination to Pi by identifying a tradeoff point.

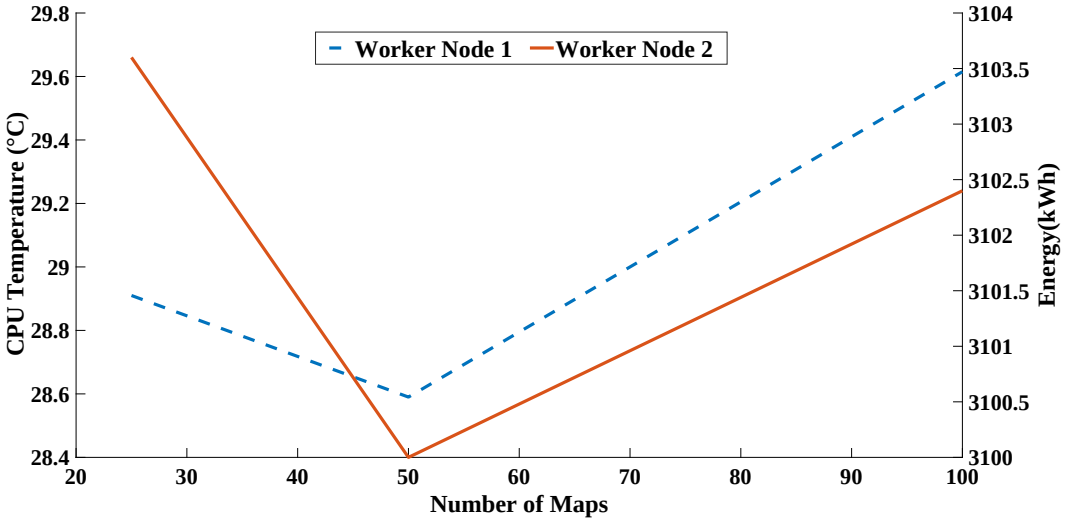


Figure 5.3: Average CPU core temperatures and energy consumption of two server nodes running 25, 50, and 100 mappers with 1,000,000,000 sampling points. The execution times for running Pi with 25, 50, and 100 mappers are 165, 300, and 600 seconds respectively.

It is evident that there is no *one-solution-fits-all* approximation technique for every Hadoop application, which exhibits a domain-specific notion of "quality". Thus, a per-application profiling must be performed to identify resiliency for the purpose of approximation [59]. After having identified elements of an application that can tolerate error and the extent to which errors can be tolerated, instructions of applications are executed for energy savings [80]. The preliminary results from running Pi show that trading off the accuracy for the computations to save energy is worth investigating.

In the case of Pi, the accuracy can be determined in several ways. To determine the tradeoff point when a user predetermines a quality level, we can define a tolerance level $eps > 0$ (where eps is the error-per-step) and declare that the error has leveled off if

$$\frac{d}{dx}(\frac{1}{sqrt(x)}) < eps \tag{5.1}$$

For instance, Fig. 5.4 illustrates that the output quality (in terms of error %) does not improve after a certain point (10^6 sampling points). This saturation point is often referred to as an *optimal point* or *tradeoff point*. Our preliminary findings suggests that there is no significant improvement in the quality of the results beyond the optimal point. It must be noted that the

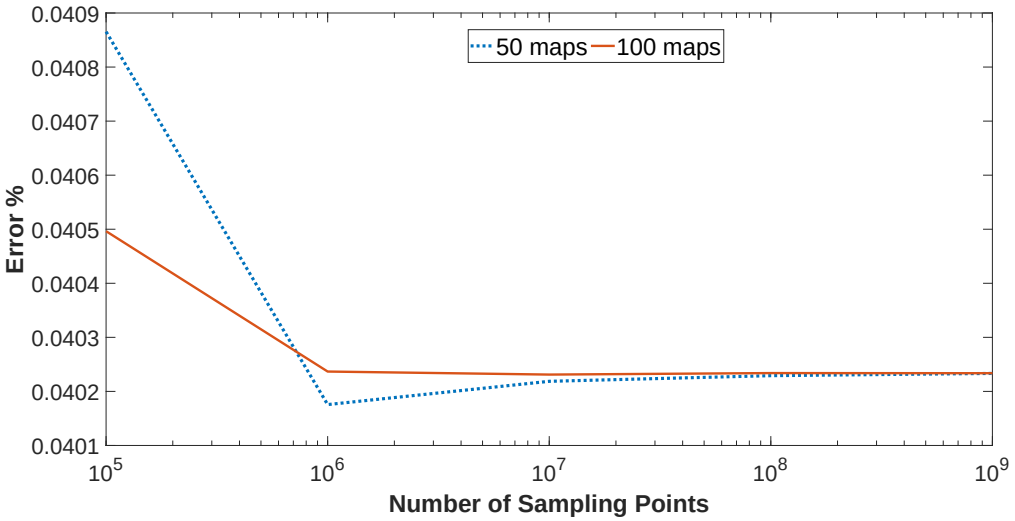


Figure 5.4: Calculated error % in the accuracy of digits in Pi with 50 and 100 mappers. The number of worker nodes is fixed at two.

execution time and energy dissipation continue increasing beyond the optimal point. Since a varying number of mappers exhibit similar trends in terms of energy consumption and CPU temperatures, we focus on the number of sampling points to determine an optimal point for the Pi estimator while overlooking the number of mappers in case when a user-defined tolerance level is not available.

To achieve energy-efficiency in the execution of Pi, we endeavor to find a tradeoff point (a.k.a., optimal point) that results in minimal degradation of result quality while keeping thermal-efficiency in consideration. To determine the right extent to which inherent resilience in each application can be exploited is through evaluating the impacts of an array of approximation techniques on the application output quality.

5.4 Summary

Recognizing that high energy efficiency of applications helps in reducing the operational cost of data centers, we proposed a framework called *tHadoop2* for MapReduce applications running on Hadoop clusters. To facilitate the development of *tHadoop2*, we incorporated an existing thermal-aware workload placement module called *tHadoop* into our *tHadoop2*. Our framework consists of three key components - *tHadoop*, a *thermal monitoring and profiling* module and a *approximation-aware thermal manager*. *tHadoop* performs thermal-aware data placement by

distributing applications' data across worker nodes. The thermal monitoring and profiling module is used to collect real-time thermal data from sensors installed on cluster nodes, ensuring that thermal constraints are not violated. The approximation-aware thermal manager or ATM helps in improving the energy efficiency of cluster nodes by applying approximation strategies on a per-application basis.

We investigated the thermal behavior of a MapReduce application called Pi running on Hadoop clusters by varying two input parameters - number of maps and number of sampling points per map. Our profiling results show that Pi exhibits inherent resilience in terms of number of precision digits present in its value. It is noteworthy that approximation-computing strategies are applied on a per-application basis. Other MapReduce applications can be scrutinized by exploring their characteristics and finding opportunities for acceptable inexactness in output. Nevertheless, the proposed framework coupled with approaches for making tradeoffs is generally applicable to any MapReduce application.

After we offered detailed analysis on the Pi running on one server, we scaled up our cluster running the Pi estimator, where we varied the number of maps and sampling points. We observed similar trends in terms of average processor temperatures and energy consumption on the data nodes. Regardless of the sampling points and execution time, the average processor temperature and energy consumption is minimum when the number of mappers is set to 50 in our experiments. We expect that the number of nodes chosen to run applications should be tuned to further conserve energy.

We avoid the discussion on heat redistribution, height of server nodes, and ambient temperatures in this part of study. Nevertheless, heat redistribution and ambient temperatures in the context of Hadoop have been investigated in the *tHadoop* project [73]. In our future work, we will extend *tHadoop2* to address these challenges.

Chapter 6

Conclusion and Future Work

The focus of this dissertation research was on designing thermal-aware resource management solutions to achieve high energy efficiency of data centers. Most of our work can be deployed in large-scale data centers to mitigate the heat build-up and rising cooling costs.

Growing number of data centers are being set up across the world in the last decade. According to the EPA report, US data centers consumed approximately 2% of total U.S. energy consumption, with 24% increase in the last five years. This continuous increase in energy consumption has motivated researchers to design low-cost solutions for operating data centers. Strong evidence indicates that cooling cost and computing power are the most significant contributors of a data center's operational cost. Prior studies confirmed that cutting cooling cost effectively improves the energy efficiency of data centers.

6.1 Past and Current Work

In this research, our focus has been on the design and development of software solutions that broadly explore thermal-aware resource management issues with particular attention to improving thermal-efficiency and energy-efficiency for the clusters housed in data centers. This dissertation research contributions span:

6.1.1 Thermal-Profiling and Thermal-Aware Job Scheduling

Characterizing thermal profiles of cluster nodes is an integral part of any approach that addresses thermal emergencies in a data center. We proposed a thermal-aware scheduler, *tDispatch* which is conducive of improving the energy efficiency of Hadoop clusters. The scheduling idea of *tDispatch* is motivated by a profiling study of CPU-intensive and I/O-intensive jobs from the perspective of thermal efficiency. We investigated the thermal behaviors of CPU-intensive and I/O-intensive applications running on Hadoop clusters by stress testing the data nodes through extensive experiments. Our findings show that CPU-intensive and I/O-intensive jobs exhibit distinct thermal behaviors on multicore processors and hard drives of Hadoop clusters. Our study suggests that one way to make Hadoop clusters thermal friendly is to dispatch CPU-intensive and I/O-intensive jobs to the cluster nodes where CPUs and disks are running at a lower temperature; we implemented this idea with the aid of two job scheduling algorithms, *tQueue* and *tDispatch*. The experimental results confirm that *tDispatch* improves the overall thermal efficiency of the data nodes in the tested Hadoop cluster.

6.1.2 Thermal-Modeling

Thermal models are used to characterize the thermal profile of a data center. We proposed a thermal model called *tModel* aiming to predict outlet temperatures of servers in Hadoop clusters running big-data applications. tModel eliminates a pressing need to estimate power consumed by Hadoop nodes prior to speculating their outlet temperatures. tModel makes use of a vital parameter K_i to resemble correlation between outlet and inlet temperatures of the ith node. Parameter K_i is obtained through profiling in the sampling phase. We studied various metrics like CPU temperature, inlet/outlet temperatures, height of the node in a rack, air recirculation to develop a prediction model for outlet temperatures. This model can be used as an input to the popular *COP* model, developed at the Intel Labs, which estimates the cooling costs for data centers. Our overall goal is to investigate a thermal modeling approach to estimate outlet temperatures of server nodes on Hadoop cluster without relying on the power consumption models of the nodes. It is noteworthy that conventional thermal models are built on top of

power consumption models. We intend to eliminate the power consumption models as a middle man in the existing thermal models.

There are three contributions in this work. First, we conduct a thermal profiling study of a cluster on which various Hadoop applications are investigated. When the applications process various data size under different number of nodes, we profile our server nodes to monitor multicore CPU temperatures. Second, we build a thermal model to estimate temperatures using inlet temperatures and CPU core temperatures. Third, as a use case of our model, we extend the model to predict the cooling cost of a Hadoop cluster.

6.1.3 Thermal-aware Approximate Computing for MapReduce Applications

Recognizing that high energy efficiency of applications helps in reducing the operational cost of data centers, we proposed a framework called *tHadoop2* for MapReduce applications running on Hadoop clusters. To facilitate the development of *tHadoop2*, we incorporated an existing thermal-aware workload placement module called *tHadoop* into our *tHadoop2*. Our framework consists of three key components - *tHadoop*, a *thermal monitoring and profiling* module and a *approximation-aware thermal manager*. *tHadoop* performs thermal-aware data placement by spreading jobs' data across cluster nodes. The thermal monitoring and profiling module is used to collect real-time thermal data from sensors installed on cluster nodes, ensuring that thermal constraints are not violated. The approximation-aware thermal manager or ATM helps in improving the energy efficiency of the cluster nodes by applying approximation strategies on a per-application basis.

We investigated the thermal behaviors of a MapReduce application called Pi running on Hadoop clusters by varying the two input parameters - number of maps and number of sampling points per map. Our profiling results show that Pi exhibits inherent resilience in terms of number of precision digits present in its output. It is noteworthy that this result quality is application-specific. Other MapReduce applications can be scrutinized for exploring their characteristics and finding opportunities for acceptable inexactness in output. The proposed framework coupled with the approach to making tradeoffs is generally applicable to any MapReduce application.

6.2 Future Directions

As a short-term plan, we will concentrate on the following two directions to extend our past and current research on energy-efficient and thermal-aware computing in high-performance clusters.

6.2.1 Efficient Data Availability in CFS

Data availability and data redundancy are used for dealing with failures in clustered file systems (CFSes). Two redundancy schemes commonly employed in CFSes are replication and erasure coding. Replication schemes create identical replicas for each data block; erasure-coding schemes store additional information as parity blocks. Lately, there has been an inclination towards adopting erasure coding technique in CFSes, because erasure coding is more storage efficient than replication techniques. We plan to conduct a study to investigate the thermal-efficiency of both replication and erasure-coding schemes. Further, we will be focusing on energy consumed in creating and repairing erasure codes to enhance the thermal-awareness in our design. Experiments will be conducted on a Hadoop cluster, because the Hadoop Data File System (HDFS) employs disk-based data replication strategy. We intend to discover if erasure-coding can improve data storage and energy consumption costs for HDFS over the default 3x-replication technique in Hadoop.

Data availability in Hadoop becomes a bigger challenge in geo-distributed data centers. The traditional MapReduce programming model, which comprises of two phases - *Map* (local computation) and *Reduce* (all-to-all communication), is not optimized for deployment across data centers. This problem is caused by the need to aggregate data, produced by mappers, for centralized processing in *Reduce* phase. To further enrich our dissertation research, we plan to delve into thermally-efficient and cost-optimized data availability for processing big-data using the MapReduce programming framework in geo-distributed data centers.

6.2.2 Modeling in HPC Applications

Batch-oriented and stream-oriented data processing are two popular paradigms used for processing jobs in high-performance clusters (HPC). The latency issues in batching and performance issues in erasure-coding have gained attention lately. We observe that the thermal impacts of the batching and streaming scheduling paradigms are still in their infancy. With these two data processing strategies in place, we plan to investigate their thermal impacts on Hadoop and Spark clusters using the workloads available in TPC-HS benchmark suite. We will build a model to resemble thermal behaviors of schedulers that manages resources for Hadoop and Spark applications. Our models will incorporate access patterns of multicore processors, memory, as well as data storage.

Energy-Efficiency in Data Stream Processing: Streaming applications are characterized by unpredictable execution scenarios owing to their variable arrival rates, deadlines, data size, latency-sensitivity and the like. It is an intriguing project to model resource allocations in streaming applications running on Spark clusters governed by the dynamic voltage and frequency scaling technique or DVFS. Our novel model will enable us to project the operational costs of streaming applications running on energy-efficient Spark equipped with DVFS. More often than not, data streaming applications are latency-sensitive in nature. After the development of this model, we will identify bottlenecks with respect to energy consumption for a group of data streaming applications running on high-performance Spark clusters.

6.2.3 Energy-Efficient High-Performance Computing

Limited power budgets will be one of the grand challenges for deploying future exascale supercomputers. The path to future large-scale HPC system development is mitigating the power and energy requirements for reaching new levels of performance. The DOE has stated a goal of achieving exascale levels of performance on a power budget of 20 MegaWatts, which is significantly beyond the capabilities seen on systems today [81]. To achieve this goal, system architects, runtime software designers, and application developers must take a holistic approach

to designing energy-efficient systems, including examining novel technologies and software approaches. As an example, data movement is envisioned to be a major consumer of power within large-scale HPC systems [82]. Reducing the overall data movement will help to reduce power consumption, although this may present other challenges in terms of load distribution and ideal load balance across the system.

In the course of this research, we will be exploring the impact of novel technologies on HPC workload power consumption and will be developing tools and techniques for the measurement of power and energy consumption. In addition, we will be exploring the intersection between power consumption, performance, and thermal characteristics on state-of-the-art HPC hardware, including the potential development of energy reduction techniques and intelligent software.

6.3 Conclusion

Characterizing thermal profiles of cluster nodes is an integral part of many approaches that addresses thermal emergencies in a data center. Thermal models are used to characterize the thermal profile of a data center. We developed a thermal model, *tModel*, that projects outlet temperatures using inlet temperatures and multicore temperatures procured from processor core sensors. For validating our model, we picked several big-data applications from a popular benchmarking suite, *HiBench*, to study the impact of real-life workloads on cluster nodes. We studied various metrics like CPU temperature, inlet/outlet temperatures, height of the node in a rack and air recirculation to develop a prediction model for outlet temperatures. This model can be used as an input to the popular *COP* model, developed at the Intel Labs, which estimates the cooling costs for data centers. Our thermal model offers two compelling benefits. First, *tModel* makes it possible to cut back thermal monitoring cost by immensely reducing the number of physical sensors needed for large-scale clusters. Monitoring temperatures is a vital issue for safely operating data centers. However, it is prohibitively expensive to acquire and set up a large number of sensors in a large data center. Second, *tModel* enables data center designers to evaluate thermal management strategies during the center design phase.

Recognizing that improving thermal efficiency of clusters helps in reducing the operational cost of data centers, we developed a thermal-aware job scheduling strategy called *tDispatch* for MapReduce applications running on Hadoop clusters. To facilitate the development of *tDispatch*, we investigated the thermal behaviors of CPU-intensive and I/O-intensive applications running on Hadoop clusters by stress testing the data nodes through extensive experiments. Our findings show that CPU-intensive and I/O-intensive jobs exhibit distinct thermal and performance impacts on multicore processors and hard drives of Hadoop clusters. Our study suggests that one way to make Hadoop clusters thermal friendly is to dispatch CPU-intensive and I/O-intensive jobs to the cluster nodes where CPUs and disks are running at a lower temperature. We implemented this idea with the aid of two job scheduling algorithms, *tQueue* and *tDispatch*. We conducted an empirical study to compare the thermal and energy performance of *tDispatch* with two scheduler counterparts, namely, CPUF and IOF. Specifically, we measured the CPU and disk temperatures of the master node and data nodes governed by the three schedulers. Our findings suggest that the schedulers make little impact on the thermal efficiency of the master node. More importantly, our experimental results show that *tDispatch* is superior to CPUF and IOF in terms of improving thermal efficiency of data nodes.

References

[1] O. V. Geet, *Trends in Data Center Design - ASHRAE Leads the Way to Large Energy Savings*, 2014.

[2] J. G. Koomey, "Estimating total power consumption by servers in the U.S. and the world," tech. rep., Lawrence Derkley National Laboratory, Feb. 2007.

[3] Gartner, "Four megatrends impacting the data center," 2016.

[4] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Computer Architecture, 2006. ISCA '06. 33rd International Symposium on*, pp. 78–88, 2006.

[5] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, *et al.*, "An overview of energy efficiency techniques in cluster computing systems," *Cluster Computing*, vol. 16, no. 1, pp. 3–15, 2013.

[6] U. E. P. Agency, "Report to congress on server and data center energy efficiency," tech. rep., EPA, August 2007.

[7] http://www.datacenterdynamics.com/research/energy-demand 2011-12, "Global data center energy demand forecasting," tech. rep., institution, 2011.

[8] A. Banerjee, T. Mukherjee, G. Varsamopoulos, and S. K. Gupta, "Cooling-aware and thermal-aware workload placement for green hpc data centers," in *Green Computing Conference, 2010 International*, pp. 245–256, IEEE, 2010.

[9] P. Chaparro, G. Magklis, J. Gonzalez, and A. Gonzalez, "Distributing the frontend for temperature reduction," in *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pp. 61–70, Feb 2005.

[10] J. Moore, R. Sharma, R. Shih, J. Chase, R. Patel, P. Ranganathan, and H. P. Labs, "Going beyond cpus: The potential of temperature-aware solutions for the data center," in *In Proceedings of the Workshop of Temperature-Aware Computer Systems (TACS-1) held at ISCA*, 2004.

[11] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling "cool": temperature-aware workload placement in data centers," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, (Berkeley, CA, USA), pp. 5–5, USENIX Association, 2005.

[12] N. Vasic, T. Scherer, and W. Schott, "Thermal-aware workload scheduling for energy efficient data centers," in *Proceedings of the 7th international conference on Autonomic computing*, ICAC '10, (New York, NY, USA), pp. 169–174, ACM, 2010.

[13] X. Jiang, M. Alghamdi, J. Zhang, M. Assaf, X. Ruan, T. Muzaffar, and X. Qin, "Thermal modeling and analysis of storage systems," in *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pp. 31–40, Dec 2012.

[14] K. Kambatla, G. Kollias, V. Kumar, and A. Grama, "Trends in big data analytics," *Journal of Parallel and Distributed Computing*, vol. 74, no. 7, pp. 2561–2573, 2014.

[15] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proceedings of the 50th Annual Design Automation Conference*, p. 113, ACM, 2013.

[16] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *Proceedings of the 52nd Annual Design Automation Conference*, p. 120, ACM, 2015.

[17] A. Sampson, *Hardware and software for approximate computing*. PhD thesis, Cornell University, 2015.

[18] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto, "Survey of energy-cognizant scheduling techniques," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1447–1464, 2013.

[19] Q. Tang, S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, pp. 1458–1472, Nov 2008.

[20] M. T. Chaudhry, T. C. Ling, A. Manzoor, S. A. Hussain, and J. Kim, "Thermal-aware scheduling in green data centers," *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, p. 39, 2015.

[21] Q. Tang, T. Mukherjee, S. K. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," in *Intelligent Sensing and Information Processing, 2006. ICISIP 2006. Fourth International Conference on*, pp. 203–208, IEEE, 2006.

[22] L. Ramos and R. Bianchini, "C-oracle: Predictive thermal management for data centers," in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pp. 111–122, IEEE, 2008.

[23] T. Cao, W. Huang, Y. He, and M. Kondo, "Cooling-aware job scheduling and node allocation for overprovisioned hpc systems," in *Parallel and Distributed Processing Symposium (IPDPS), 2017 IEEE International*, pp. 728–737, IEEE, 2017.

[24] M. Polverini, A. Cianfrani, S. Ren, and A. V. Vasilakos, "Thermal-aware scheduling of batch jobs in geographically distributed data centers," *IEEE Transactions on cloud computing*, vol. 2, no. 1, pp. 71–84, 2014.

[25] S. Alsubaihi and J.-L. Gaudiot, "Pets: Performance, energy and thermal aware scheduler for job mapping with resource allocation in heterogeneous systems," in *Performance Computing and Communications Conference (IPCCC) Poster, 2016 IEEE 35th International*, pp. 1–2, IEEE, 2016.

[26] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, pp. 1458–1472, Nov. 2008.

[27] E. M. Elnozahy, M. Kistler, and R. Rajamony, "Energy-efficient server clusters," in *International Workshop on Power-Aware Computer Systems*, pp. 179–197, Springer, 2002.

[28] O. Sarood and L. V. Kale, "A 'cool' load balancer for parallel applications," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, (New York, NY, USA), pp. 21:1–21:11, ACM, 2011.

[29] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," in *Advances in computers*, vol. 82, pp. 47–111, Elsevier, 2011.

[30] G. Varsamopoulos, A. Banerjee, and S. K. Gupta, "Energy efficiency of thermal-aware job scheduling algorithms under various cooling models," *Contemporary Computing*, pp. 568–580, 2009.

[31] J. Leverich and C. Kozyrakis, "On the energy (in)efficiency of hadoop clusters," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 61–65, Mar. 2010.

[32] R. T. Kaushik and M. Bhandarkar, "Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster," in *Proceedings of the USENIX Annual Technical Conference*, p. 109, 2010.

[33] A. Che, K. Lv, E. Levner, and V. Kats, "Energy consumption minimization for single machine scheduling with bounded maximum tardiness," in *Networking, Sensing and Control (ICNSC), 2015 IEEE 12th International Conference on*, pp. 146–150, IEEE, 2015.

[34] J. Li, M. Qiu, J.-W. Niu, L. T. Yang, Y. Zhu, and Z. Ming, "Thermal-aware task scheduling in 3d chip multiprocessor with real-time constrained workloads," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 12, no. 2, p. 24, 2013.

[35] M. T. Chaudhry, T. C. Ling, A. Manzoor, S. A. Hussain, and J. Kim, "Thermal-aware scheduling in green data centers," *ACM Comput. Surv.*, vol. 47, pp. 39:1–39:48, Feb. 2015.

[36] X. Qin, H. Jiang, A. Manzanares, X. Ruan, and S. Yin, "Communication-aware load balancing for parallel applications on clusters," *Computers, IEEE Transactions on*, vol. 59, no. 1, pp. 42–52, 2010.

[37] X. Qin, H. Jiang, A. Manzanares, X. Ruan, and S. Yin, "Dynamic load balancing for i/o-intensive applications on clusters," *ACM Transactions on Storage (TOS)*, vol. 5, no. 3, p. 9, 2009.

[38] S. Taneja, S. Kulkarni, Y. Zhou, and X. Qin, "Thermal-aware task assignments in high performance computing clusters," *Concurrency and Computation: Practice and Experience*, vol. 29, 2017.

[39] M. Pastorelli, A. Barbuzzi, D. Carra, M. Dell'Amico, and P. Michiardi, "Hfsp: Size-based scheduling for hadoop," in *Big Data, 2013 IEEE International Conference on*, pp. 51–59, Oct 2013.

[40] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in *Proceedings of the 5th European Conference on Computer Systems*, EuroSys '10, (New York, NY, USA), pp. 265–278, ACM, 2010.

[41] R. Nanduri, N. Maheshwari, A. Reddyraja, and V. Varma, "Job aware scheduling algorithm for mapreduce framework," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pp. 724–729, Nov 2011.

[42] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving mapreduce performance in heterogeneous environments.," in *OSDI*, vol. 8, p. 7, 2008.

[43] L. Mashayekhy, M. M. Nejad, D. Grosu, D. Lu, and W. Shi, "Energy-aware scheduling of mapreduce jobs," in *Proceedings of the 2014 IEEE International Congress on Big Data*, BIGDATACONGRESS '14, (Washington, DC, USA), pp. 32–39, IEEE Computer Society, 2014.

[44] K. Mukherjee, S. Khuller, and A. Deshpande, "Algorithms for the thermal scheduling problem," in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, pp. 949–960, May 2013.

[45] T. Wirtz and R. Ge, "Improving mapreduce energy efficiency for computation intensive workloads," in *Proceedings of the 2011 International Green Computing Conference and Workshops*, IGCC '11, (Washington, DC, USA), pp. 1–8, IEEE Computer Society, 2011.

[46] L. Li, C.-J. M. Liang, J. Liu, S. Nath, A. Terzis, and C. Faloutsos, "Thermocast: a cyber-physical forecasting model for datacenters," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, (New York, NY, USA), pp. 1370–1378, ACM, 2011.

[47] S. Li, H. Le, N. Pham, J. Heo, and T. Abdelzaher, "Joint optimization of computing and cooling energy: Analytic model and a machine room case study," in *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*, pp. 396–405, IEEE, 2012.

[48] R. Ahmed, P. Ramanathan, and K. K. Saluja, "Necessary and sufficient conditions for thermal schedulability of periodic real-time tasks under fluid scheduling model," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 15, no. 3, p. 49, 2016.

[49] S. Taneja, Y. Zhou, and X. Qin, "Thermal benchmarking and modeling for hpc using big data applications," *Future Generation Computer Systems*, 2018.

[50] S. Taneja, A. Chavan, Y. Zhou, M. Alghamdi, and X. Qin, "Thermal profiling and modeling of hadoop clusters using bigdata applications," in *High Performance Computing Workshops (HiPCW), 2017 IEEE 24th International Conference on*, pp. 18–24, IEEE, 2017.

[51] L. Parolini, B. Sinopoli, B. H. Krogh, and Z. Wang, "A cyber physical systems approach to data center modeling and control for energy efficiency," *Proceedings of the IEEE*, vol. 100, pp. 254–268, Jan 2012.

[52] R. T. Kaushik and K. Nahrstedt, "T*: A data-centric cooling energy costs reduction approach for big data analytics cloud," in *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, pp. 1–11, IEEE, 2012.

[53] B. Rountree, D. H. Ahn, B. R. De Supinski, D. K. Lowenthal, and M. Schulz, "Beyond dvfs: A first look at performance under a hardware-enforced power bound," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pp. 947–953, IEEE, 2012.

[54] S. Mittal, "A survey of techniques for designing and managing cpu register file," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 4, 2017.

[55] I. Goiri, R. Bianchini, S. Nagarakatte, and T. D. Nguyen, "Approxhadoop: Bringing approximations to mapreduce frameworks," in *ACM SIGARCH Computer Architecture News*, vol. 43, pp. 383–397, ACM, 2015.

[56] S. T. Chakradhar and A. Raghunathan, "Best-effort computing: Re-thinking parallel software and hardware," in *Design Automation Conference*, pp. 865–870, June 2010.

[57] W. Baek and T. M. Chilimbi, "Green: a framework for supporting energy-conscious programming using controlled approximation," in *ACM Sigplan Notices*, vol. 45, pp. 198–209, ACM, 2010.

[58] X. Hui, Z. Du, J. Liu, H. Sun, Y. He, and D. A. Bader, "When good enough is better: Energy-aware scheduling for multicore servers," in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pp. 984–993, May 2017.

[59] I. Akturk, K. Khatamifard, and U. R. Karpuzcu, "On quantification of accuracy loss in approximate computing," in *Workshop on Duplicating, Deconstructing and Debunking (WDDD)*, p. 15, 2015.

[60] M. Polverini, A. Cianfrani, S. Ren, and A. V. Vasilakos, "Thermal-aware scheduling of batch jobs in geographically distributed data centers.," *IEEE Trans. Cloud Computing*, vol. 2, no. 1, pp. 71–84, 2014.

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

[62] Q. Tang, S. Gupta, and G. Varsamopoulos, "Thermal-aware task scheduling for data centers through minimizing heat recirculation," in *Cluster Computing, 2007 IEEE International Conference on*, pp. 129 –138, sept. 2007.

[63] F. Kong and X. Liu, "A survey on green-energy-aware power management for datacenters," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, p. 30, 2015.

[64] H. Xu, C. Feng, and B. Li, "Temperature aware workload managementin geo-distributed data centers," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 26, no. 6, pp. 1743–1753, 2015.

[65] Wikipedia, "Mapreduce," 2016.

[66] A. Silberschatz, *Operating System Concepts, 9th Edition*. John Wiley Sons, 2012.

[67] S. Huang, J. Huang, J. Dai, T. Xie, and B. Huang, "The hibench benchmark suite: Characterization of the mapreduce-based data analysis," in *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on*, pp. 41–51, IEEE, 2010.

[68] E. Pakbaznia and M. Pedram, "Minimizing data center cooling and server power costs," in *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*, pp. 145–150, ACM, 2009.

[69] "lm-sensors." `http://www.lm-sensors.org/`.

[70] T. Heath, A. P. Centeno, P. George, L. Ramos, Y. Jaluria, and R. Bianchini, "Mercury and freon: Temperature emulation and management for server systems," in *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XII, (New York, NY, USA), pp. 106–116, ACM, 2006.

[71] Intel, "The problem of power consumption in server," 2009.

[72] Apache, "Apache hadoop," 2008.

[73] A. Chavan, *Thermal-Aware File and Resource Allocation in Data Centers*. PhD thesis, Auburn University, 2017.

[74] J. Ansel, Y. L. Wong, C. Chan, M. Olszewski, A. Edelman, and S. Amarasinghe, "Language and compiler support for auto-tuning variable-accuracy algorithms," in *Proceedings of the 9th Annual IEEE/ACM International Symposium on Code Generation and Optimization*, CGO '11, (Washington, DC, USA), pp. 85–96, IEEE Computer Society, 2011.

[75] A. Subcommittee, "Top ten exascale research challenges," *US Department Of Energy Report*, 2014.

[76] A. Chavan, X. Jiang, M. I. Alghamdi, X. Qin, M. Jiang, and J. Zhang, "Tiger: Thermal-aware file assignment in storage clusters," *Mass Storage Systems and Technologies, IEEE / NASA Goddard Conference on*, vol. 0, pp. 1–5, 2013.

[77] S. Taneja, Y. Zhou, M. I. Alghamdi, and X. Qin, "Thermal-aware job scheduling of mapreduce applications on high performance clusters," in *Parallel Processing Workshops (ICPPW), 2017 46th International Conference on*, pp. 261–270, IEEE, 2017.

[78] A. L. Moore and L. Shi, "Emerging challenges and materials for thermal management of electronics," *Materials Today*, vol. 17, no. 4, pp. 163–174, 2014.

[79] Wikipedia, "Halton sequence," 2017.

[80] L. Kugler, "Is good enough computing good enough?," *Communications of the ACM*, vol. 58, no. 5, pp. 12–14, 2015.

[81] D. S. Holmes, A. L. Ripple, and M. A. Manheimer, "Energy-efficient superconducting computing-power budgets and requirements," *IEEE Transactions on Applied Superconductivity*, vol. 23, no. 3, pp. 1701610–1701610, 2013.

[82] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie, "Quantifying the energy cost of data movement in scientific applications," in *Workload Characterization (IISWC), 2013 IEEE International Symposium on*, pp. 56–65, IEEE, 2013.