Thermal-Aware File and Resource Allocation in Data Centers

by

Ajit Chavan

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

Auburn, Alabama
December 16, 2017

Keywords: Thermal-aware, File Allocation, Scheduling, Storage Systems, Hadoop

Approved by

Xiao Qin, Professor of Computer Science and Software Engineering
Saad Biaz, Professor of Computer Science and Software Engineering
Wei-Shinn Ku, Associate Professor of Computer Science and Software Engineering
Sanjeev Baskiyar, Professor of Computer Science and Software Engineering

Abstract

After addressing the issue of reducing power consumption by computing nodes in data centers, in recent years, computer scientists are focusing on reducing cooling cost of the data centers, thereby making the data centers thermal-aware. Due to the dramatic increase in power-density of data centers, thermal-management strategies are gaining more and more significance in the area of high performance computing data centers. Most of the previous studies towards achieving this goal focus on the nodes performing computation-intensive tasks, in which, the processors are the major consumer of the node's power. However, there is a lack of study on the ways of thermal-aware data placement in storage clusters housing thousands of storage nodes, where the disk subsystems are the major power consumer.

In this dissertation, we propose thermal-aware file and resource allocation policies to reduce the cooling cost of data centers. We first propose a thermal-aware file assignment policy -TIGER- to make file assignments in distributed storage clusters, followed by out resource allocation policy- TASH, which addresses the issue of thermal management in specific framework (Hadoop). Our proposed thermal-aware policies utilize nodes' contributions towards heat re-circulation in data centers while making file and resource allocation decisions. The proposed policies make use of cross-interference matrix to calculate node's contribution in heat re-circulation. Our experimental results show that the proposed policies significantly reduce the cooling cost of data centers as compared to existing thermal-aware policies, and at the same time, maintains performance penalties within acceptable margin.

Acknowledgments

There is a long list of people who contributed in one way or the other during my graduate studies at Auburn. First of all, I would like to express my deepest appreciation to Dr. Xiao Qin for being a great advisor, an excellent teacher, and an awesome person. I got to learn a lot of things from him which not only helped me in improving my technical skills but also helped me in becoming a good person. Without his continuous guidance, creative vision, and constant supervision, this research would have never been possible. Every chapter of this dissertation is influenced by him in some way or another. It has been a great honor and pleasure for me to work under Dr. Qin's supervision.

I would also like to thank Dr. Ku, Dr. Baskiyar, and Dr. Biaz for serving on my dissertation committee, reviewing the original version of my dissertation, and giving insightful suggestions, which helped me improve many chapters in the dissertation.

I am indebted to Dr. Sushil Bhavnani, who served as Faculty advisor for Indian Student Association for all the three years I was in the executive committee. I got to learn many things from him - accepting new culture while not forgetting who we are, importance of punctuality, and planning big events - are few of them.

I would like to express my gratitude to Dr. Ji Zhang and Dr. Xunfei Jiang for helping me understand many technical aspects such as cluster setup, lab network topology and many more things at the beginning of my graduate studies. I would also like to thank Sanjay Kulkarni, for helping me understand the source code of Hadoop framework. Sanjay is also a very good friend and a peer reviewer. I am also thankful to all my colleagues at Computer Systems Lab, in particular, Tausif Muzaffar, Yuanqi Chen, Chetan Sonami for their help and suggestions.

I am deeply indebted to Vikalp Narayan and Aditya Singh - best friends and awesome roommates. They were always there to help me with all of my big decisions and have supported me in my emotional low points. I am grateful to Jasma didi, Bhumi ben, Avanti Kulkarni, Niranjana Aunty for ensuring that I missed none of the home food, and Sadhwi Ravichandran for offering me great suggestions in my social life and for South Indian culture and food.

I would also like to thank Vishal Kothari, Kunal Sevak, and Gayatri didi for playing a role of elder siblings. I must thank Vibudh Mishra, Adarsh Jain, Digvijay Gholap, and Mikhail Zade for helping me in the vital beginning of my life as a graduate student. In addition to this, I owe a debt of gratitude to my friends Nakul Kothari, Shantanu Deshpande, Vijith Beemireddi, Robin Muthukumar, Amey Rane, Prachi Sangle, Aditya Agrawal, Shubham Garg, Micah Bowden for all their support, laughs and great memories. I would also like to thank all ISA committee members.

Most of all, would like to express my deepest gratitude to my parents. You have sacrificed a lot so that I can pursue my dreams and words cannot express my feelings nor my gratitude for all your love and compassion. I am who I am, only because of you. My thanks also goes to my elder brother Abhay Chavan for being my brother, friend, teacher, advisor, and role model in my personal life. I must thank Nivas kaka, Sushma aatya, Mai aatya, Amey dada, Sachin dada, and Chaitrali tai for my unforgettable childhood memories.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Advent of applications such as YouTube and Facebook initiated a need of large scale data centers to support Internet-scale services [2]. These data centers house storage clusters as well as High Performance Computing (HPC) clusters. There is a dramatic increase in the processing and storage capacity of nodes in the data centers, as a result of continuous improvements in hardware technologies. This increased processing and storage capacity comes at a cost of high power density resulting in thermal emergencies in the data centers. Unfortunately, these thermal emergencies have to be resolved using sophisticated and expensive cooling infrastructure, which parts substantially towards annual operational cost of the data centers [3].

We believe that an efficient way to address these thermal emergencies is to proactively target the cause of the problem; heat re-circulation; and attempt to avoid the thermal emergencies in the first place. Minimizing the heat re-circulation in the data centers has proven to be crucial in avoiding thermal emergencies. In particular, dynamic workload placement strategies that consider heat re-circulation in the data center while making workload placement decision can avoid thermal emergencies more effeciently at very low cooling cost. The objective of this dissertation is to investigate workload placement strategies to reduce the cooling cost in data centers housing both storage clusters and HPC clusters.

This chapter first presents the problem statement (see Section 1.1). In section 1.2, we describe the scope of this research. Section 1.3 highlights the contributions of this dissertation, and section 1.4 outlines the dissertation organization.

1.1 Problem Statement

In this section, we first summarize emerging trends in large-scale data centers. Section 1.1.2 provdies an overview of several challenges that big data industries are facing from the perspective of thermal emergencies and cooling cost. Finally, section 1.1.3 presents the initial motivation for the dissertation research by illustrating the limitations of existing solutions.

1.1.1 New Trends

Data centers (e.g., Facebook's Oregon data center) house thousands of servers arranged in the form of clusters. Initially, these data centers are meant to support High Performance Computing application such as medical imaging, DNA mapping, heavy bio-scientific simulations, and business operations for the organizations. In order to provide high processing power required to run such applications, the HPC clusters consist of computational nodes comprising high power processors and few disks. They are also known as clusters of commodity machines.

In the last decade, with the emergence of social media; an invention that reformed data mining and business intelligence; huge amount of data is being created and used everyday. To store this terabytes of data being created everyday, current data centers house storage clusters comprising of storage nodes. Typically, these storage clusters stores humongous amount of data, usually in redundant replicas to provide data reliability and to improve data processing efficiency by offering parallel access to the data. Each such data node typically consists of low power processors and RAID arrays containing multiple (e.g., 4 to 32) disks.

Current trends show exponential growth in both compute and storage capacity with the emergence of blade servers and disks arrays. However, the increased compute and storage density comes at the cost of associated power and heat density [4]. On one side, due to ever increasing storage and processing capacity requirements, data centers are growing in

size. On the other hand, due to deployment of power hungry hardware components and decreasing form factor, power density of data centers is increasing drastically [4][5].

As a result of these technological advancements and growing needs of data centers, huge amount of power is being consumed by the data centers and the rate of increase in power consumption is escalating per year. According to recent report, electricity used in global data centers in 2010 likely accounted for between 1.1% and 1.5% of the total electricity used. For U.S., this number was between 1.7% and 2.2% [6]. The annual power costs of the data centers are in the order of millions of dollars [7]. High operation cost of data centers are attributed to two sources 1) power consumed by the nodes in the data centers, 2) power consumed by the cooling systems in the data centers. Previous research have confirmed that the cooling cost is almost half of the total energy cost of a data center [8].

In 2004, Google introduced MapReduce, a simplified programming model and an associated implementation for processing and generating large data sets[9]. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines [9].

It is been originally optimized for large batch jobs such as web index construction. However, a use case where MapReduce being shared by multiple users and runs a mix of long batch jobs and short interactive queries over a common data set has soon become a common picture in most of the data centers [10]. Apache Hadoop - an open source implementation of MapReduce framework- has now become the platform of choice for developing large-scale data intensive applications [11].

1.1.2 Thermal Emergencies in Data Center

Recall that the power and heat density in the data centers are growing at exponential rate. In a typical data center layout, the hot air exiting from the outlet of the node re-circulates through the data center and contributes to the inlet temperature of other nodes in the data center. The heat re-circulation causes the inlet temperatures of some nodes in the data center to rise above the allowed operational temperature (also known as *Redline temperature*). The nodes with inlet temperatures higher than the redline temperature are called as hot spots and should be avoided as the operation outside the accepted temperature exponentially decreases the reliability of hardware [12][13][5].

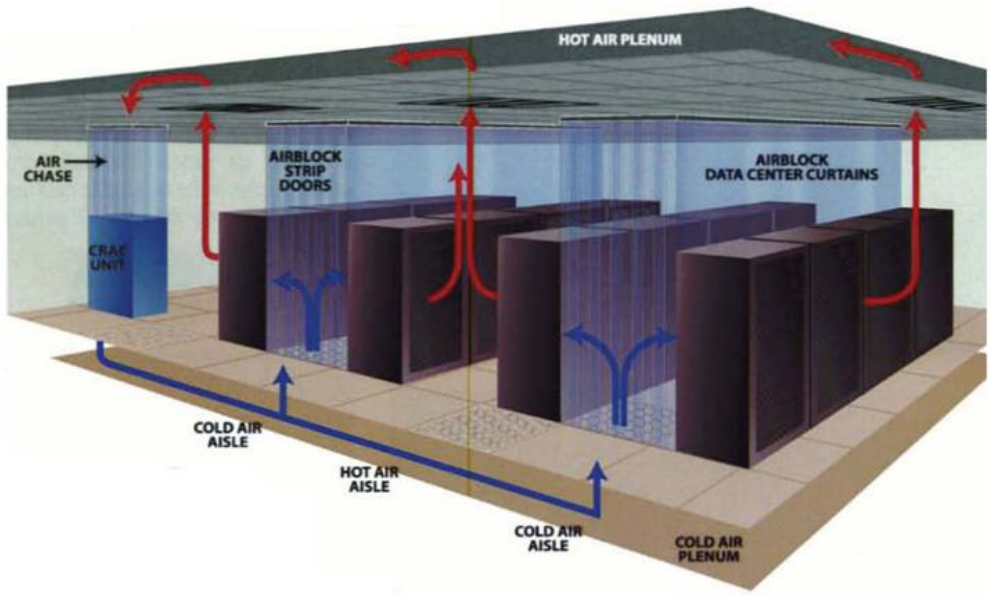


Figure 1.1: Data center layout with hot and cold aisles and CRAC.

Situations like creation of hot spots are called for thermal emergencies and need to be resolved as early as possible in order to improve the data center reliability and to reduce the failure rate. In order to keep the inlet temperatures below the redline temperature, *Computer Room Air Conditioners* (CRACs) are deployed in the data center [1]. Typical

data center layout with CRAC is depicted in FIG.1.1.2. The racks are arranged in alternate cold and hot aisles with all the inlets of the nodes facing towards the cold aisle and outlets facing hot aisle. The hot air is extracted from the ceiling of the data center, passed through CRAC, where it is cooled down and supplies thorough the vents from the raised floor. Recall that the cooling cost is almost hald of the total energy cost of the data centers. Therefore, *in today's power dense data centers it is very important to reduce the cooling cost in order to control the operational cost of the data centers.*

Heat re-circulation is not only the cause we need to have a cooling system but it is also the reason for low cooling system efficiency. Heat re-circulation in the data center results in the increased inlet temperatures and may cause creation of hot spots [14]. A large amount of heat re-circulation forces computer room air conditioners to continuously work at lower supply temperatures, decreasing the efficiency of the CRAC. In order to improve the cooling system efficiency, we should run the CRAC at higher supply temperature (see section 3.2). Therefore, *it is very important to reduce the heat re-circulation, which is the cause of low cooling efficiency, in order to reduce the cooling cost.*

The best way to minimize the heat re-circulation for a given data center layout is by controlling the power profile of the data center [1]. A workload placement algorithms can be optimized to distribute the workload among the available nodes in a way to reduce the power consumption of particular nodes, in order to reduce the heat re-circulation [1] [15]. Recall that data centers house two different types of clusters: HPC clusters and storage clusters. Each type of clusters service different workloads and have different types of nodes.

In the case of HPC clusters, each node consists of a chassis containing multiple blade servers. A blade server is essentially a server on a card and represents further minimization and modularity from the rack optimized form factor [4]. Each blade server consists of very powerful multicore, multiprocessors and few (typically 2 to 4) disks. In such setups, processors consume most of the power drawn by the server. As compared to processors, disks consume very little power and can be neglected from heat re-circulations point of view. The

power consumption of the processor is linearly proportional to the processor's utilization. Therefore, *in order to control the power consumption by the node containing multiple blade servers, it is important to control the utilization.*

Unlike blade servers, storage clusters consist of storage nodes. The main task of storage nodes is to store and retrieve large amount of data as requested by the client. In order to provide such functionality, a storage node requires huge storage capacity and little processing capabilities. Therefore, a storage node contains low power processor and a disk sub-system comprised of multiple disks (typically 4 to 32). Modern storage systems account for almost 27% of the total energy consumption and have non-negligible impact of the heat re-circulation [16] [17]. Recent study indicates that data placement has impacts on I/O workloads, which in turn affect heat dissipation in the node [18]. Therefore, *controlling power consumption by disks sub-systems is crucial in minimizing heat re-circulation in storage clusters.*

An important point to note here is, unlike processors, which exhibits linear correlation between power consumption and utilization, disks have three different modes of operations, namely, active, idle, and sleep mode, each of which has specific power needs. In the case of storage nodes, lowering the power needs can be achieved by transitioning disks into the sleep mode when large idle periods are observed [19]. Disk idle periods largely depends upon both the file assignment and access pattern of data placed on the disks. Therefore, in order to file assignment policies must be incorporated with other I/O techniques (e.g., prefetching and write off-loading) [16] [20], which are applied to generate large idle periods. Therefore, *data placement policy must take thermal profile into account while making file allocation decisions.*

In most of the today's storage systems, file assignment decisions are made on the fly, i.e., decisions are made for the files (or data) as they are created and not in the batch process. Though it represents a scenario, where a separate decision is made for each files, as these systems usually consist of thousands of storage nodes servicing hundreds of client, the file

placement problem can be view as *a problem of assigning m number of files on D available disks residing in N nodes.* To support our argument, we present following two examples:

Example 1: Hadoop Distributed File System or HDFS is one of the most popular distributed file systems developed in today's data centers [21]. HDFS offers scalable and reliable data storage for hosting terabytes of data and supporting an enormous number of clients. In HDFS, data placement is accomplished by the NameNode, which is the centerpiece keeping the directory tree of all files in the file system. Therefore, NameNode manages a pool of files to be allocated in Hadoop clusters.

Example 2: Parallel File Systems (e.g., GPFS [22] and PVFS [23]) have been employed in many of the world's largest supercomputers. To achieve high I/O performance, parallel file systems partition large files into small fragments placed across multiple disks or storage nodes. Therefore, it can be assumed that considering each partition as a separate file, these file systems address the problem of assigning m file to D disks.

In the first half of this dissertation, we present a thermal aware file assignment policy to reduce the cooling cost by addressing the problem of assigning m files to D disks residing on N nodes in the storage clusters in a way to reduce the heat re-circulation. In the next half, we present thermal aware scheduling policy to reduce the cooling cost of the data center housing Hadoop clusters.

1.1.3 Limitations of Existing Approaches

A large body of work can be found in the literature that addresses the issue of saving energy in data centers by improving energy efficiency of nodes in the data centers. At node level, Springer *et al* proposed a policy to conserve energy consumed by the node by using Dynamic Voltage and Frequency Scaling (DVFS) technique, in which, a working frequency of the processor is changed based on workload on system [24]. The right sizing technique offers energy conservation at data center level, by adjusting the size of the data center according to workload demands [25]. On the other hand, many researchers focused on improving storage

system energy efficiency [26] [27]. Although all these techniques affect heat re-circulation as they control the power consumption by the nodes in the data center, they cannot guarantee to reduce the heat re-circulation as they do not consider thermal profile while attempting to improve the energy efficiency of the data nodes. This dissertation focuses on the improving the energy efficiency of the cooling system.

In the last decade, many researchers have have focused on addressing an issue of thermal management in the data centers. The research in the area of thermal management in data centers can be divided into two sub-streams: thermal modeling and thermal management. Former stream includes developing different models to capture the thermal profile of the data centers, while later focuses on designing workload placement strategies to control the thermal profile of the data center. For example, Moore *et al* developed a simple yet effective method to infer the detailed model of thermal behaviors within a data center from a stream of instrumentation data [28]. Many researchers have proposed novel thermal aware workload placement strategies to reduce the cooling cost of the data centers [1] [15]. Although these strategies offer significant improvements in cooling efficiency of the data centers, most of them focus on HPC clusters, where processors are the major power consumers. Recall, disk sub-systems consume 27% of the total energy consumption of data centers and have non-negligible impacts on cooling cost of the data center. Therefore, while addressing the issue of thermal management in storage clusters, impact of disks sub-systems cannot be ignored. This dissertation investigates plausible way to conserve the cooling energy of storage clusters by offering thermal aware file assignment.

1.2 Scope of the Research

This dissertation research focuses on thermal-aware workload placement strategies for data centers housing storage clusters as well as HPC clusters.

We have proposed thermal-aware workload placement scheme to reduce cooling needs of data centers. This approach is able to reduce the cooling cost while keeping the performance

penalties to minimum. Importantly, we have extended the research efforts in the area of data center thermal management in three different ways. First, we have incorporated modified power consumption and heat re-circulation model to characterize impact storage clusters in the thermal profile of data centers. Second, we have proposed a thermal-aware file assignment policy to reduce cooling cost of data centers through heat re-circulation minimization. Finally, we have proposed a thermal-aware resource allocation policy for data centets housing Hadoop clusters.

1.3 Contributions

To address the challenges of thermal-management in a diversified cluster environment, our research investigates dynamic file and resource allocation policies that are capable of achieve high cooling energy conservation in both storage and HPC clusters. In what follows, we list the key contribution of the dissertation.

- **Thermal Models for Data Centers:** We have extended existing thermal models presented by [29], to characterize the impact of storage nodes in the heat re-circulation of the data centers. We have also extended the existing model to incorporate power consumption by disks and to calculate contributions of each node in the heat re-circulation of the data centers.

- **A New File Assignment Policy:** We hve developed a novel, generic file assignment policy to reduce the cooling cost of the data centers. The proposed strategy aims at minimizing the maximum inlet temperature of nodes in the data centers through controled power distribution among data center nodes.

- **A New Resource Allcation Piolicy:** We have proposed a thermal-aware resource allocation policy to control heat re-circulation in the data center housing Hadoop cluster. The proposed strategy dynamically allocates resource on the nodes based on node's contribution in heat re-circulation and resource demand in the system.

1.4 Organization

This dissertation is organized as follows. In Chapter 2, existing research efforts in the area of thermal management are briefly reviewed.

In Chapter 3, we have developed different models to characterize power consumtion of storage nodes, heat re-circulation, and cooling cost of data centers.

In Chapter 4, we have developed a novel thermal-aware file assignment policy to reduce cooling cost of the storage clusters in the data centers.

In Chapter 5, we describe our resource allocation policy to reduce the cooling cost of the data centers housing Hadoop clusters.

In Chapter 6, we present our experimental results to evaluate the proposed file and resource allocation policies.

In Chpater 7, we provide approaches to improve the proposed policies and to extend them for legacy framework such as MR1.

Finally, Chapter 8 concludes the dissertation by summarizing the main contributions of this dissertation and providing details on future directions for the research.

Chapter 2

Related Work

Exponentially increasing operational cost is a growing concern for data centers offering enormous storage and computing capabilities, and a variety of techniques have been proposed to improve the energy efficiency of the data centers. This chapter briefly presents previous approaches found in the literature that are most relevant to our research from three perspectives - 1) energy efficiency of data centers, 2) thermal efficiency of data centers, and 3) energy efficiency of MapReduce clusters.

2.1 Energy Efficient Data Center Nodes

In general, energy conservation techniques control the workload distribution when the system is within appropriate workload range in order to achieve energy benefits enduring performance penalties. There is always a trade-off between energy savings and performance of the system, and one should assure that the performance penalties are within acceptable range when aiming for energy conservation. This section presents a summary of work related to conserving energy consumption by nodes in the data center. Specifically, we describe the important features in a wide range of energy saving techniques. These approaches can be categorized into two main categories: energy saving in computational nodes (see section 2.1.1), and energy savings in storage nodes/disks subsystems (see section 2.1.2).

Figure 2.1 depicts the high level taxonomy of the approaches to the problem of energy conservation in the data centers. Each category can be further divided based on certain specific attributes to a group of schemes. It can be observed from the figure that this dissertation focuses on the issue of thermal management of data center through job scheduling and data placement.

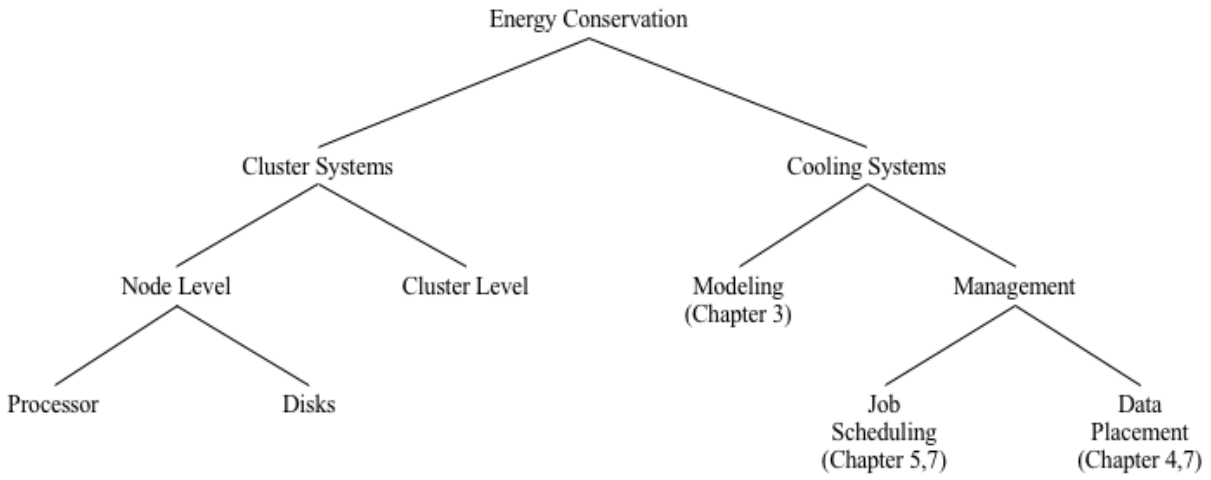


Figure 2.1: A simplified taxonomy of the approaches to the energy conservation problem in data centers.

2.1.1 Computational Node Energy Savings

A variety of literature is available on the issue of energy conservation in comptational nodes. One of the most popular approaches for energy conservation is *Dynamic Voltage and Frequency Scaling (DVFS)*. Dynamic voltage scaling takes advantage of the fact that lowering voltage can reduce power quadratically to reduce the enrgy consumption [30]. A voltage scheduler is deployed, which decides when and to what level to change the voltage and operational frequency of the processor based on workload conditions [31]. A rich set of policies for the voltage scheduler have been proposed ranging from simple *earliest deadline first (EDF)* [32] to sophisticated adaptive policies based on recursive learning and empirical studies [33].

Shin and Choi proposed a power conscious version of fixed priority scheduling to yield power reduction by the processor [34]. This method exploits slack time inherently present in the system as well as arising from variations of execution time. The scheduler is invoked

when the run queue is empty and based on the system conditions the processor will either be switched to power-down mode if there is no current active jobs or adjust the operational frequency such that the current active job finishes at its deadline or at the release time of the next job.

In another research study, Quan and Hu presented two novel energy efficient DVFS policies for real-time systems with job having pre-defined release time, deadlines and required number of CPU cycles [35]. The policies offer two fold benifits. First, it provides the minimum constant voltage (or speed) needed to complete a set of jobs as constant voltage tends to result in low power consumption. Second, it produces the voltage schedule which results in even lower energy consumption as compared to using the minimum constant voltage and shutting down the system when it is idle.

Weirman *et al* studied the problem of optimally scale the speed to balance the trade-off between mean response time and mean energy consumption under processor sharing environment [36]. They provided bounds and asymptotic for the speeds used by optimal dynamic speed scaling scheme, and showed that it significantly improves the robustness of the system to bursty traffic along with energy efficiency improvements. Springer *et al* also proposed a variation of DVFS scheme. They proposed a policy to create a frequency schedule based on a target program and constraints on power consumption in the cluster to minimize execution time while staying within the power constraints [24].

In the right-sizing technique offered at the data center level, the size of a data center is adjusted according to workload demands [25]. This approach conserves energy by turning as many servers as possible to the power-saving mode while maintaining desirable system performance. The right-sizing scheme allows requests to be dispatched to a small group of servers under light loads while keeping a large number of servers in the power-saving mode [37][38][39]. All these approaches focus on energy conservation in computing nodes. On the other hand, our study focuses on thermal impact of workload placement on cooling energy conservation.

2.1.2 Energy Savings in Storage Nodes

In the era of internet, enormous amount of data is being generated every day. Data centers usualy deploy a storage clusters to manage such amount of data. Since, power consumption by storage clusters is a problem impacting total cost of owernship, extensive amount research efforts have been dedicated to provide energy efficient storage systems [20] [26] [27] [40] [41]. Most of these techniques try to create longer idle preriods for the disks so that the disks can be spun down to reduce power consumption by the disk subsystems. In this section, we will take a closer look at some of such techniques.

Write off-loading is a technique used to create longer idle periods by allowing write requests on spun down disks to be temporarily redirected to persistent storage elsewhere in the data centers. Naraynan *et al* provided a policy that can off-load write requests transparentlly and efficiently at the block level, withot sacrificing cosistency and failure resilience [20]. Another set of techniques use buffer disks to allow access to the popular data while other disks can be spun down to save energy. Zong *et al* developed a heat-based dynamic caching policy (a.k.a. *BUD*) to minimize energy consumption in storage systems. BUD strives to allocate as many requests as possible to buffer disks, thereby keeping a large nuber of idle data disks in low-power mode [27]. Zhu *et al* proposed an on-line power-aware cache replacement policy called *PB-LRU*, to further improve the caching policies used to conserve the energy consumption by disks [26].

Papathanasiou and Scott presented a new rule for prefetching and caching- mechanisms previously designed to maximize performance- to maximize power-down opportunities by creating an access pattern characterized by intense bursts of activity separated by long idle times [41]. On the other hand, *PRE-BUD* uses aggressive prefetching along with buffer disk architecture to improve the energy efficiency of disks subsystems [40]. Many other prefetching and caching techniques use prediction or machine learning techniques to offer energy savings in storage systems (e.g.[42][43][44]). All the above mentioned strategies focused on energy consumption by the nodes in the data centers and do not pay attention to its impact on heat

re-circulation and thermal profile of the data centers. On the other hand, in this study we focused on the thermal impact of the power consumption by nodes, and have used energy efficiency policies to control heat re-circulation in the data centers.

2.2 Thermal Efficiency of Data Centers

The research in the area of thermal management in the data center can be categorized in two main streams: first stream of studies focus on deriving models to characterize the heat re-circulation and thermal profile of the data centers whereas second stream aims at developing the policies to control the thermal profile in order to reduce the cooling cost of the data centers. Thermal models include C-Oracle [5], ThermoCast [45] and Cross-Interference Matrix [29]. There are several thermal management policies such as XInt [1], ZBD, MinHR [15], Weatherman [28]. This section sheds some light on these models and techniques.

2.2.1 Thermal Modeling

Thermal profile can be created by using Computational Fluid Dynamics (CFD) simulations, which are time consuming and very expensive in terms of processing power. To overcome this drawback, thermal models are developed to captured the thermal profile of the data centers. From this thermal profile, many useful information such as the location of hot and cold spots, contribution of the nodes in the heat re-circulation in the data center can be derived. In this section we take a look at few such models.

Ramos and Bianchini developed a software infrastructure called C-Oracle that dynamically predicts the temperature and performance impacts of different thermal management reaction into future [5]. Such prediction facilitates the employed thermal management policy to select the best reaction at each point. C-Orace calculates heat transfers between nodes by using power consumption of the nodes. While C-Oracle uses power consumption of nodes, *ThermoCast*; a cyber-physical forecasting model proposed by Lei *et al*; makes use

of continuous stream of temperature and airflow measurements to predict the temperatures of surrounding servers in the data center [45]. ThermoCast then uses predicted temperatures to identify future potential overheating events. Tang *et al* derived a matrix called Cross-Interference Matrix in order to model the heat re-circulation in the data center [29].

All the above mentioned models either uses linear power model or temperature measurements through temperature sensors. Linear power models assumes that the processor's power consumption is linearly proportional to its utilization. While in computation nodes, the processor is the major power consumer, same cannot be said in terms of storage nodes. In storage nodes disks sub-systems containing multiple disks are the major consumers of power and disks do not follow linear model for power consumption. Therefore, our study focuses on the power utilization by disks and its impact on the heat re-circulation of the data centers.

2.2.2 Thermal Management

Thermal management policies aim at making use of existing models to capture the thermal profile of the data center and control the workload or power distribution in the data center to reduce the cooling cost. Thermal-aware workload placement strategies were proposed in recent studies, which indicate that energy efficiency of the CRAC can be improved by reducing the peak inlet temperature of nodes in data center. XInt is one of such policies, which manages workload in a way to reduce the maximum inlet temperatures [1]. Moore *et al* designed the ZBD scheme that uses poaching where the effect of heat re-circulation is observed to reduce the inlet temperature of nodes in a data center [15].

The MinHR approach manages workload in a way that each pod in a data center generates the same amount of heat, minimizing the heat re-circulation [15]. Wang *et al* have proposed a different approach, where jobs are sorted in descending order of their hotness (i.e., heat generated by jobs) and nodes are sorted in ascending order of their inlet temperatures,

then, hottest jobs are placed on the coldest nodes [46]. Abbasi *et al* addresses the issue of temperature profiles and workload distribution in the context of Internet data centers [47].

2.3 Energy Conservation in MapReduce Clusters

In recent years, use of distributed framework such as MapReduce to support the data intensive applications running in the data centers is increasing dramatically. Therefore, much attention has been paid to develop energy-efficient data centers running Hadoop and MapReduce clusters. For example, Leverich *et al.* introduced the concept of *Covering Subset* to assure 100% data availability when a cluster is running on reduced capacity to save energy [48]. Lang *et al.* proposed the All-In Strategy (AIS) that drives a cluster to run in its full capacity and brings down the entire cluster after completing jobs [49]. GreenHDFS - an energy-efficient distributed file system - divides a cluster into hot zones and cold zones based on access frequency of data [50].

We summarize the differences between TIGER and the existing solutions in Table 2.1.

Table 2.1: Comparison of TIGER and the existing solutions

Techniques	Thermal Model	Thermal Management	Job Scheduling	File Placement
Thermal Characteristics [29]	✓	—	—	—
XInt [1]	—	✓	✓	—
C-Oracle [5]	✓	—	—	—
MinHR [15]	✓	✓	✓	—
TIGER	✓	✓	—	✓

2.3.1 Energy Efficiency

In recent years, cloud computing and MapReduce frameworks have become standard data center frameworks. Number of algorithms have be proposed and extensively evaluated to reduce the energy consumption by the clusters deploying these frameworks. Kaushik *et al* present Lightning- an energy conserving, self adapting Commodity Green Cloud Storage, which uses data-classification driven data placement algorithm. It dynamically configures data center servers into hot and cold zones in order to achieve energy conservation while

maintaining data availability [51]. GreenHDFS [50] uses similar approach to provide energy efficient storage for Hadoop framework.

GEMS is an online energy minimization path algorithm used to schedule MapReduce tasks in cooperation with sleeping policies on serves as well as switched [52]. Zhu *et al* proposed an energy aware scheduling algorithm; EARH; which employs a rolling-horizon optimization policy to achieve energy conservation in cluster servicing real-time, aperiodic, independent tasks [53]. *ECCO* is another distributed system, which dynamically redefine the set of active resources of a data center with a purpose of drastically reducing energy cost without degrading the performance of the system within acceptable range [54]. However, all the above strategies focus of conserving energy consumed by the nodes in the data center. Although, the power consumption of the nodes affect cooling cost of the data centers, different set of parameters must be considered to achieve drastic energy conservation in cooling system of the data center.

2.3.2 Thermal Efficiency

Number of approaches to manage thermal emergencies in data centers is discussed in section 2.2. We mention few more studies in this section as these are closely related to our study. Goiri *et al* proposed a run time system, CoolAir, that embodies different strategies to limit absolute temperature, temperature variations, humidity, and cooling energy in free-cooled data centers [55]. Unlike TASH, CoolAir makes scheduling decisions based on web data and data collected from the sensors from all the nodes. Tang *et al* [56] proposed a data locality aware power controller with thermal consideration to dynamically switch the power state and to switch the executing frequency of each server. While it changes the power state and executing frequency of each server to control the power profile of the data center, TASH employs resource allocation policy to manage power profile. GreFar is an online algorithm, which addresses the problem of scheduling batch jobs to multiple geographically distributed data centers [57]. It delays job execution until sufficient number of jobs are in the queue

or power is sufficiently cheap, in order to maintain the maximum server inlet temperature constraints. It decides when to schedule a job as opposed to TASH, which decides how many containers to run.

Another interesting area of research focuses on using "Green Energy" to power the data centers in an effort to reduce CO_2 emission of data centers. Many of such studies proposed methodologies to predict the amount of Green energy (photo-voltaic and solar energy) that will be available in near future and schedules jobs to maximize the green energy consumption. GreenHadoop [58] proposes such strategy for Hadoop framework, while GreenSlot [59] proposes parallel batch job scheduler for Green data center. Number of scheduling policies have been proposed to offer optimal geographical load balancing aiming to minimizing the brown energy (fossil fuel energy) consumption of geographically distributed data centers [60] [61] [62]. JouleMR is another such cost-effective and green-aware data processing framework, spefically focusing on data processing by Hadoop cluster [63]. Although, all of these strategies maximize the use of renewable energy, they rarely focus on conserving cooling cost.

Chapter 3

Modeling

In this section, we describe three models used o characterize heat re-circulation (see section 3.1), cooling cost of a data center (see section 3.2), and power consumption (see Section 3.3). These models provide a foundation for building our thermal-aware file assignment scheme to reduce cooling cost.

Table 3.1 summarizes all the symbols used in this paper.

3.1 Heat Re-circulation Model

Heat re-circulation is one of the main reasons for creating hot spots in a data center; hot spots adversely affect the cooling cost of data centers [15]. Therefore, it is very important to characterize the heat re-circulation and its impact on inlet temperatures of nodes.

A number of approaches have been designed to characterize the heat re-circulation in data centers [45][5][29]. All these approaches are well investigated and well validated and they predict the inlet temperature of nodes with reasonable accuracy. We start from the approach described in [29] and modify it to accommodate contribution of disks in data center heat re-circulation.

In a typical data center, hot air is extracted from the ceiling and cold air is supplied from a raised floor [64]. The outlet heat rate Q_i^{out} of chassis i is affected by chassis i's inlet heat rate Q_i^{in} and the heat generated by the chassis component. Thus, the outlet heat can be expressed as (3.1).

$$Q_i^{out} = Q_i^{in} + Q_i^{Node} \tag{3.1}$$

Table 3.1: Notations

Symbol	Definition
N	Number of nodes in the storage cluster
m	Number of files to be placed
s_k	Service time of k^{th} file
λ_k	Arrival rate of k^{th} file
D_i	Number of disks in i^{th} node
D	Total number of disks in the storage cluster
U_i^{Th}	threshold on the utilization of disks in i^{th} node
P_C	Power consumed by the nodes in the data center
P_i^{Node}	Power consumed by i^{th} node
P_i^{base}	Power consumed by the hardware other than CPU and disks (e.g. fans) of i^{th} node
P_i^{CPU}	Power consumed by CPU of i^{th} node
P_i^d	Power consumed by the disks subsystem of i^{th} node
$P_{i,j}^d$	Power consumed by j^{th} disks of i^{th} node
P_{AC}	Power consumed by the cooling system of the data center
ρ	Air density (typical value 1.19 Kg/m^3)
c_p	specific heat of air (typical value $1005 JKg^{-1}K^{-1}$)
a_i	Air flow rate of i^{th} node (typical value 0.2454 m^3/s)
$\alpha_{i,j}$	Cross-interference co-efficient from node i to node j
S_i	Sum of all the cross-interference coefficients of i^{th} node, normalized over sum of all the cross-interference coefficients of all the nodes in the data center
Q_i^{in}	Inlet heat of i^{th} node
Q_i^{out}	Outlet heat of i^{th} node
Q_i^{Node}	Heat rate of i^{th} chassis
T_i^{in}	Inlet temperature of i^{th} node
T_i^{out}	Outlet temperature of i^{th} node

where Q_i^{Node} is heat rate of chassis i. Because power drawn by the chassis is dissipated as heat and according to the *law of energy conservation*, Q_i^{Node} can be measured as the amount of power consumed by the chassis per unit time [29]. Also, by the *law of thermodynamics*, $Q_i^{in} = \rho c_p a_i T_i^{in}$ and $Q_i^{out} = \rho c_p a_i T_i^{out}$. Therefore, substituting the values in (3.1):

$$\rho c_p a_i T_i^{out} = \rho c_p a_i T_i^{in} + P_i^{Node} \tag{3.2}$$

where ρ, c_p, a_i are thermo-physical values. Specifically, ρ is the density of air measured in grams per cubic meter; the typical value of ρ is 1.19 Kg/m^3. c_p is the specific heat of air measured in joules per gram Kelvin; the typical value of c_p is 1005 $JKg^{-1}K^{-1}$. a_i is the airflow rate of node i; a_i can be measured in cubic meters per second. A typical value of a_i is 520 CMF, which equals to 0.2454 m^3/s [1]. These values can be expressed as

thermodynamic constant K_i. Two similar models for outlet temperatures were validated in the recent studies [5][1].

The hot air exiting from the outlet of a chassis recirculates in the data center and contributes to the inlet temperature of other chassis in data centers including itself. There are a few efficient ways of characterizing the effect of heat re-circulation in a data center. The best way is to use *Computational Fluid Dynamics* (CFD); however, it is computationally expensive and CFD models take a long time to produce results. Some other models, such as Supply Heat Index (SHI) and Heat Re-circulation Index (HRI) [15][28], characterize heat re-circulation in different ways. In this study, we use a similar model that characterizes heat re-circulation as cross-interference coefficient matrix $A_{n\times n} = \{\alpha_{ir}\}$, where α_{ir} denotes the fraction the outlet heat of ith node contributes to the inlet temperature of rth node [29]. Therefore, the inlet heat at node i can be calculated as:

$$Q_i^{in} = \sum_{r=1}^{n} \left(\alpha_{ri} \left(Q_r^{in} + P_r^{Node}\right)\right) + Q_i^{sup} \tag{3.3}$$

where Q_i^{sup} is the amount of heat supplied by the air blown by the cooling system. It can be calculated by using (3.4).

$$Q_i^{sup} = \rho c_p \left(a_i - \sum_{r=1}^{n} \alpha_{ir} a_r\right) T^{sup} \tag{3.4}$$

Let $\overrightarrow{T^{in}} =< T_1^{in}, T_2^{in}, ..., T_n^{in} >$ be a vector of inlet temperatures of all servers in a data center. We denote $\overrightarrow{P} =< P_1, P_2, ..., P_n >$ as a power consumption vector of all the servers. Let us organize the thermodynamic constants K_i into an n by n diagonal matrix $K = diag(K_1, K_2, ..., K_n)$. Thus, we can express the inlet temperature vector as a function of the cooling supply temperature $\overrightarrow{T^{sup}}$, thermodynamic constant matrix K, and cross-interference coefficient matrix A (see details in [1]).

$$\overrightarrow{T^{in}} = \overrightarrow{T^{sup}} + [(K - A^T K)^{-1} - K^{-1}]\overrightarrow{P}. \tag{3.5}$$

Equation (3.5) suggests that supply temperature T^{sup} has a direct and measurable impact on inlet temperature of nodes in data centers. Intuitively, inlet temperatures can be lowered by reducing the supply temperature. A reduction δT in supply temperature is derived from the maximum value (i.e. $\max(T_i^{in})$) of all the inlet temperatures and redline temperature $T^{redline}$. Thus, the supply-temperature reduction δT can be expressed as $\delta T = max(T_i^{in}) - T^{redliine}$ [1].

3.2 Cooling Cost Model

Heat re-circulation and node power consumption lead to an increase in inlet temperature. To control the raised inlet temperatures, a cooling system is applied. Temperature of the air supplied by the cooling system is adjusted according to the maximum inlet temperature. Supply temperature T_{sup} affects the efficiency of the cooling system. The efficiency of a cooling system is quantified in terms of *Coefficient of Performance* (COP) [15][1] as:

$$COP(T_{sup}) = 0.0068T_{sup}^2 + 0.0008T_{sup} + 0.458 \tag{3.6}$$

The Coefficient of Performance can be used to calculate the power consumption by using the following equation:

$$P_{AC} = \frac{P_C}{COP(T_{sup})} \tag{3.7}$$

where P_C is the total power consumed by the storage nodes in a data center [1].

Previous studies demonstrate that as supply temperature T^{sup} of a cooling system decreases, cooling system's efficiency COP(T^{sup}) drops accordingly, which in turn increases the power consumption (P_{AC}) of the cooling system. Equation (3.5) and (3.6) immediately indicate that cooling cost can be significantly reduced by minimizing the inlet temperature of data nodes in a cluster. An efficient approach to managing inlet temperature is to control the power distribution among the date nodes. In other words, we handle the power distribution in a data center in a way to reduce the maximum inlet temperature without lowering cooling

system's supply temperature. To judiciously manage the power distribution in a data center, TIGER controls the workload distribution of data nodes, where power distribution largely depends on the utilization of the nodes.

3.3 Power Model

It is worth mentioning that equation (3.2) in section 3.1 makes use of the storage node's power consumption to derive the node's outlet heat, which in turn is used (see equation (3.5) and (3.5) in section 3.1) to calculate the inlet temperature of the node in a data center. Equation (3.3) and (3.5) suggest that for a storage cluster with a fixed physical layout, the power consumption of the cluster's storage nodes plays a vital role in affecting the inlet temperatures of the nodes. The TIGER algorithm judiciously makes the file placement decisions in a way to control $\overrightarrow{P}$, which is the power consumed by the storage nodes in the data center. In this subsection, we analyze; in detail; the factors affecting power consumed by the storage nodes.

The power consumption of node i (P_i^{node}) can be derived from a fixed amount of power P^{base} consumed by node i's hardware (e.g., fans) other than processor and disks, power P^{cpu} consumed by node i's CPU, and power P^d consumed by disks residing in the node. Thus, we can calculate P_i^{node} as:

$$P_i^{Node} = P_i^{base} + P_i^{CPU} + P_i^d \tag{3.8}$$

Most storage nodes contain multiple (e.g., 16) disks. The measurement of power consumption P_i^d in (3.8) is computed as a summation of all the disks equipped in storage node i.

$$P_i^d = \sum_{j=1}^{D_i} P_{i,j}^d \tag{3.9}$$

where D_i represents total number of disks in storage node i and $P_{i,j}^d$ is the power consumed by j^{th} disk in the node.

Two factors make it reasonable to model the power consumed by processors in storage nodes as a constant. First, low-power processors are used in the storage nodes to build energy-efficient storage clusters, as storage nodes are unlikely to perform computing-intensive tasks. Second, power consumption in storage nodes is dominated by a large number of disks rather than processors.

In what follows, we model the power consumption $P^d_{i,j}$ of disk j in i^{th} node. We denote the power consumed by a single disk in the active, idle, and in sleep mode as $P^{d,active}$, $P^{d,idle}$, and $P^{d,sleep}$, respectively. Power overhead is incurred when disks are transitioning among the mode (e.g., from the sleep mode to active or vice versa). We denote the power required to spin down a disk as $P_{S_{down}}$ and power needed to spin up a disks as $P_{S_{up}}$. Given a time interval T, let $t^{active}_{i,j}$, $t^{idle}_{i,j}$, and $t^{sleep}_{i,j}$ represent time periods when disk j in node i is active, idle, and sleep, respectively. We denote $N^t_{i,j}$ as the number of power-state transitions. Now, we model the disk power consumption $P^d_{i,j}$ as:

$$\begin{aligned} P^d_{i,j} &= \frac{1}{T}\Big(t^{active}_{i,j} \times P^{d,active}_{i,j} + t^{idle}_{i,j} \times P^{d,idle}_{i,j} \\ &\quad + t^{sleep}_{i,j} \times P^{d,sleep}_{i,j} + \frac{N^t_{i,j}}{2}\left(P_{S_{down}} + P_{S_{up}}\right)\Big) \end{aligned} \tag{3.10}$$

Chapter 4

Thermal-Aware File Assignment in Storage Clusters

TIGER faces a unique challenge of reducing cooling costs by placing files to minimize heat re-circulation. File placement decisions made by TIGER are guided by a disk utilization threshold. Furthermore, TIGER is focused on the issues of load balancing across all disks as well as minimizing variance of the service time at each disk.

In this chapter, we first present the basic idea behind the TIGER algorithm with two scenarios in which TIGER can be ivoked (see Section 4.1). Then, we discuss; in detail; how TIGER calculates disk utilization thresholds based on the workload and contribution of nodes in the heat re-circulation in data centers (see section 4.3). Finally, we present our file assignment algorithm in section 4.4.

4.1 Overview

TIGER is a software infrastructure that allows storage clusters to achieve a thermal friendly, low power, and scalable I/O architecture in future data centers. Fig. 4.1 outlines the design of the TIGER framework, which includes clients that issue file requests, a file-access-pattern monitor that collects file access and temperature information, meta-data manager that organizes file attributes (e.g., permissions, namespace, and disk space quotas), a request handler that receives file-assignment requests from the client and the monitor, and the file placement policies that make important file-assignment decisions. The meta data manager, the file placement policies, and the request handler are implemented in a storage cluster's master node, which coordinates the I/O activities of all the peer data nodes of the cluster.

The TIGER infrastructure can be applied in the following two use case scenarios to manage the file placement issues:

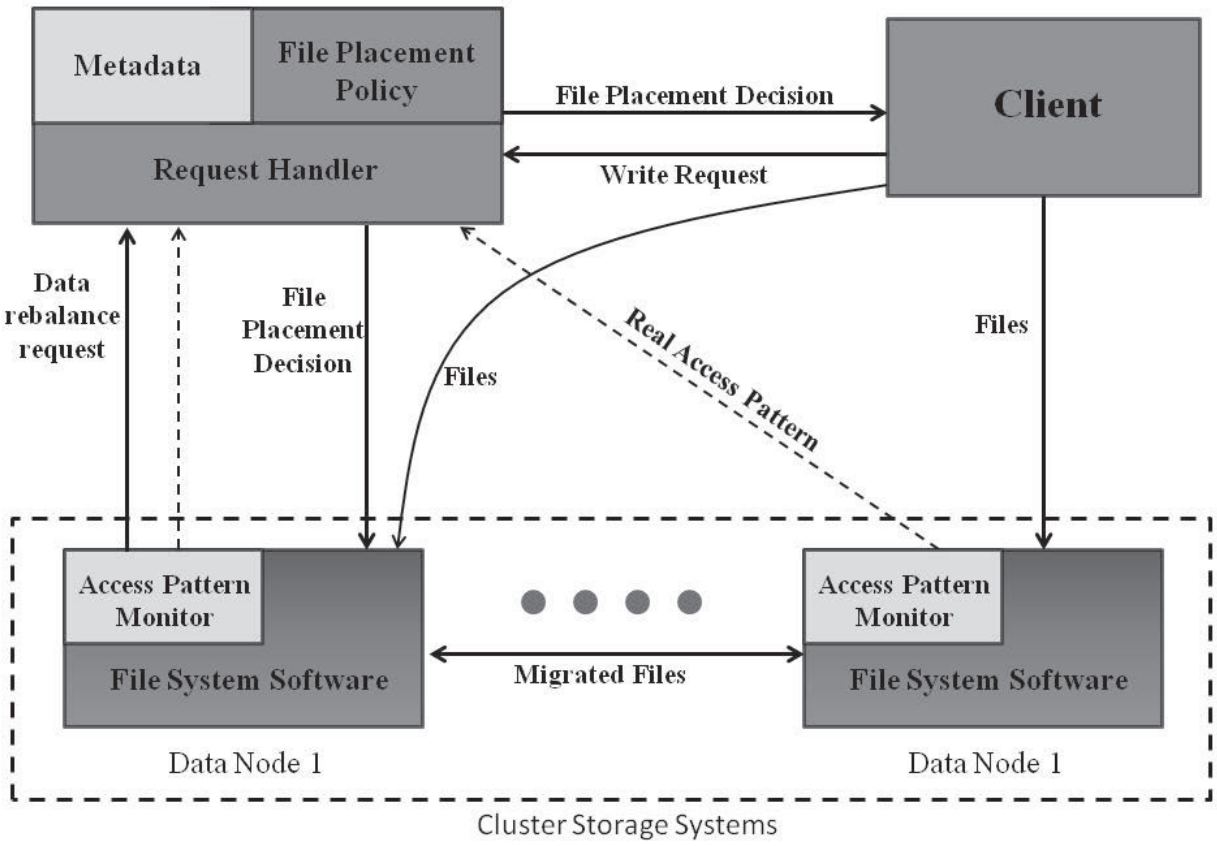


Figure 4.1: TIGER:System Overview

- the creation of new files and
- the migrations of existing files.

It is worth mentioning that Algorithm 1 is invoked to deal with both scenarios. From the perspective of TIGER, the difference between the two scenarios lies in the list of the files used as an input (i.e., file_info)to the algorithm. In the file creation case, the file list contains newly created files; in the file migration case, TIGER makes file assignment (a.k.a., file relocation) decisions for existing files. TIGER can be applied to make file assignment and relocation decisions in both file creation and migration scenarios.

4.1.1 Creation of New Files

In the file-creation case, TIGER assigns newly created files to data nodes according to a file-placement policy. A client follows four steps to create new files and store them into TIGER-enabled storage clusters. First, the client contacts TIGER where the request handler processes the file-creation requests. Second, the request handler drives TIGER's file-placement policy to make file assignment decisions using meta data and file access pattern information. Third, the file assignment decisions determined by the policy are delivered to the client. Last, the client directly transfers the newly created files to the storage nodes according to the file assignment decisions provided by TIGER. This procedure of handling newly created files is similar to many cluster file systems (e.g., HDFS [21]).

In a large-scale storage cluster servicing an enormous number of users, multiple file requests may be simultaneously issued from the clients. In such a case, the master node handles a pool of file requests in a batch manner. Thus, after receiving file assignment decisions from TIGER, the clients organize a pipeline to send the new files to the cluster's data nodes. TIGER can be incorporated into a parallel file system, where a large file is partitioned into multiple fragments. Before TIGER makes assignment decisions for file fragments, the parallel file system partitions large files into a set of fragments.

4.1.2 File Migration and Access Pattern Monitor

Considerable evidence shows that dynamically changing workload patterns (e.g., diurnal workload variations) are observed across all servers in a data center; more importantly, workload patterns shift during a certain period of time. Although the pattern changing rate differs from one data center to another, such a pattern shift widely exists in data centers. Access pattern information initially facilitated to TIGER making file assignment decision becomes obsolete over time; therefore, the utilization of nodes is dynamically changing.

There is a considerable demand for TIGER to yield file assignment decisions under dynamic workload conditions to maintain thermal benefits. To address this issue, TIGER

incorporates an access pattern monitor, which is a software module that runs on each storage node and continuously monitors access patterns of all files residing on the node. The monitor collects the metadata of all the files, thereby keeping track of drastic changes in file access patterns. File migration requests are issued by the monitor and forwarded to the master node in a storage cluster. Upon receiving these requests, the file migration procedure in TIGER can be triggered in three ways.

- First, the master node initiates the file migration procedure periodically (e.g., hourly, daily, or weekly). The period length affects file migration performance, which in turn makes impact on thermal efficiency. Specifically, a large period is incapable of optimizing I/O performance in a timely manner; a small period introduces high file migration overhead. Choosing an appropriate file migration period can make good trade-offs between file placement optimization and file migration overhead.

- Second, the procedure starts migrating files when the percentage of files to be migrated exceeds a threshold, which is referred to as file migration threshold. Like the aforementioned file migration period, this threshold varies across different data centers. Both the threshold and the file migration overhead can be tuned by system administrators based on the characteristics of I/O load, access patterns, and the types of workload in data centers.

- Third, system administrators may manually launch the file migration procedure in accordance to the workload change rate, which is quantified as a percentage change in the current I/O load.

It is noteworthy that the file migration period, file migration threshold, and workload change rate are three vital factors affecting the thermal efficiency and performance of data centers. In one of our future studies, we will further investigate the impact of these three parameters on the system performance of storage clusters.

4.2 Design Goal

The overall goal of TIGER is to place a set F of m files to a group of N storage nodes in such a way to reduce cooling cost of a data center while maintaining high I/O throughput. The cooling cost is reduced by minimizing heat re-circulation; whereas the I/O performance is improved by assigning files with similar I/O service times to the same disk within a storage node.

The file access-pattern monitor (see, for example, [65]) offers TIGER with the service time s_i and access rate λ_i of file f_i in set F. The file assignment procedure in TIGER is comprised of two phases. In the first phase, thresholds for disk utilization are determined by the file-access-pattern monitor (see Section 4.3). In the second phase, files with similar service times are assigned to each disk until its utilization threshold is reached (see Section 4.4).

4.3 Disk Utilization

Now we discuss how to calculate disk utilization threshold to be used in the second phase of our approach. Recall that the utilization threshold is introduced to guide the file assignment.

4.3.1 Computing Disk Utilization Threshold

In the process of calculating the disk utilization threshold, we take into account both performance and thermal management. To improve I/O performance, we apply a load balancing strategy to uniformly distribute I/O load among all the disks. When it comes to thermal management, we follow the principle that workload placed on the node should be inversely proportional to the contribution of the node in the heat re-circulation in a data center. To place workload uniformly according to this principle, one has to ensure that all the nodes should contribute equally in heat re-circulation. Achieving this goal may be difficult;

therefore, it is normally useful to have a calibration phase, where we adjust the calculated threshold according to each node's contribution in the heat re-circulation.

As mentioned above, we first calculate the threshold using the load balancing strategy. The utilization of the disk d_i is increased by u_k due to the allocation of file f_k. The utilization u_k is a product of service time s_k and access rate λ_k of the file. Therefore:

$$u_k = s_k * \lambda_k \tag{4.1}$$

Our file assignment algorithm aims to distribute the total utilization U generated by all the files to D disks. We use the greedy algorithm to uniformly balance load among all the available disks. Disk utilization threshold U_{avg}^{Th} can be calculated using the following expression:

$$U_{avg}^{Th} = \frac{1}{D} \sum_{k=1}^{m} s_k \times \lambda_k \tag{4.2}$$

4.3.2 Considering Heat Re-circulation

Equation (4.2) can be used to uniformly distribute the workload in the data center. Although it is desirable to uniformly distribute the workload among the available nodes to gain performance benefits, our goal is to imbalance the workload to control the heat re-circulation in the data center. We use the principle that the workload to be assigned on a node should be inversely proportional to the contribution of the node in the heat re-circulation in the data center. If we apply this principle, uniformly distributing the workload means each node controbutes the same amount towards the heat re-circulation in the data center.

The heat re-circulation is characterized as the cross-interference coefficient, which is the fraction of total outlet heat of the node i that contributes to the inlet temperature of node j (see section 3.1). Then, the total contribution of a node in the heat re-circulation of a data center can be obtained as:

$$S_i = \sum_{j=1}^{n} \alpha_{ij} \tag{4.3}$$

$$S_i = \frac{S_i}{S_{total}} \tag{4.4}$$

where $S_{total}$is the sum of all the cross-interference coefficients of all the nodes. $S_i > S_j$ means node i contributes more in heat re-circulation than node j.

Therefore, uniformly distributing workload makes all the nodes identical in terms of heat re-circulation. Thus:

$$S_i = S_{avg}, \quad \forall i \in N \tag{4.5}$$

where N is the set of all nodes and $S_{avg} = \frac{1}{N} \sum_{i=1}^{N} S_i$.

Equation (4.5) does not hold for most of the nodes in the data center. In real-world scenarios, a node's contribution to heat re-circulation might be either higher or lower than the average contribution S_{avg}. To handle such situation we first calculate the normalized difference between S_i and S_{avg}.

$$\Delta S = \frac{S_i - S_{avg}}{S_{avg}} \tag{4.6}$$

There are two possible cases, which are discussed next.

Case 1: $S_i > S_{avg}$**.** This case holds for most nodes that are nearer to the floor, which contribute more towards the heat re-circulation. We decrease the disk utilization threshold by the normalized difference ΔS.

$$U_1^{Th} = U_{avg}^{Th} - \left(|\Delta S| \times U_{avg}^{Th}\right) \tag{4.7}$$

Case 2: $S_i < S_{avg}$**.**This case holds for most of the nodes nearer to the ceiling. As these nodes contribute less to the total heat re-circulation, we increase the disk utilization threshold by the normalized difference ΔS, at the same time, making sure that the disk utilization threshold does not exceed maximum utilization (i.e. 100%).

$$U_{high}^{Th} = U_{avg}^{Th} + \left(|\Delta S| \times U_{avg}^{Th}\right) \tag{4.8}$$

$$U_2^{Th} = \begin{cases} U_{high}^{Th} & \text{if } U_{high}^{threshold} < 1.0 \\ \\ 1.0 & \text{if } U_{high}^{threshold} > 1.0 \end{cases} \tag{4.9}$$

The disk utilization can be determined by the following expression derived from (4.7) - (4.9). Thus, we have:

$$U_i^{Th} = \begin{cases} U_1^{Th} & \text{if } S_i > S_{avg} \\ \\ U_2^{Th} & \text{if } S_i < S_{avg} \\ \\ U_{avg}^{Th} & \text{if } S_i = S_{avg} \end{cases} \tag{4.10}$$

4.4 The File Assignment Algorithm

TIGER solves the thermal management problem in data centers by applying thermal-aware file assignment to storage clusters. TIGER imbalances I/O workload in order to achieve significant thermal benefits. Such imbalanced workload conditions tend to degrade system performance due to a few highly utilized disks. TIGER takes two measures to mitigate the adverse impact caused by imbalanced workload. First, TIGER ensures that the utilization of any disk does not exceed a given threshold specified based on I/O requirements. Second, TIGER assigns files sharing similar service times to the same disks, thereby minimizing the variance in service times among requests in each disk [66]. To achieve this goal, TIGER sorts files in a decreasing order of their service times.

During the file assignment procedure, storage nodes are sorted in an increasing order of their heat re-circulation and the list of files is sorted in a decreasing order of their service times. For each node in the list, files are assigned to each disk on the node until the threshold

Algorithm 1 TIGER(file_info, node_info)

1: $U \leftarrow 0$
2: **for** $f_i \in m$ **do**
3: $\quad U \leftarrow U + s_i \times \lambda_i$
4: **end for**
5: $U_{avg}^{Th} \leftarrow \frac{1}{D}U$
6: $S_{total} \leftarrow 0$
7: **for** node $i = 1 \rightarrow N$ **do**
8: $\quad S_i \leftarrow \sum_{j=0}^{n} \alpha_{ij}$
9: $\quad S_{total} \leftarrow S_{total} + S_i$
10: **end for**
11: $S_{avg} \leftarrow \frac{1}{N}$
12: sort the nodes according to S_i
13: sort files according to service time
14: $k \leftarrow 0$
15: **for all** node $i \in$ sorted list **do**
16: $\quad S_i \leftarrow \frac{S_i}{S_{total}}$
17: $\quad$ **if** $S_i > S_{avg}$ **then**
18: $\quad\quad$ Calculate threshold using (**??**)
19: $\quad$ **end if**
20: $\quad$ **if** $S_i < S_{avg}$ **then**
21: $\quad\quad$ Calculate threshold using (**??**)
22: $\quad$ **else**
23: $\quad\quad U_i^{Th} \leftarrow U_{avg}^{Th}$
24: $\quad$ **end if**
25: $\quad$ **for all** disk $j \in D_i$ **do**
26: $\quad\quad$ **while** $U_j < U_i^{Th}$ **do**
27: $\quad\quad\quad$ assign file f_k to disk j
28: $\quad\quad\quad U_j \leftarrow U_j + (\lambda_k \times s_k)$
29: $\quad\quad\quad k \leftarrow k + 1$
30: $\quad\quad$ **end while**
31: $\quad$ **end for**
32: **end for**
33: **if** $k < m$ **then**
34: $\quad$ {still some files are remaining}
35: $\quad$ Start from the first node of the sorted list,
36: $\quad$ keep assigning files to the disk in the node until the utilization of the disks reaches 0.9
37: $\quad$ Repeat line 35 for consequent nodes in the sorted list until k=m.
38: **end if**

has been reached. We keep doing this until either the node list is empty or there are no files remaining. If the node list is empty and there are some files remaining, then we will start from the first node in the node list and keep assigning files until either utilization reaches 90% or all files have been assigned.

Prior to making any file placement decision, TIGER calculates the average disk utilization threshold U_{avg}^{Th} (see lines 2-6), thereby using greed method to uniformly distribute I/O load among available disks. After the initial assignment is complete, TIGER computes three important factors (i.e., S_{avg}, S_i, and S_{total}), which are used to calibrate the disk utilization threshold of each node(see lines 6-11). Next, TIGER sorts the list of nodes in an ascending order of their heat re-circulation contribution S_i, which is determined using equation (4.4) (see line 12). TIGER then picks the first node from the sorted list, and adjusts the disk utilization threshold for all the disks in the selected node depending upon the values of S and S_{avg} (see lines 16-22). Finally, TIGER assigns files to each disk in the selected node until either the threshold is reached or the disk's free capacity becomes empty (lines 24-28). TIGER repeatedly performs steps 15-31 until all the files are placed to the disks.

Please note that after assigning files to all the available disks according to the tuned utilization thresholds, some files may remain unassigned. TIGER checks whether there exists any unassigned files (see line 32) and starts assigning these files to the nodes using Steps 34-36. TIGER stops placing files to a disk if its utilization reaches 90% or there is no free capacity.

The time complexity of making file placement decisions is $O(m)$, where m is the number of files to be placed. We conduct an experiment to measure the execution time of making file placement decisions. Our finding reveals that it takes 15 ms to make the decisions of placing 23,000 files to a 50-node storage cluster. Both time-complexity analysis and run-time analysis confirm that the overhead of making file placement decisions is negligible.

Chapter 5

Thermal-Aware Scheduling in Hadoop Cluster

In the past decade, most of the large companies like Facebook, Google, Amazon etc are shifting MapReduce as a main framework in their data centers. MapReduce framework enables automatic parallelization and distribution of large-scale computation on large clusters of commodity machines [9]. Apache Hadoop is one the most popular MapReduce framework, where each file is partitioned into several equal size chunks, and for each chunk, multiple replicas (typically 3) are stored to improve the performance and data reliability. The primary job on most of the applications running on the Hadoop clusters is to deliver data or to run simple computations on large data sets, which at the lowest level, can be viewed as associating each data request to a specific replica among multiple available replicas.

As most of the businesses are shifting to Hadoop as their main framework for their data centers, it is very important to address the thermal emergencies in such clusters. We developed as scheduling policy to reduce the cooling cost the data center housing Hadoop clusters. We used the model proposed in Section 3.3 to calculate the contribution of each node in the heat re-circulation in the data center. Then, we assign the load on the nodes based on its contribution and the current load on the entire system. We implement our approach in Apache Hadoop 1.0.3 by modifying *FIFO Scheduler*, which is already available in Hadoop 1.0.3.

The remainder of the chapter is organized as follows: Section 5.1 explains the revised power model we have used for this particular study. In section 5.2, we presented an overview of existing scheduling policies and their commonalities in Apache Hadoop 1.0.3. Next, we presented out thermal-aware job scheduling approach in detail. Finally, we provided implementation details to incorporate our approach into Apache Hadoop 1.0.3.

5.1 Revised Power Model

Unlike storage clusters, where each node consists of large number of disks and disk sub-system is the major consumer of node power, in the case of Hadoop clusters, each node consists of multiple blade servers installed in a chassis. Though each blade server has multiple disks, usually their number is small (typically 2 to 4) and each server also has powerful processor which draws large amount of power. Therefore, in case of Hadoop clusters. processors are the major consumer of power. Hence, we cannot use the same power model presented in section 3.3. In this section, we present our revised power model which accounts for blade servers in the Hadoop clusters.

Apart from blade servers, chassis also draws constant amount of power to run its power unit. This constant power consumption can be denoted as P_{Node}^{base} and has two parts (1) constant amount of power consumed by chassis and (2) constant amount of power consumed by each blade server in the chassis. Therefore, P_{Node}^{base} can be calculated as sum of base power consumption of chassis and sum of base power consumption of all the servers in the chassis:

$$P_{Node}^{base} = P_{chassis}^{base} + \sum_{j=1}^{n} P_j^{base} \tag{5.1}$$

where $P_{chassis}^{base}$ is the base power consumption by the chassis, P_j^{base} is the base power consumption by blade server j in the chassis, and n is the number of servers in the chassis.

As mentioned above, processors are the major contributor to the power consumption in blade servers. Most of the blade servers are multicore, multiprocessor systems with 2-4 disks. As a result, disks consume very little power as compared to processors and can be neglected. Now, due to the linearity of power consumption of processor with the processor utilization [1], the power consumed by server j with CPU utilization u_j can be calculated as:

$$P_j^{CPU}(u_j) = \left(P_j^{CPU}(100) - P_j^{CPU}(0)\right) \times u_j \tag{5.2}$$

where $P_j^{CPU}(100)$ and $P_j^{CPU}(0)$ are the power consumption by the processor at utilizations of 100% and 0% respectively.

Therefore, the power consumption by node i can be calculated as sum of constant base power consumption (see equation (5.1)) and sum of variable power consumed by all the blade servers (see equation (5.2)).

$$P_i^{Node} = P_{Node}^{base} + \sum_{j=1}^{n} P_j^{CPU}(u_j) \tag{5.3}$$

It is important to note that to characterize the heat re-circulation in the data center and to calculate the cooling power required by the data center, we are using the same models described in section 3.1 and section 3.2 respectively. Equation (5.3) is used to calculate the power consumption of each node and replaces the variable $\overrightarrow{P}$ in equation (3.5).

5.2 Apache Hadoop: A MapRedue Framework

In order to understand our thermal-aware scheduling algorithm, it is very important to introduce the underlying MapReduce and Hadoop framework. In this section, we will take a quick overview of both MapReduce programming model and Hadoop Framework (see section 5.2.1). Then, we briefly describe existing scheduling policies available in Hadoop (see section 5.2.2). Finally, we explain the impact of job scheduling on power consumption and thermal profile of data center (see Section 5.2.3).

5.2.1 The MapReduce Framework

MapReduce is one of the most popular programming models processing and generating large data sets [9]. A MapReduce program typically make use of simple operations to perform on a huge amount of data. A MapReduce program consists of a pair of map and reduce functions. A map function is invoked for each record in the input data sets, producing intermediate records in the form of $\langle key, value \rangle$ pairs. These intermediate records, in turn,

are sorted in the way that all the records with the same *"key"* are sent to the same reducer. A reduce function, then, applies processing logic to combine these records and generates final output in the form of $\langle key, value \rangle$ pairs.

Hadoop is an open source software, initially developed to run MapReduce application. Data in a Hadoop system is typically stored in the Hadoop Distributed File System or HDFS, which is responsible for storing and managing data. HDFS consists of multiple DataNodes coupled with a NameNode allocating data to DataNodes and maintaining all the meta-data. The runtime system in MR1 consists of two main processes, namely, a master process called as JobTracker(JT) and multiple slaves referred to as TaskTrackers. The recent version of Hadoop provides multiple NameNodes and ability to port range of frameworks apart from MapReduce. YARN achieves this by decoupling programming model from resource management infrastructure, and delegating many scheduling functions to per-application components. YARN breaks down the functionalities of into two modules- *ResourceManager* (or *RM* for short) and *ApplicationMaster* (or *AM* for short). Per-cluster RM tracks resource usage, enforces allocation invariant, and arbitrates contentions among tenants. Responsibilities of per-application AM include coordinating logical plan of a single job by requesting resources from RM, generating a physical plan, and coordinating the execution of that plan around faults [67]. YARN also runs NodeManager, which is similar to TaskTracker of MR1. NodeManager(NM) communicates with RM to update it about the available resources and executes applications assigned by AM using resources assigned to the AM by RM.

5.2.2 Hadoop Scheduling Policies

A handful of scheduling policies have been developed as a plugin for the Hadoop framework. Popular schedulers include *FIFO*, *FairScheduler*, and *CapacityScheduler*. In the FIFO approach, all resource requests are queued and served in a first in first out manner. Resource allocation requests are serviced based on their arrival times rather than their priorities. To overcome this drawback, FairScheduler allocates the resources to the AM in a way that each

AM obtains an equal share of available resources. The FairScheduler implementation creates a set of pools, to each of which the resource requests are assigned. FairScheduler monitors each pool to ensure that each application gets fair share of available resources. A third intriguing scheduling policy is the Capacity scheduler, which offers great control as well as the minimum capacity guarantee. This policy also provides an ability to share excess resources among available application managers.

All the aforementioned scheduling policies are imported to YARN. Instead of scheduling jobs, RM allocates resources to AMs, each of which then runs actual jobs using those resources. Therefore, the FIFO scheduler maintains a queue to add the resource requests from AMs and serves the requests in the first-in-first-out manner. Resource requests are made in terms of containers. Each container contains resources like memory and processor cores. While processing resource request from an AM, RM calculates how many containers can be assigned on a given NodeManager governed by one of the scheduling policy. The policies pick an AM to which the available containers are assigned. Once the resources are allocated to the AM, the resources are fully utilized at the AM's discretion.

5.2.3 Impact of Resource Allocation on Power Consumption

Recall that RM assigns containers to the AM, which decides how to utilize the allocated resources (see Section 5.2.2). AM may make use of partial or all of the allocated resources. The goal of the AMs are to complete jobs as early as possible. Therefore, it is safe to assume that AM aggressively utilizes all of the allocated resources. Most of the containers are consists of processor core and memory. Thus, container allocation can be converted to the actual utilization of the nodes in a data center. Previous studies have shown that processor utilization is linearly proportional to the power consumption of a node, which in turn results in heat dissipation [1].

TASH aims at minimizing the heat re-circulation in a data center by controlling the power utilization of the nodes in the data center. *TASH* achieves this by manipulating

utilization of the nodes in the data center. As we explained, utilization is related to the resource allocation by the RM. Hence, in order to minimize the heat re-circulation in the data center, *TASH* controls the power utilization of the nodes in the data center.

5.3 TASH : Thermal-Aware Scheduling in Hadoop

TASH addresses the problem of reducing data center's cooling cost by minimizing the heat re-circulation in data centers. In order to achieve this goal, *TASH* controlls power consumption of computing nodes by constraining allocated resources on the nodes. *TASH* judiciously determines the amount of resources to be allocated to each node by examining contribution of the node towards heat re-circulation of a data center. In this Section, we first lay out an overview of *TASH* (see Section 5.3.1), followed by a detailed description of the algorithm (see Section 5.3.2).

5.3.1 Overview

TASH is a thermal-aware scheduler, which can be seamlessly integrated with any existing scheduling mechanism in clusters (e.g., Hadoop and Spark). Specifically, *TASH* facilitates *ResourceManager* to make thermal-aware resource allocation decisions. Fig. 5.3.1 depicts three collaborative modules in *TASH*; we show how the modules interact with each other. Recall that (see section 5.2.2) *NodeManager* sends *HeartBeats* to update the node's resource availability to RM and *ApplicationMaster* makes resource requests in the form of containers. *TASH* calculates each node's heat re-circulation contribution to a data center. Based on the contribution coupled with the overall resource demands from the systems, *TASH* determines the amount of resources (i.e., number of containers) allocated on the node. Once the number of containers are been determined, ResourceManager uses selected scheduling policy (e.g. FIFO, FairScheduler etc.) to decide which ApplicationMaster should acquire the available resources.

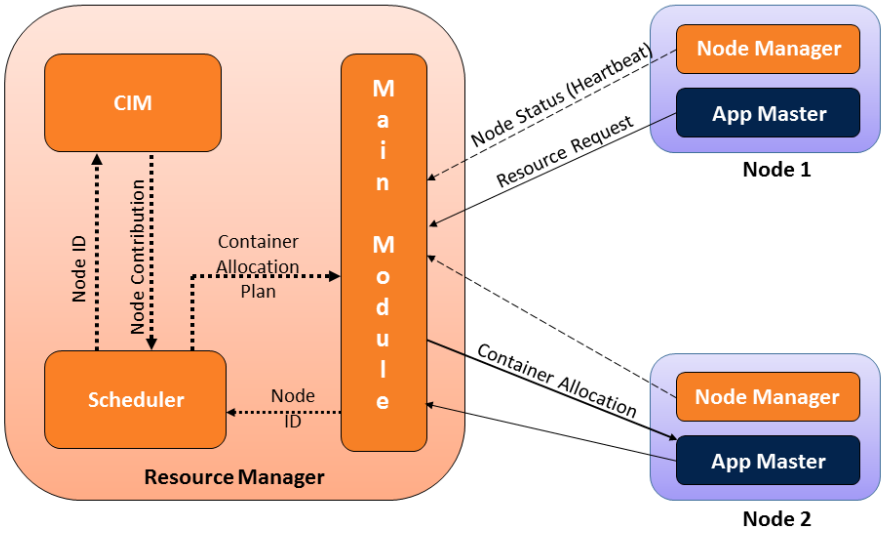


Figure 5.1: The system architecture of TASH, which integrates (1) a schedule, (2) the cross-interference matrix (CIM), and (3) the main module (a.k.a., resource manager).

It is important to note that *TASH* is independent of specific scheduling policy that selects ApplicationMasters to allocate available resources on a node; rather, *TASH* determines the amount of available resources should be allocated. Such a flexibility allows Hadoop cluster administrators to pair *TASH* with any existing policy like the Fair scheduler and the Capacity scheduler. A compelling feature of *TASH* is that it can be readily combined with existing schedulers to offer cooling-cost reduction, thereby allowing *TASH* to fully benefit from the perks of the existing scheduling algorithms.

Important parameters affecting resource allocation decisions in *TASH* include (1) contribution of each node in the heat re-circulation of a data center, (2) the maximum number of allowable containers on the node, (3) total resource demand in the system. Each node's contribution in the heat re-circulation is derived from the cross-interference matrix of a validated model. Recall that (see Section 3.1) the cross-interference matrix depicts the fraction of outlet heat from each node that contributes to the inlet temperature of all the other nodes.

The number of containers that can be allocated on the node is dependent of the node's hardware configuration like the number of cores and memory capacity as well as current resource allocation on the node. Given the available resources and the container's specification for an ApplicationMasters, ResourceManager determines the number of containers with given specifications that can be allocated on the node. *TASH* is aware of thermal factor while determining the number of containers. In particular, *TASH* incorporates the contribution of the node towards heat re-circulation in the data center.

5.3.2 The Thermal-Aware Scheduler

In this section, we shed light on the design of the *TASH* resource allocation policy. As mentioned earlier, *TASH* is driven by three decisive parameters, namely, contribution of a node in heat re-circulation, available resources on a node, and resource requests from the ApplicationMaster (see Section 5.3.1). The existing YARN scheduler takes into account the resource requests and available resources, ignoring thermal profiles of data centers during the process of resource allocation.

Recall that (see Section 3.1) cross-interference matrix $A_{N\times N}$ stores values $\alpha_{i,j}$ fraction of outlet heat from node i contributing to the inlet heat of node j. In order to calculate the contribution of node i towards the heat re-circulation of data center, we calculate a sum of all the values from i^{th} row normalized over sum of all the values from cross-interference matrix. We express this contribution as follows:

$$S_i = \frac{\sum_{j=1}^{N} \alpha_{i,j}}{S_{total}} \tag{5.4}$$

where N is the total number of nodes in a data center. S_{total}, the overall heat contribution, can be written as $S_{total} = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{i,j}$.

One way of calculating the number of containers to be allocated on a node is to uniformly distribute all the requested containers among all the nodes. For example, if the total

number of containers requested by all the available ApplicationMaster is 10 and there are five nodes in the cluster, then each node will be assigned two containers. In our design, we consider a container as a basic unit of resource allocation. We refer to this baseline policy as the uniform resource allocation. On the other hand, a fundamental principle that governs resource allocation decisions in *TASH* is that *resources allocated to a node is inversely proportional to the contribution of the node in the heat re-circulation in a data center*. In other words, *TASH* attempts to distributes available resources among all available nodes such that a node contributing highest heat re-circulation should consume the least amount of power.

In an ideal scenario where all the nodes equally contribute towards data center's heat re-circulation, we simply uniformly distribute containers among all the nodes. Let S_{avg} be an average heat re-circulation contribution. Applying normalized value of heat contribution, we express average contribution S_{avg} as:

$$S_{avg} = \frac{1}{N} \tag{5.5}$$

where N is the total number of nodes in a data center.

The YARN scheduler calculates the number of containers C_i to be allocated based on the container specification and available resources. Unlike YARN, *TASH* fosters the awareness of thermal efficiency by incorporating the node contribution of heat re-circulation into the management of container assignment. In order to incorporate thermal awareness into the resource allocation process, *TASH* compares two vital parameters, namely, (1) the ideal heat re-circulation contribution (see S_{avg} in Eq. 5.5) and (2) normalized contribution of node i in the heat re-circulation (see S_i in Eq. (5.4)). Such a heat-related comparison may yield three results. First, a node's contribute is larger than the average heat contribution (i.e., $S_i > S_{avg}$). Second, a node's heat contribution is smaller than that of the average one (i.e., $S_i < S_{avg}$). Finally, a node's heat contribution is on a par with the average contribution (i.e., $S_i = S_{avg}$), which is an ideal scenario. To integrate the heat comparison process into

ResourceManager's capacity calculation, *TASH* observes the difference (i.e., ΔS) between the average heat contribution (i.e.,S_{avg}) and the normalized contribution of node i (i.e., S_i). We express comparison difference ΔS as:

$$\Delta S = \frac{S_i - S_{avg}}{S_{avg}} \tag{5.6}$$

Once the heat-contribution comparison is completed (see (5.6), *TASH* calculates the number of containers to be allocated. We show the following three cases to illustrate how *TASH* dynamically configures the number of containers.

Case 1: $S_i > S_{avg}$. In the first case, node i's heat contribution is larger than the average contribution. According to our design principle, *TASH* reduces the number of containers to be assigned to node i. The reduced number of containers (i.e., C_i') is derived from the previous containers assignment (i.e., C_i) and the heat-contribution comparison result ΔS. We express updated containers C_i' as:

$$C_i' = C_i - (|\Delta S| \times C_i) \tag{5.7}$$

Case 2: $S_i < S_{avg}$. Node i's heat contribution is smaller than the average contribution, meaning that this node is eligible for processing an increased number of containers. Pushing up the number of containers on node i alleviates the over-heating problem illustrated in *case 1*. *TASH* modifies the container allocation using following equation. Thus, we have

$$C_i' = C_i + (|\Delta S| \times C_i) \tag{5.8}$$

Case 3 : $S_i = S_{avg}$. The last case resembles an ideal scenario, in which *TASH* keeps the container allocation unchanged. Thus, we conclude that there is no change in containers number C_i (i.e., $C_i' = C_i$).

It is worth mentioning that resource requests issued by an ApplicationMaster are serviced by an underlying scheduler, where **TASH** determines the number of containers to be allocated for that ApplicationMaster on a given node. By judiciously managing the number of allocated containers, *TASH* controls the nodel's in a thermal-friendly way.

Algorithm 2 TASH

1: $S_{total} \leftarrow 0$
2: **for** node $i = 1 \rightarrow N$ **do**
3: $S_i \leftarrow \sum_{j=0}^{n} \alpha_{ij}$
4: $S_{total} \leftarrow S_{total} + S_i$
5: **end for**
6: $S_{avg} \leftarrow \frac{1}{N}$
7: for given node i
8: use selected policy to select the application request
9: get the number of containers that can be allocated on the node C_i
10: $S_i \leftarrow \frac{S_i}{S_{total}}$
11: **if** $S_i > S_{avg}$ **then**
12: Calculate new number of containers using (5.7)
13: **end if**
14: **if** $S_i < S_{avg}$ **then**
15: Calculate new number of containers using (5.8)
16: **end if**
17: Allocate calculated number of containers for the application.

TASH computes the average and normalized heat re-circulation contribution of each node (see Lines 1-6). ResourceManager initiates the containing assignment process in response to the node's update event on receipt of a HeartBeat. After assessing available resources, ResourceManager invokes the scheduler to allocate containers to the ApplicationMaster by carrying out the following four steps. First, depending upon the employed scheduling policy, the scheduler picks the application request for a resource allocation. Next, the scheduler determines an initial number of containers to be allocated C_i. Then, *TASH* calibrates the number of containers based on the node's heat contribution (see lines 9-15). Finally, ResourceManager assigns the containers to the node.

5.4 Implementation Details

Recall that (see Section 5.2.1) there are two popular implementations of the Apache Hadoop computing platform. An earlier implementation of Apache Hadoop (a.k.a., MR1) is restricted in a sense that it is mainly focused on executing batch style mapreduce jobs. The newer version (i.e., MR2 or YARN) fully supports interactive jobs in addition to batch style mapreduce jobs. The two implementations are quite different from design's point of view and; therefore, we have to address these design issues differently for both the implementations. This section is dedicated to the design issues for the Yarn implementations (see section 7.2 for MR1 design issues).

YARN architecture (see also Section 5.2.2) decouples the programming model from the resource management infrastructure [67]. In this architecture, *ResourceManager* assigns resources to *ApplicationMasters* in terms of containers. Due to the different architecture, we face various design issues in this infrastructure. In this section, we shed light on the configuration of the thermal-aware resource manager. We also elaborate a way of implementing TASH by modifying the *FairScheduler* module.

5.4.1 Configuration

It is important to note that we provide *TASH* as a pluggin to the existing YARN implementation. We provide the number of parameters to facilitate configuration of *TASH* in Hadoop clusters. These parameters can be set in configuration file *yarn-site.xml*. Table 5.1 summarizes these parameters. In yarn-site.xml:

There are three parameters prescribed in the *yarn-site.xml* file. The first parameter (*yarn.thermal.awareness*) sets the mode of the ResourceManager. Before determining container assignments, *ResourceManager* first checks if the thermal-aware mode is enabled or not and; then container assignment decisions are made. When parameter *yarn.thermal.awareness* is set to 'true', *ResourceManager* will run in the thermal-aware mode. If the value of this parameter is 'false', the ResourceManager will run in the normal mode.

Parameter	Value	Note
yarn.thermal.awareness	if thermal aware mode is on or off	boolean value, default is false.
yarn.matrix.file.path	path to the CIM matrix file	If not specified, resourceman-ager will run in normal mode
yarn.cluster.node.count	number of nodes available in the CIM	optional

Table 5.1: A list of configuration parameters for YARN.

While running in the thermal-aware mode, *ResourceManager* assigns containers on nodes based on the nodes' contributions towards heat re-circulation of a data center. In order to calculate a node's contribution in heat re-circulation, the scheduler and the resource manager should have access to the cross-interference matrix. We implement a cross-interference matrix API, which loads the matrix from a file and calculates normalized contribution of each node in the data center. The second parameter (*yarn.matrix.file.path*) specifies the path under which the cross-interference matrix is stored. For the sake of simplicity, third parameter defines the number of rows available in the cross-interference matrix.

Recall that (see Section 5.2.1) the cross-interference matrix is dependent of the physical layout of a data center; such a matrix is unlikely to frequently and dramatically change. Therefore, it is straightforward for administrators to calculate the cross-interference matrix using either CFD software or the *XInt* methodology [29]. Our cross-interference matrix API first calculates the normalized contribution of each node in the data center hear re-circulation; then the API maintains a map of node's address and its normalized contribution. One of the API's perk is to offer an interface to query the contribution of a particular node by the node's IP address. The next subsection (Section 5.4.2) articulates the implementation details of *TASH*, which incorporates FairScheduler, our cross-interference matrix API, and the configuration parameters.

5.4.2 Thermal-Aware Yarn

Recall that (Section 5.2.2) *ResourceManager* dynamically allocates resources in terms of containers to *ApplicationMasters* to run on computing nodes [67]. Governed by the selected scheduling policy (e.g., FIFO, FairScheduler), ResourceManager handles resource requests from *ApplicationMasters*, thereby assigning available containers to service *ApplicationMasters'* resource requests. We implement *TASH* through the modification of FairScheduler. Hence, let us first introduce the implementation of FairScheduler.

NodeManager or *NM* periodically communicates with ResourceManager through heartbeats, which contains information pertaining to available resources. A scheduler also maintains its own map of all the nodes to keep track of all the containers launched on the nodes. While processing a heartbeat, the scheduler first frees completed containers on the node, followed by scheduling new containers on the node. If the continuous scheduling mode is enabled, FairScheduler first sorts all the nodes in a descending order of available resources. Next, the scheduler picks a node from the beginning of the sorted list; containers are allocated to ApplicationMaster on the node based on the available resources and resource-request specification. We provide an API to manipulate available resources on a node based on the node's contribution towards heat re-circulation. This API is invoked twice: (1) in the node-sorting procedure and (2) in the resource assignment module in which the scheduler determines whether a container should be allocated to the node or not.

It is noteworthy that container specifications and available resources are represented in terms of memory and CPU cores. When it comes to such physical resources, we cannot apply Eq. (5.8) to allocate the amount of resources that exceeds the physical capacity. Nevertheless, we are able to manage available resources on the nodes that are major heat re-circulation contributors. Therefore, we calculate ΔS for a nodes with high contributions and then deduct the resources from available resources to adjust the available resources (see Eq. (5.7). Importantly, TASH is orthogonal to the existing scheduling policies, which assign containers to ApplicationMaster. For example, FairScheduler, advised by the scheduling policy, selects

an ApplicationMaster followed by allocating containers to that ApplicationMaster based on its resource requests. TASH is in charge of deciding the number of containers be allocated on a given node.

Chapter 6

Experimental Results

We evaluated our proposed thermal-aware file and resource allocation policies through extensive experimentation. In this chapter, we present detailed analysis of results gathered during our experimentation. The remaining chapter is organized as follows: we first present the experimentation results of TIGER (see Section 6.1), our file assignment policy. In section 6.2, we present experimental results for our thermal-aware resource allocatio policy-TASH.

6.1 TIGER Evaluation

We evaluate the energy efficiency and performance of the TIGER algorithm. First, we explain the existing algorithms, which we used for the evaluation of the TIGER. We chose one load balancing algorithm and one thermal-aware file assignment algorithm for the comparison of performance of TIGER in terms of enegy efficiency and system response time. Next, we presented few preliminary results to evaluate the performance of TIGER.

6.1.1 Baseline Algorithms

As mentioned above, we chose two algorithms to compare against TIGER. First one is Greegy Load Balancing algorithm while the second one is CoolestInlet algorithm.

The Greedy Load-balancing Algorithm

The greedy load balancing algorithm uniformly distributes I/O load among all available disks in storage clusters. We assume that service time and access rate of file is known a priori. The greedy algorithm applies equation (4.2) to calculate the average threshold of

each disk. Next, the algorithm picks a node in which files are assigned to each disk until load threshold is reached. The algorithm follows this strategy to assign all the files, thereby uniformly distributing the workload among the storage nodes in a cluster.

CoolestInlet

The CoolestInlet algorithm distributes the workload according to inlet temperatures of the nodes. The algorithm places more workload on the nodes with lower inlet temperatures. We consider the range of inlet temperatures with upper and lower bound on the inlet temperature. In our study, we set the upper bound (T^{UB}) to 25^oC, which is a common redline temperature; we set the lower bound (T^{LB}) to 15^oC, which is an ideal temperature supplied by real world cooling systems. If a temperature of a node is equal to or greater than upper bound (i.e. 25^oC, in our case), then no load is assigned to the node. If the node's inlet temperature is equal to or less than lower bound (i.e. 15^oC, in our case), then the threshold for all the disks in a node is configured as 100%. For node i's inlet temperature (T_i^{in}), that is in the range between 15^oC and 25^oC, we calculate the node's threshold by using following equation:

$$U_i^{Th} = \frac{T^{UB} - T_i^{in}}{T^{UB} - T^{LB}} \tag{6.1}$$

The above threshold largely depends on the inlet temperature and the lower bound. Thus, a low inlet temperature and high lower bound give rise to high workload threshold. Similar to TIGER, the CoolestInlet scheme sorts files according to their service times. CoolestInlet then sorts the nodes according to their inlet temperatures. Next, starting from the first node, CoolestInlet calculates disk-utilization thresholds and places files to disks until their corresponding threshold is reached.

TIGER Vs. Workload Placement Algorithms

Storage clusters are focused on addressing I/O performance problems encountered in I/O-intensive workload conditions, under which storage nodes claim to offer high I/O throughput for data-intensive applications. The utilization of nodes in a data center is mainly driven by data placement (a.k.a. file allocation) and file access patterns. Modern cluster file systems (see, for example, [68] and [69]) leverage data replicas to boost I/O performance. Workload placement and file placement strategies are of equal importance in the context of thermal-aware replica management, because data placement algorithms are in charge of determining location of data replicas, among which workload placement algorithms judiciously pick the most appropriate replicas to make I/O accesses thermal friendly.

Workload and file placement algorithms are comparable in the sense that both types of algorithms are contributing factors of node utilization in a data center. Although, the Greedy Load Balancing and CoolestInlet algorithms are focusing on the workload placement issues, workload placement decisions made by these two algorithms ultimately affect the utilization of storage nodes. Similarly, TIGER manages node utilization through file placement. TIGER and the other algorithms are comparable, because all the three schemes can be applied to make trade-offs between cooling efficiency and I/O performance.

6.1.2 Experimental Setup and Workload Characteristics

Fig. 6.1 shows the layout of a data center, which contains two rows of five racks. Each rack is comprised of five chassis or nodes, each of which consists of six 1U RAID systems. Each RAID system has a RAID controller and four hot swappable disks. And each RAID draws 118 W power when no disks are active or attached.

$$P_a^{idle} = 118W \tag{6.2}$$

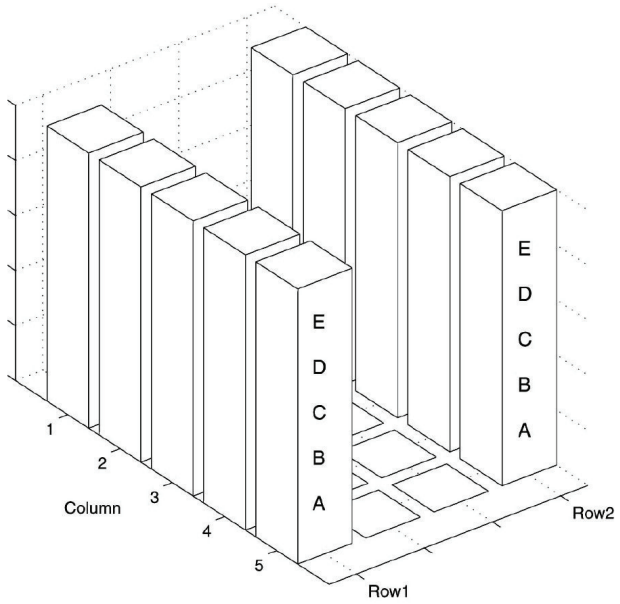


Figure 6.1: Data Center Layout [1]

We generate I/O workload that resemble online access patterns, which follow the Zipf like distribution [70][71][72][73]. For example, a recent study shows that the Youtube video popularity is closely correlated with the Zipf distribution [73]. As such, in our experiments files are accessed following the Zipf distribution. Evidence also shows that most popular files are small in size [66]. To reinforce the results of access patterns obtained from the previous studies, we inversely correlate the distribution of access rates and service time of the files used in our experiments.

6.1.3 Thermal Impact of Energy Efficient Disks

Scenario 1

Fig. 6.2 shows the experimental results for the best case scenario. In this case, we test an energy-saving algorithm that spins down disks to the sleep mode when the disks are idle. Under the control of this algorithm, the power consumed by a disk have three components - power consumed by the disk when it is in the active mode and the sleep mode, and the

power consumed by the disk when it is transitioned between the two modes. The total power $P_{i,j}^{d}$ consumed by disk j in node i is expressed as (6.3) below:

$$\begin{aligned} P_{i,j}^{d} &= \frac{1}{T}\Big(t_{j}^{active} \times P_{i,j}^{d,active} + t_{j}^{sleep} \times P_{i,j}^{d,sleep} \\ &\quad + \frac{N_{j}^{t}}{2}\left(P_{S_{down}} + P_{S_{up}}\right)\Big) \end{aligned} \tag{6.3}$$

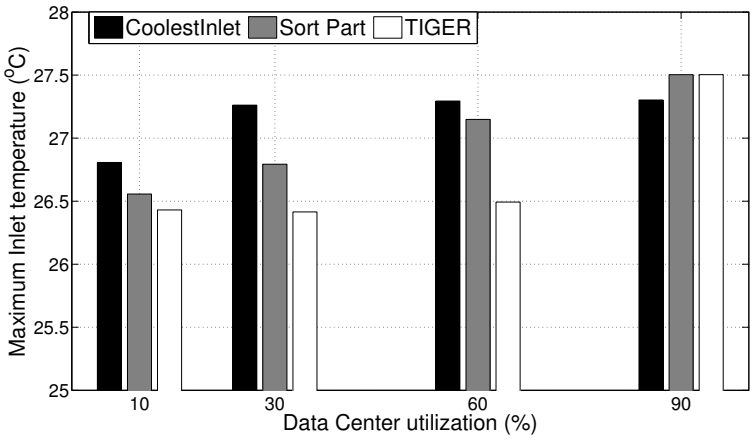
We observe from both Figs. 6.2(a) and 6.2(b) that TIGER conserves more cooling energy than CoolestInlet and SortPart - two existing solutions. Compared with these two algorithms, TIGER significantly reduces the cooling cost of the data center by more than 15% when the data center's utilization is in a range of 30% to 60%. The cooling cost discrepancy between TIGER and the two existing schemes diminishes when the utilization is either below 30% or above 70%.

Scenario 2

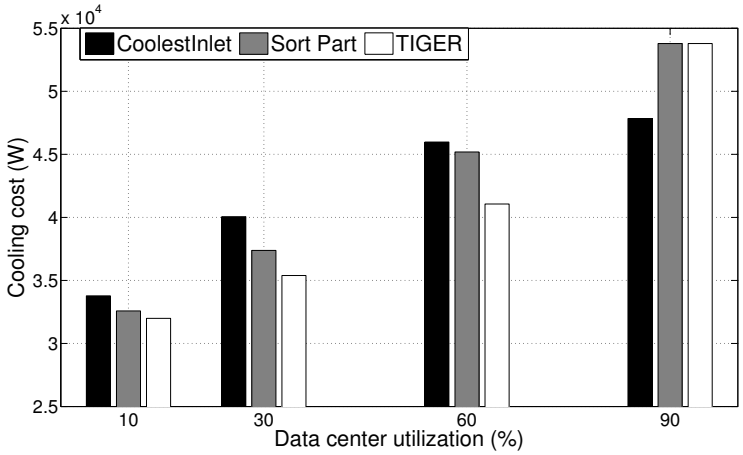
Fig. 6.3 shows the results for the second case where no energy-saving techniques are deployed to spin up and spin down disks. This experiment can be considered as the worst case scenario from the perspective of energy conservation. A disk is either in the active mode or the idle mode. There are no transitions from the active/idle mode to the sleep mode and vice versa. The disk's total power $P_{i,j}^{d}$ can be rewritten as (6.4).

$$P_{i,j}^{d} = \frac{1}{T}\left(t_{j}^{active} \times P_{i,j}^{d,active} + t_{j}^{idle} \times P_{i,j}^{d,idle}\right) \tag{6.4}$$

Fig. 6.3 reveals that TIGER slightly outperforms the other two algorithms when none of the three tested schemes incorporate the disk energy-saving technique. We observe that the power discrepancy between the active and the idle mode is almost negligible. Therefore, the workload distribution does not significantly manipulate the power distribution among the nodes in the data center. Furthermore, because idle disks consume an equally large amount



(a) Maximum T^{in}



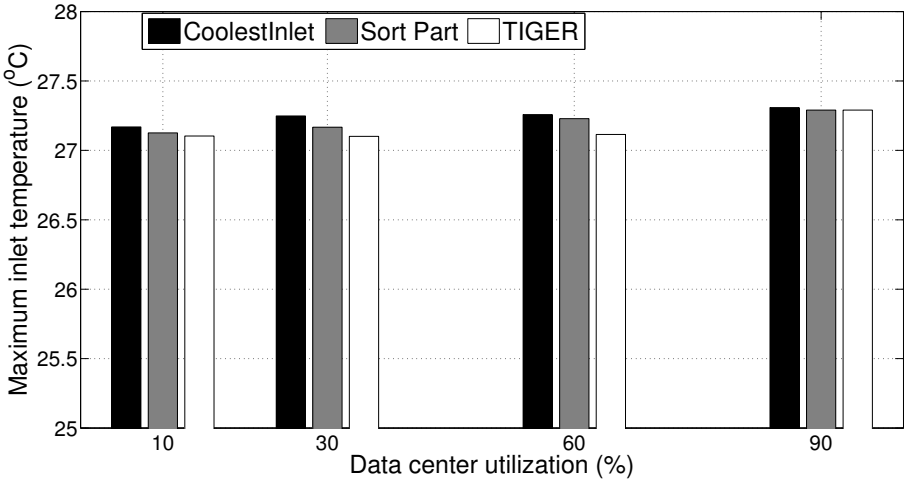
(b) Cooling Cost

Figure 6.2: Inlet temperatures and cooling cost of CoolestInlet, SortPart, and TIGER algorithms in scenario 1, where idle disks are transitioned into the sleep mode to conserve energy.

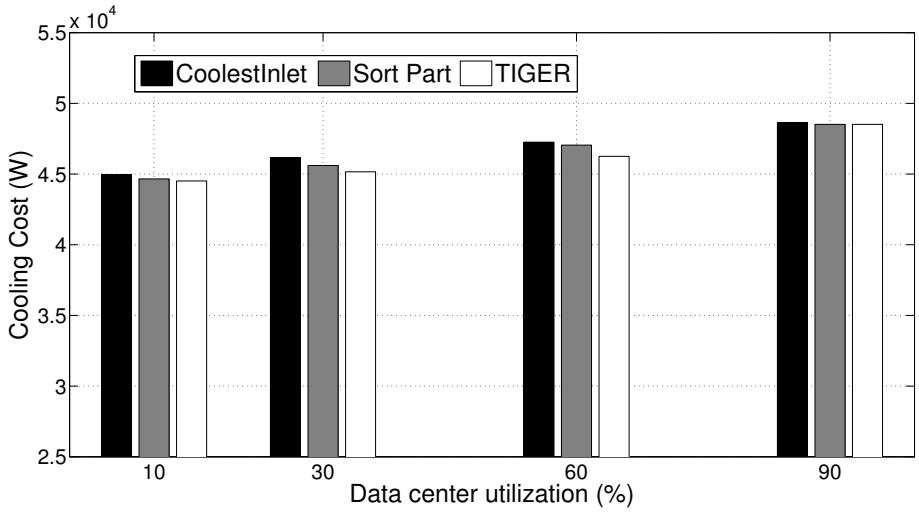
of energy as active disks, the overall energy consumption in the data center is very high. This power feature results in high cooling cost for scenario 2 for all the three solutions.

Impact of Sleep mode percentage

Section 6.1.3 and 6.1.3 delineate the extreme-case scenarios. Disks are put into sleep mode to conserve energy if disk-idle-time percentage is larger than a threshold. In this group of experiments, we investigate the impact of such an idle-time-percentage threshold (a.k.a., sleep mode percentage) on storage clusters. We keep the system utilization at the level of



(a) Maximum T^{in}

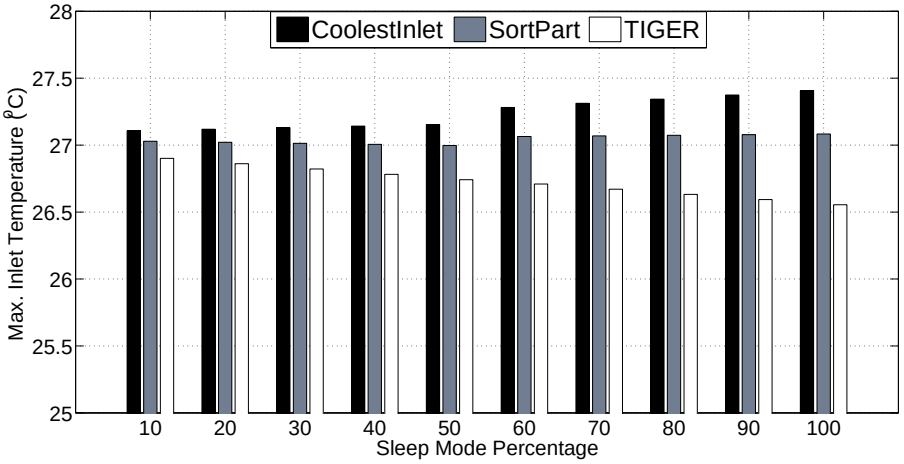


(b) Cooling Cost

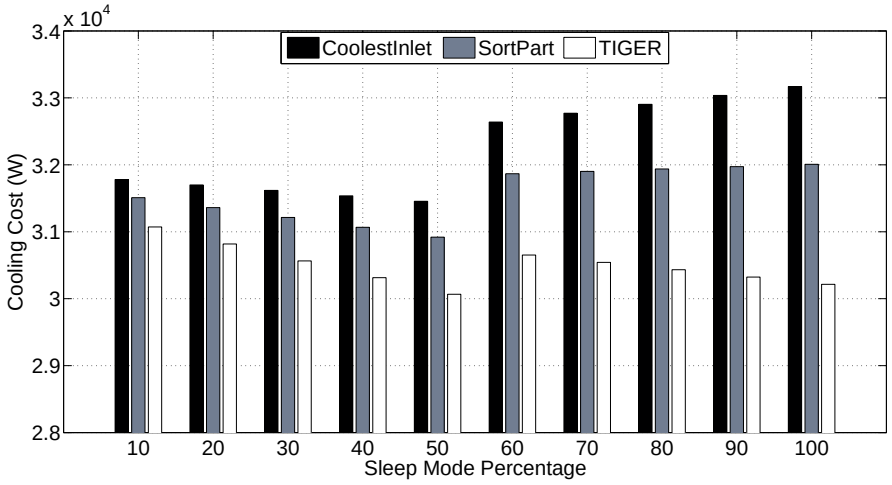
Figure 6.3: Inlet temperatures and cooling cost of CoolestInlet, SortPart, and TIGER algorithms in scenario 2, where idle disks are never transitioned into the sleep mode.

50% while varying the idle-time-percentage threshold referred to as sleep mode percentage (see Fig. 6.4).

Fig.6.4(a) shows the minimum inlet temperatures of the nodes in a data center governed by all three strategies. We observed from the experimental results that regardless of sleep mode percentage, TIGER outperforms the other two schemes. Also, as the energy saving policy becomes more aggressive (i.e. the sleep mode percentage is large), the performance gains offered by TIGER become more pronounced.



(a) Maximum T^{in}



(b) Cooling Cost

Figure 6.4: Impact of sleep mode percentage on inlet temperatures and cooling cost of CoolestInlet, SortPart, and TIGER. The system utilization is kept at 50%

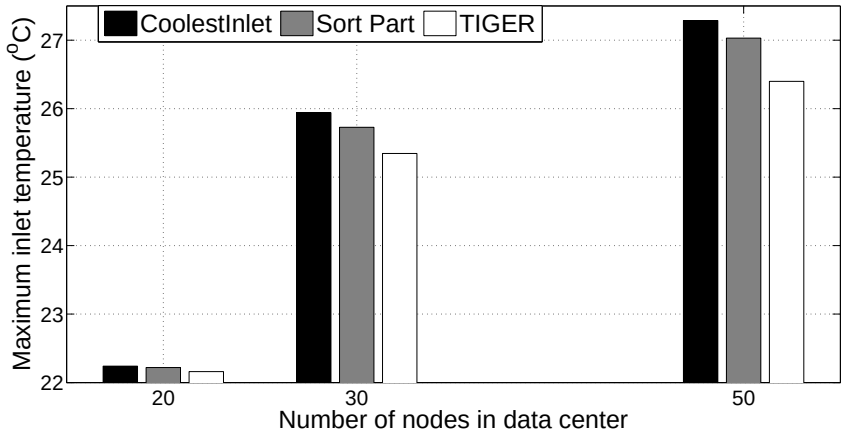
An intriguing observation drawn from Fig.6.4 is that when it comes to *CoolestInlet* and *SortPart*, the maximum inlet temperatures tend to go up with the increasing value of the sleep mode percentage. In contrast, with TIGER, the maximum inlet temperature drops when the sleep mode percentage increases. This performance trend is driven by the workload distribution of three strategies. More specifically, *CoolestInlet* attempts to assign an excessive amount of workloads to nodes exhibiting low temperatures. Therefore, some nodes in the data center are likely to be over utilized while others may be sitting idle. When

the number of disks' power transitions increases, energy consumption incurred by disk power transitions becomes a significant overhead, which offsets energy savings provided by placing idle disks into the sleep mode. On the other hand, TIGER slightly imbalances I/O load in a way to control the heat re-circulation in the data center. Consequently, even if the energy consumed by the nodes is high, TIGER reduces the maximum inlet temperature of the nodes in the data center.

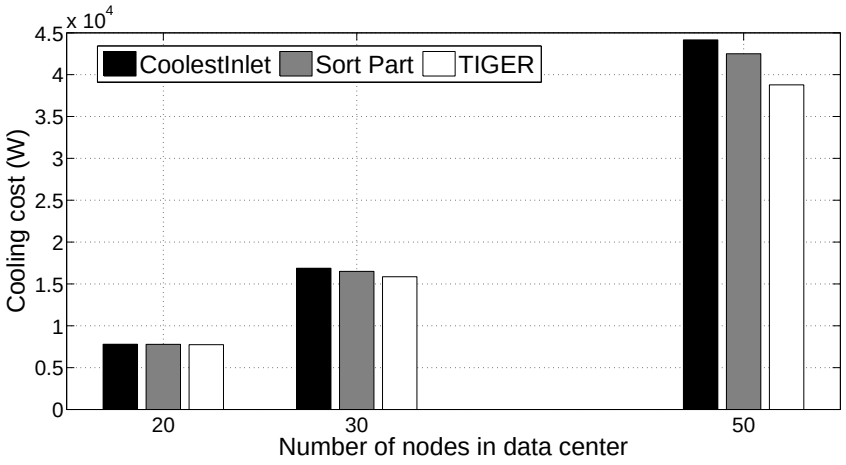
6.1.4 Scalability

Due to the tremendous rate of growth in the data center size, scalability is one of the most important features any software framework developed to support ever-growing data centers. In order to investigate the sensitivity of TIGER towards data center size, we ran experiments on cluster with three different sizes keeping the data center utilization constant at 50%. Fig. 6.5 shows the impact of the number of nodes on maximum inlet temperature and cooling cost of data center under the three schemes. Fig. 6.5 shows that compared with CoolestInlet and SortPart, TIGER reduces both inlet temperature and cooling cost of the data center. More importantly, the improvements offered by TIGER become more pronounced with the increasing number of nodes. This observation is attributed to the fact that heat re-circulation becomes more severe when the number of nodes scales up.

Fig. 6.5(a) shows that the inlet temperature discrepancy between the 20-node case and the 50-node case is almost 5 ℃ ; Fig. 6.5(b) reveals that the cooling cost of the 50-node case is more than four times larger than that of the 20-node case. The cooling cost is comprised of two components (see equation (3.7)): co-efficient of performance $COP(T_{sup})$ and power consumption P_C of computing resources. $COP(T_{sup})$ relies on the data center's supply temperature, which in turn depends upon maximum inlet temperature. When the data center scales up in terms of the number of nodes, the increased inlet temperatures cause a reduction in $COP(T_{sup})$. Furthermore, expanded computing resources increase the total



(a) Maximum T^{in}



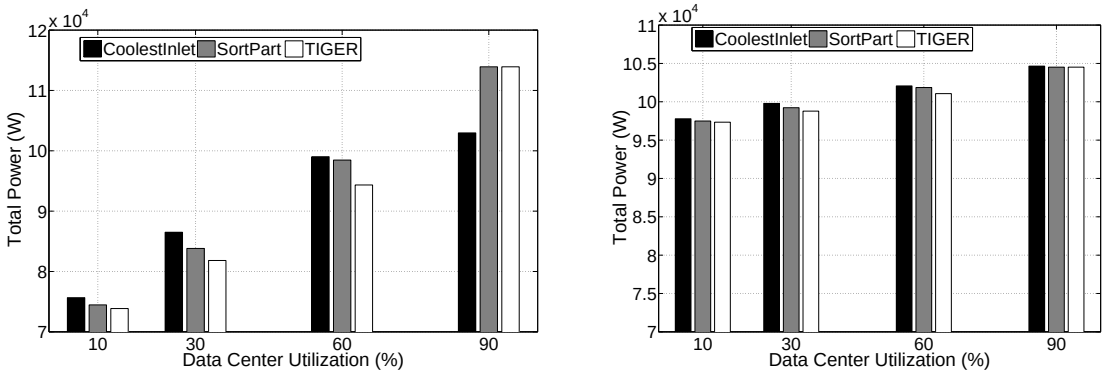
(b) Cooling Cost

Figure 6.5: Impact of the number of nodes on maximum inlet temperature and cooling cost.

power consumption of the data center's computing facility, which considerably increases the cooling cost.

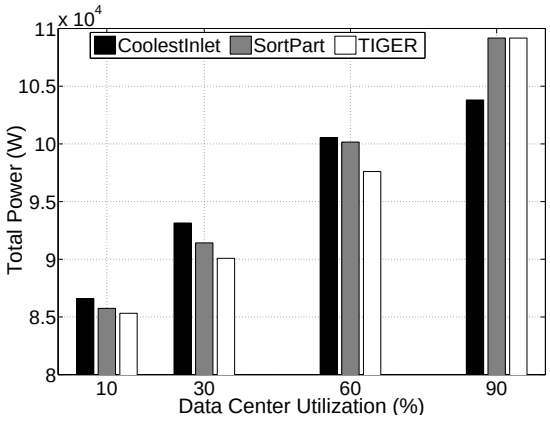
Figs.6.7- 6.8 plot maximum inlet temperatures and cooling cost of the data center when the initial supply temperature is set to 12 ℃ and 13.5 ℃ . In our measurement method, we configure the supply temperature followed by distributing I/O load among the nodes in the data center; then we measure inlet temperatures of the nodes affected by heat re-circulation and the supply temperature. According to the newly collected inlet temperatures, we set the new supply temperature and calculate the cooling cost derived from the supply temperature.

We repeatedly perform the above procedure to measure inlet temperatures, which determines the changes in the supply temperature.



(a) Scenario 1

(b) Scenario 2



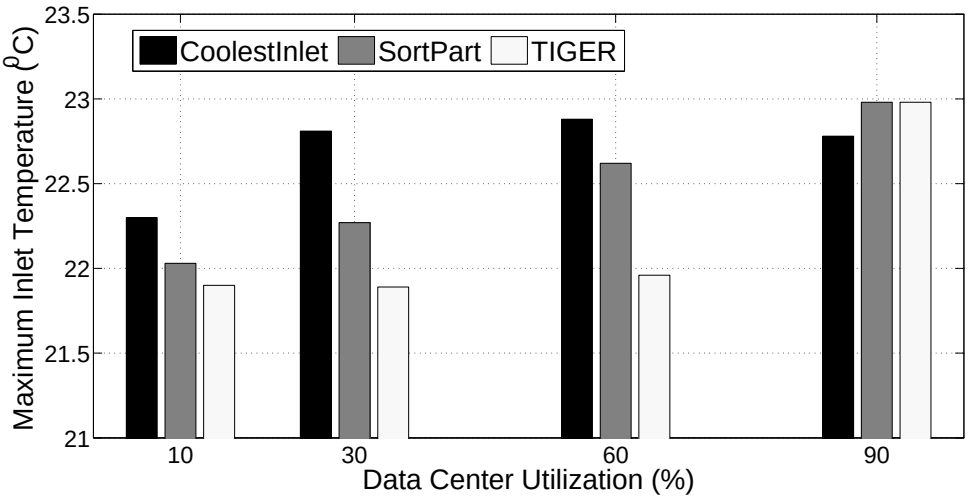
(c) Scenario 3

Figure 6.6: Total power consumption (i.e., cooling and operational cost) of the data center managed by TIGER, CoolestInlet, and SortPart in a) Scenario 1 (see Section 6.1.3), b) Scenario 2 (see Section 6.1.3), and c)sleep mode percentage 50%

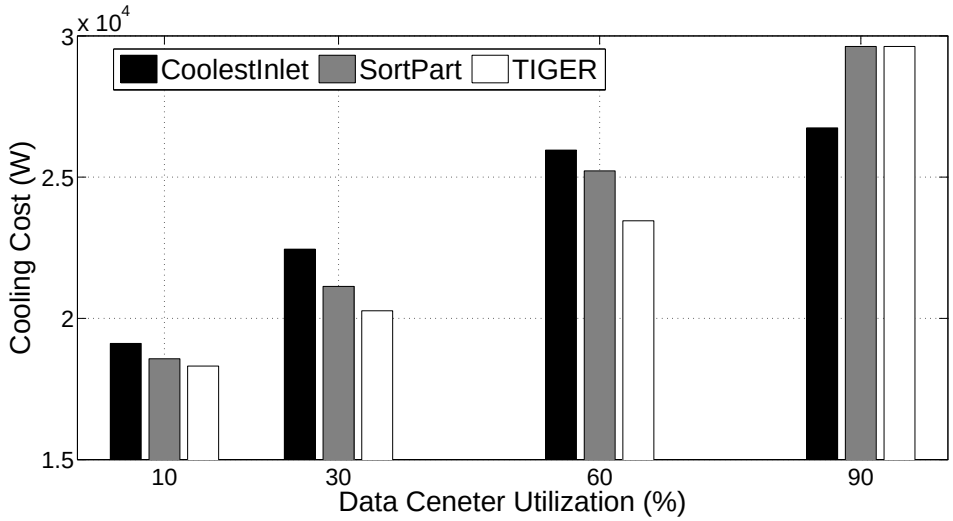
6.1.5 Impact of Initial Supply Temperature

Figs.6.7- 6.8 plot maximum inlet temperatures and cooling cost of the data center when the initial supply temperature is set to 12 ℃ and 13.5 ℃ . In our measurement method, we configure the supply temperature followed by distributing I/O load among the nodes in the data center; then we measure inlet temperatures of the nodes affected by heat re-circulation and the supply temperature. According to the newly collected inlet temperatures, we set the new supply temperature and calculate the cooling cost derived from the supply temperature.

We repeatedly perform the above procedure to measure inlet temperatures, which determines the changes in the supply temperature.



(a) Maximum T^{in}

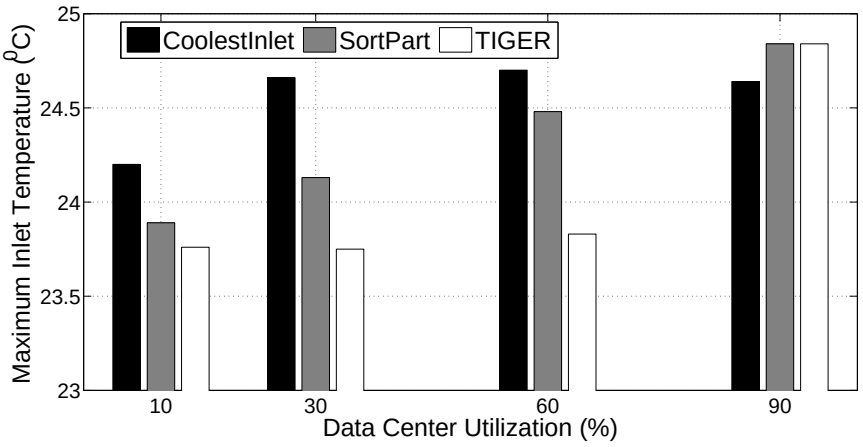


(b) Cooling Cost

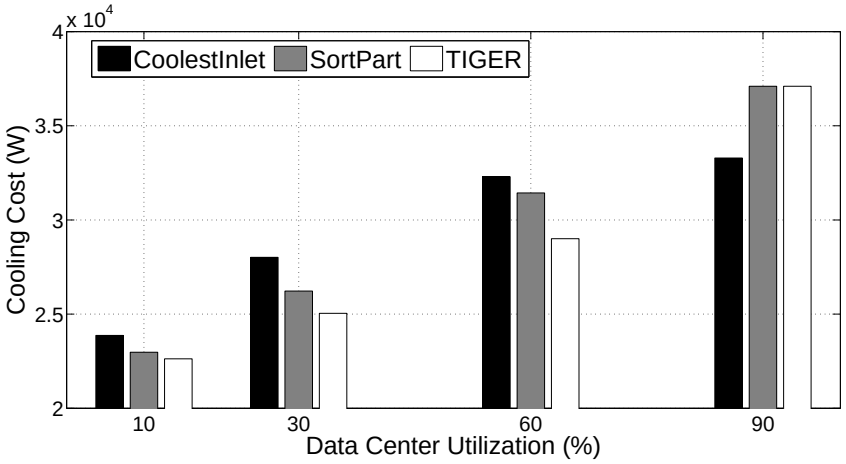
Figure 6.7: Impact of initial supply temperature on maximum inlet temperature and cooling cost. Initial $T_{sup} = 12^0C$

Given an initial supply temperature, in the first phase cooling cost for all the three schemes partially depends upon the energy-conservation technique employed to spin down idle disks. By the end of the first phase, we measure the inlet temperatures, from which we calculate the new supply temperature. The supply temperatures of the data center governed by the three schemes are different from each other; therefore, the cooling cost of the data

center controlled by the three strategies vary substantially (see Figs. 6.2(b), 6.3(b), and 6.4(b)).



(a) Maximum T^{in}



(b) Cooling Cost

Figure 6.8: Impact of initial supply temperature on maximum inlet temperature and cooling cost. Initial $T_{sup} = 13.5^0C$

Comparing Figs. 6.7(a) and 6.8(a), we observe that the maximum inlet temperature in the case where the initial supply temperature is 12^0C is much lower than that in another case where the initial supply temperature is set to 13.5^0C. This trend is not surprising because of the nature of the initial supply temperature. Due to a lower maximum inlet temperature, at the beginning of the next phase there is ample room to increase the supply temperature

to gain cooling cost benefits. Therefore, the cooling cost shown in Fig. 6.8(b) is much lower than that plotted in Fig. 6.7(b).

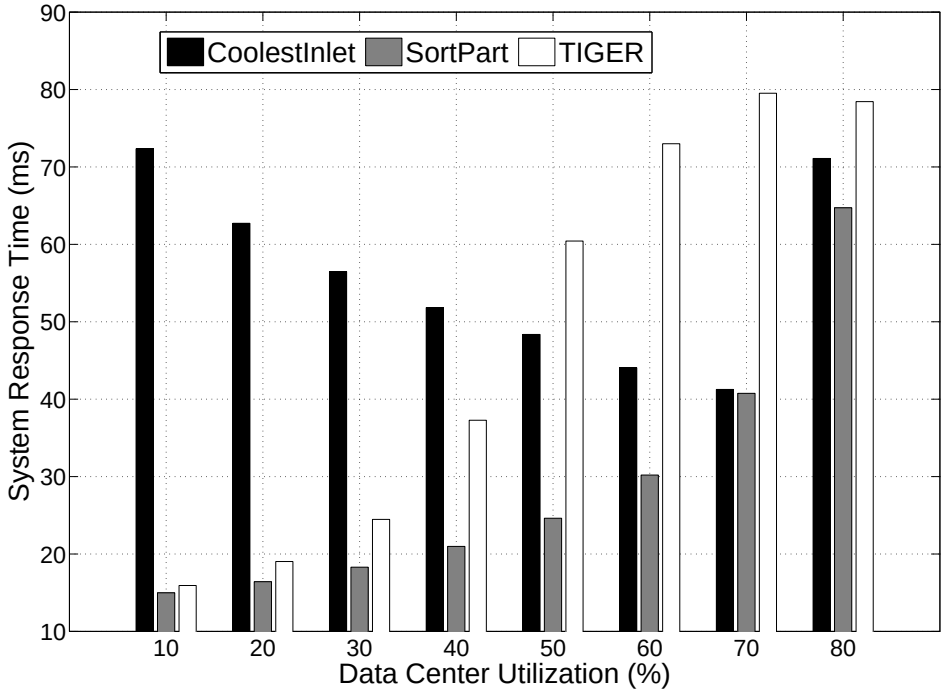


Figure 6.9: Average response time of the data center managed by TIGER, CoolestInlet, and SortPart.

The contribution of storage node in terms of heat re-circulation in a data center largely depends on the node's physical location coupled with the other parameters (e.g., power consumption). The node physical locations in a data center are unlikely to change in a frequent manner; therefore, given a data center layout, the contributions toward the data center's heat re-circulation remain unchanged. Thanks to the constant heat re-circulation factors, there is no need to calculate the heat re-circulation contributions each time prior to making file placement decisions. The heat re-circulation factors can be calculated during a calibration phase whenever there is any change in a data center's layout. For example, the calibration phase is invoked when (1) new racks or nodes are added to the data center or; (2) existing racks or nodes are removed from the data center. In a data center managing an

enormous number of files hosted on hundreds of nodes, such a one-time cost of calculating the heat re-circulation contributions is considered negligible.

6.2 TASH Evaluation

We conducted a series of experiments to evaluate the performance of *TASH* in terms of cooling cost savings and system's runtime. In this section, we describe these experiments in detail. First, we provide detailed description of our test bed and initial condition of our Hadoop cluster(see section 6.2.1). Section 6.2.2 provides the summary of all the experiments and conclusions drawn from the results. Finally, section 6.2.3 performs detailed analysis of our experimental results.

6.2.1 Experimental Setup

Our test bed consists of a Hadoop cluster consisting 18 nodes, where 16 nodes are data nodes with separate two nodes running ResourceManager and NameNode.

Hardware	
Computer	Dell OptiPlex 3020
Processor	Intel Core i5-4590 Processor Quad 3.3 GHz
Memory	8GB DDR3 SDRAM at 1600 MHz
Network	1 Gigabit Ethernet
Disks	500 GB SATA
Software	
Operating System	Ubuntu 14.0
Hadoop dist.	Apache Hadoop 2.7.3

Table 6.1: Cluster Specifications.

These nodes are arranged in 2 rows, where each row contains two racks. Each rack contains four nodes. As shown in the table, we used Apache Hadoop 2.7.3 to compare against our resource allocation policy. In the calibration phase, we calculated cross-interference coefficient for the cluster using the methodologies described in [29]. Fig. 6.2.1 depicts cross-interference matrix for our cluster.

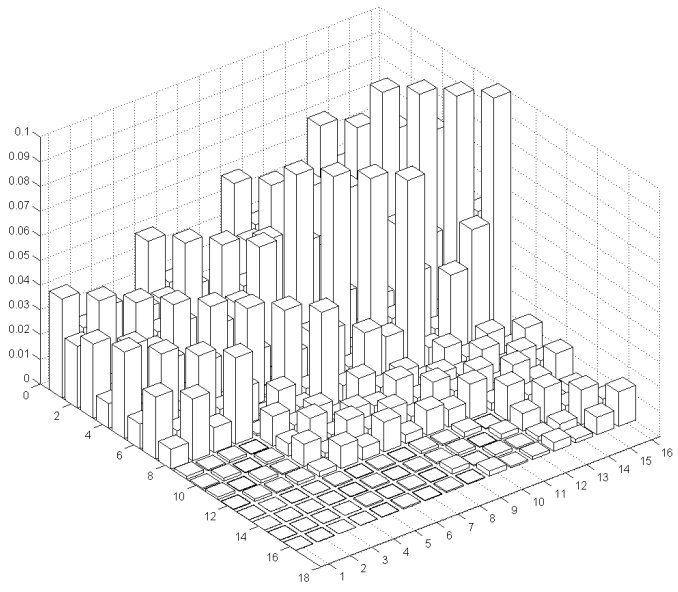


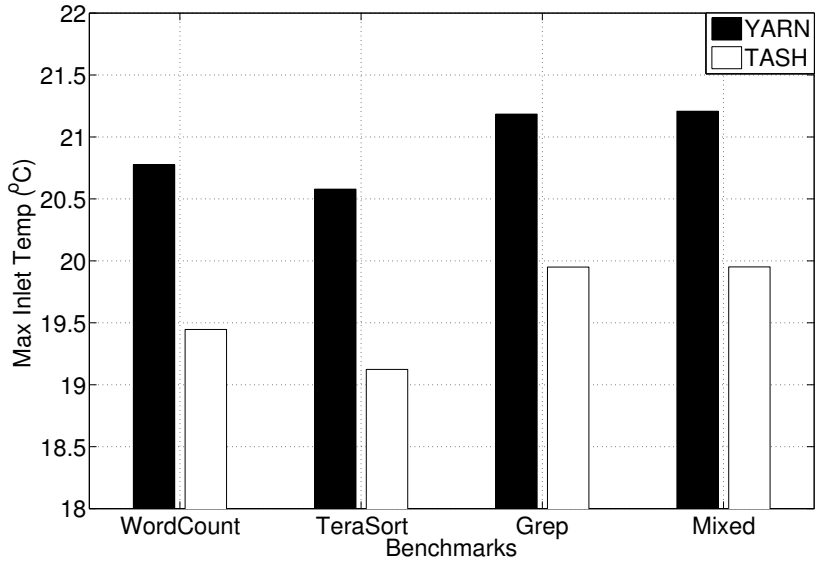
Figure 6.10: Cross Interference Coefficient of Data Center

6.2.2 TASH performance on MapReduce Benchmarks

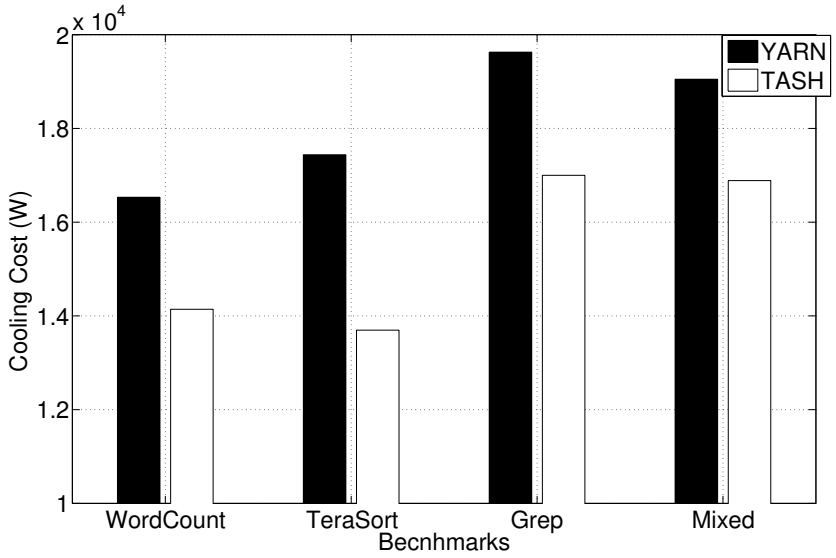
In this section, we evaluate the performance of *TASH* against that of existing YARN implementation. We run multiple map reduce benchmarks on both frameworks. It is important to note that we just need to set the configuration parameter to activate *TASH*. Since we are not making any changes to HDFS, we can run benchmarks using same data sets. We log the processor utilization of each node during the application run. Then, by using equations described in section 3, we calculate the maximum inlet temperature and cooling cost of the cluster.

Figure 6.11 depicts the results for three mapreduce benchmarks. We run WordCount, TeraSort, and Grep- three most popular map reduce benchmarks. Each of the benchmark was run on the input data set of size 100 GB. It is evident from figure 6.11 that *TASH* outperforms YARN in all the cases. WordCount represents the computation intensive workloads, where as TeraSort represents I/O intensive workloads. Though *TASH* performs better than YARN in terms of cooling cost conservation, it is important to note that the performance benefits achieved in computation intensive workloads is better than that of I/O intensive workloads. Observations from Fig. 6.11 can be explained by taking a closer look at the equation 5.2.

Equation 5.2 shows that the power consumed by the node is mostly dependent upon the utilization of the processor.



(a) Maximum T^{in}



(b) Cooling Cost

Figure 6.11: Inlet temperatures and cooling cost of YARN and TASH in for different benchmarks. Input size 100GB for each benchmark.

In case of computation intensive workloads like WordCount, processor is extensively utilized. Therefore, we can see the higher power consumption by the nodes, which results in higher maximum inlet temperature. On the other hand, I/O intensive workloads like

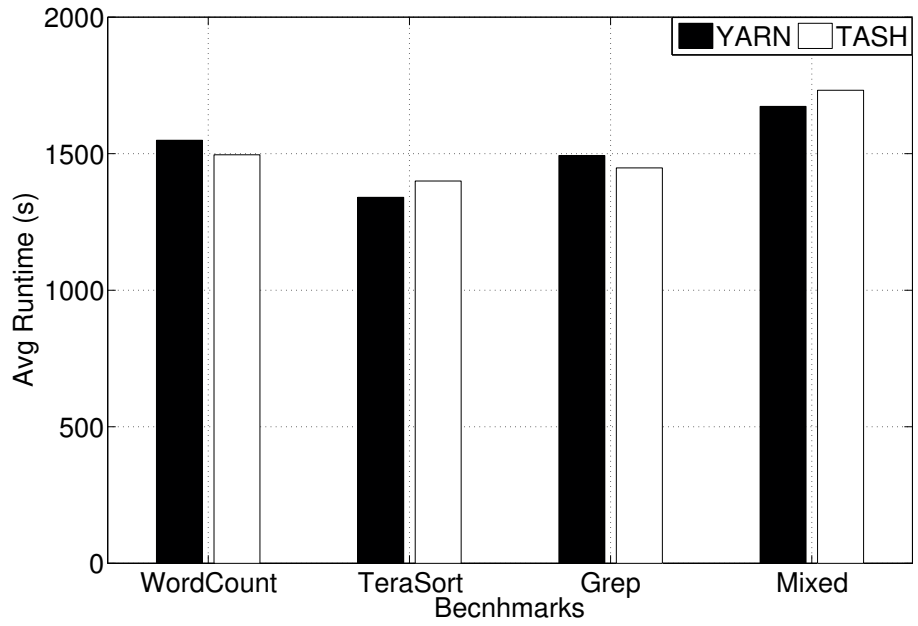


Figure 6.12: Average runtime for benchmarks.

TeraSort performs lot of I/O operations, which lead to lower processor utilization. This lower utilization affects the power utilization by the nodes, which in turn affects the heat re-circulation (maximum inlet temperature) in data center.

Benchmarks	**Max. Inlet temp.** (^oC)		**Cooling Cost (W)**		**Avg. Time (s)**	
	YARN	TASH	YARN	TASH	YARN	TASH
WordCount	20.775985	19.446231	16531.43	14141.85542	1549	1496
TeraSort	20.578716	19.123342	17438.3679	13695.20042	1340	1400
Grep	21.184383	19.949573	19628.25501	16999.95128	1493	1448
Mixed	21.207426	19.950965	19050.26605	16886.58	1673	1732

Table 6.2: Hadoop Benchmarks Results

From the figure 6.11, it can be concluded that not only the container assignment, but the ApplicationMaster's policy of how to utilize the container resources affects the heat re-circulation in the data center. We also run mixed workload containing two jobs- a WordCount and a TeraSort - running simultaneously. It can be seen from figure 6.11(b) that it yields higher heat re-circulation than the cluster running exclusive jobs. We interpret this result as follows. Since the workload is consisting of mixture of I/O intensive and computation

intensive tasks, a node may have been assigned containers for both the workloads. Therefore, while some of the containers are fully utilizing the processor cores assigned them, some are performing I/O operations utilizing disks. Therefore, the power consumption and in turn heat re-circulation in the data center is higher.

Figure 6.12 depicts the average run time for the benchmarks. The figure shows that though TASH does not outperform YARN, performance degradation is within acceptable margin.

6.2.3 Impact of Resource Allocation on Power Consumption

In this section, we will investigate the impact of container allocation on the power consumption of a node. Recall from section 3, that power profile of a cluster in turn impacts the thermal profile of the cluster. In order to better understand how TASH affects the power consumption of a node, we analyze only one of the rack in the data center.

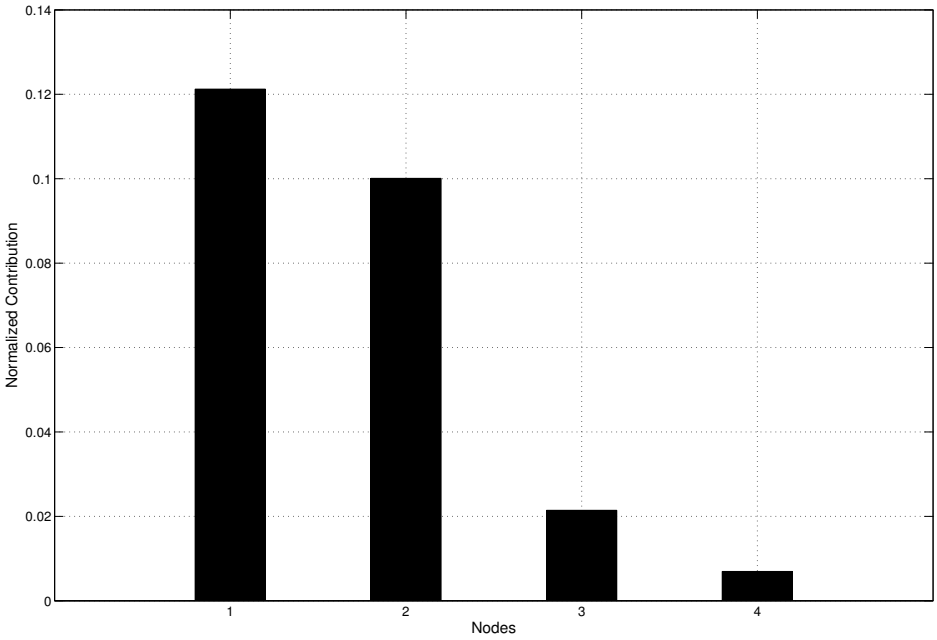
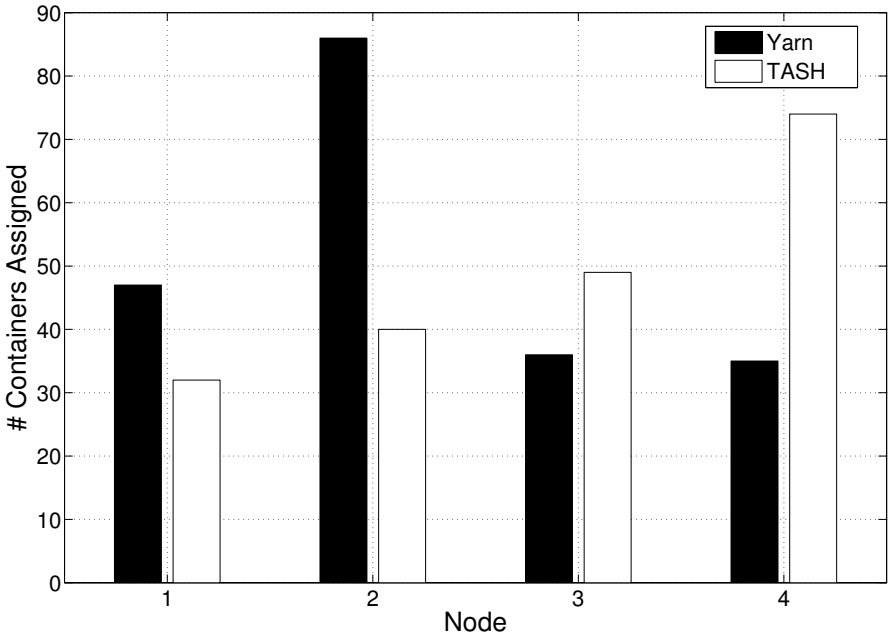
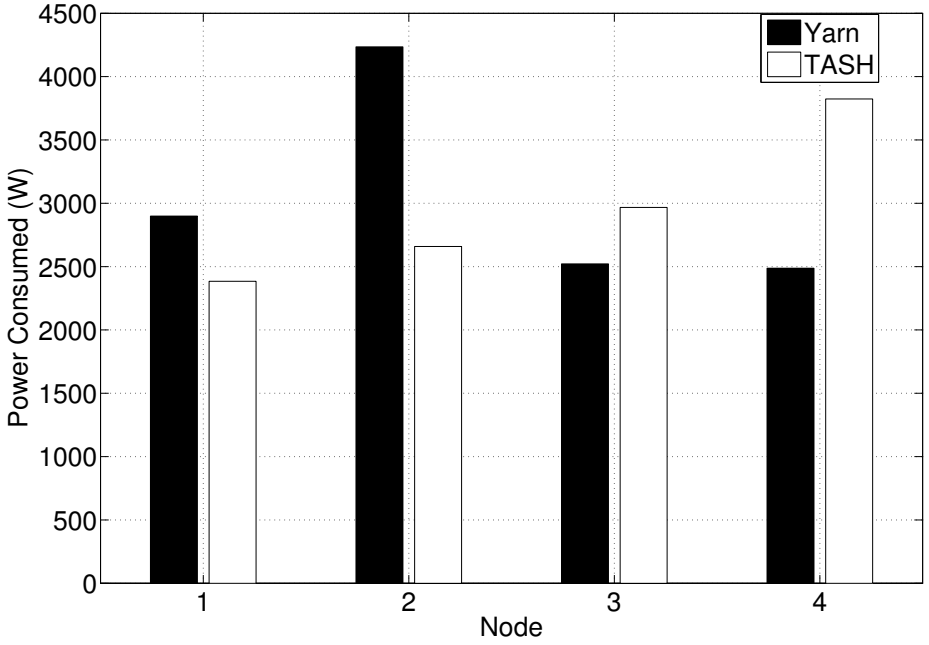


Figure 6.13: Cross Interference Coefficient of Data Center

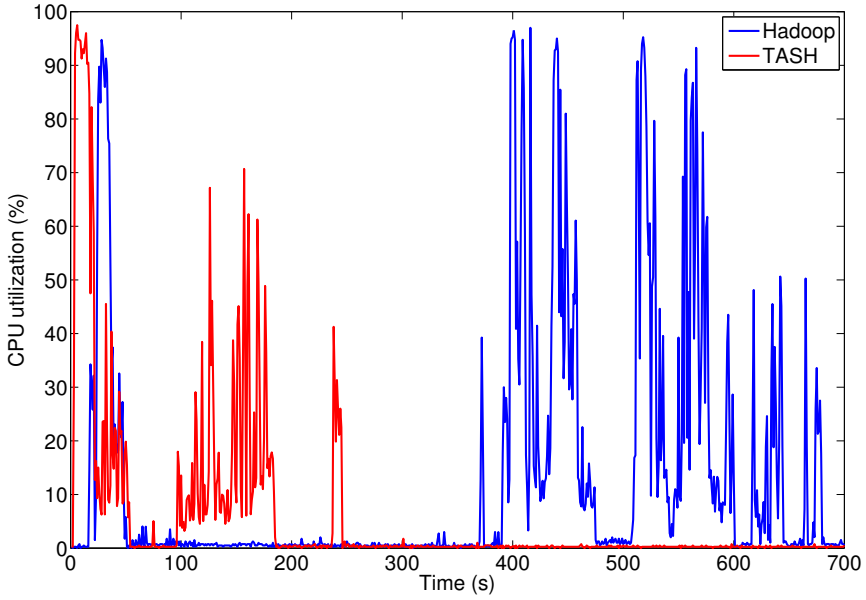


(a) Number of containers assigned to each node in a rack.

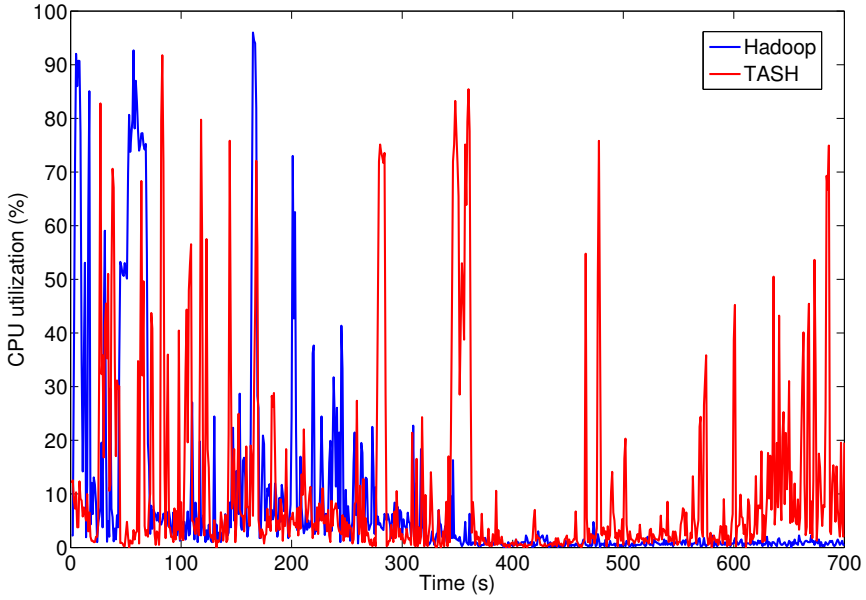


(b) Power consumed by each node in a rack.

Figure 6.14: Figure depicting number of containers assigned and power consumed by each node in a rach under YARN and TASH.



(a) CPU utilization for Node 1



(b) CPU utilization for Node 4

Figure 6.15: CPU utilization of Original Yarn and TASH for running TeraSort.

First of all, it is very important to know which nodes are the major contributor to the heat re-circulation. Figure 6.13 shows the normalized contribution of all the nodes in a rack in the cluster. Note that node 1 is closest to the floor and node 4 is closest to the ceiling

of the room. It can be seen from the figure that node 1 and 2 are the major contributors among four nodes. We ran terasort on a data size of 100GB on our cluster. Figure 6.14(a) depicts the number of containers assigned to each node in a selected rack. It is evident from figure 6.14(a) that TASH assigns less number of containers on the nodes contributing more towards heat re-circulation.

Now, we test whether the container assignment actually affects the CPU utilization or not. In order to understand the CPU utilization of nodes, we selected Node 1 and 4; one with highest and lowest contribution, respectively. Figure 6.15 plots the CPU utilization of two selected nodes. It is important to note that for the sake of simplicity, it only shows map phase of the entire run. It can be observed from figure 6.15 that there is substantial difference between CPU utilization of a node under two policies. In yarn, both nodes are somewhat equally utilized, while TASH utilizes Node 4 more than Node 1. We used the model defined in section 3 to calculate the power consumed by the nodes. Figure 6.14(b) presents the power consumption of each node in a selected rack. It can be seen that TASH significantly re-distributes the power profile of a cluster. In the next section, we will demonstrate the impact of a power profile on inlet temperature and cooling of the data center.

Chapter 7

Extension of File and Resource Allocation Policies

In this chapter we provide insights about some of the anomalous behaviors we observed during evaluating TIGER and describe how we provide solutions to observed anamolies. We also provide detailed guidelines to extend our thermal-aware resource allocation policy for legacy Apache Hadoop version 1 (a.k.a. MR1). The remaining chapter is organized as follows: in section 7.1, we provide detailed design of HybridTIGER- an extension of our file assignment policy TIGER. Section 7.2 talks about implementing resource allocation policy in MR1.

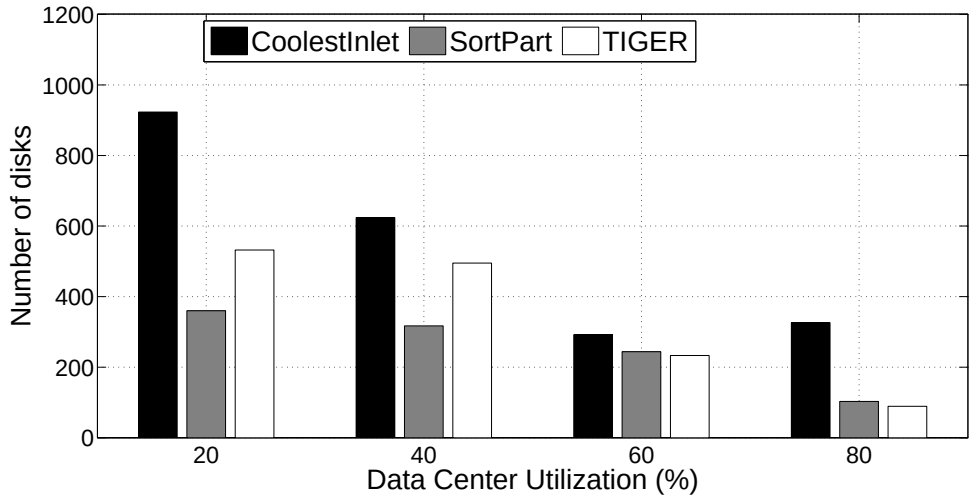
7.1 Extension of File Assignment Policy

In this section, we proposed HybridTIGER -an extension of our original thermal-aware file assignment algorithm for storage clusters. Section 7.1.1 describe anomalous behavior we observed during evaluation of TIGER and provide detailed analysis we performed to pin point the reason behind such behavior. In light of these observations, we tuned our proposed algorithm to rectify this performance issue (see Section 7.1.2). Finally, Section 7.1.3 provides experimental results to confirm that Hybrid TIGER performs better than the TIGER under high workload conditions.

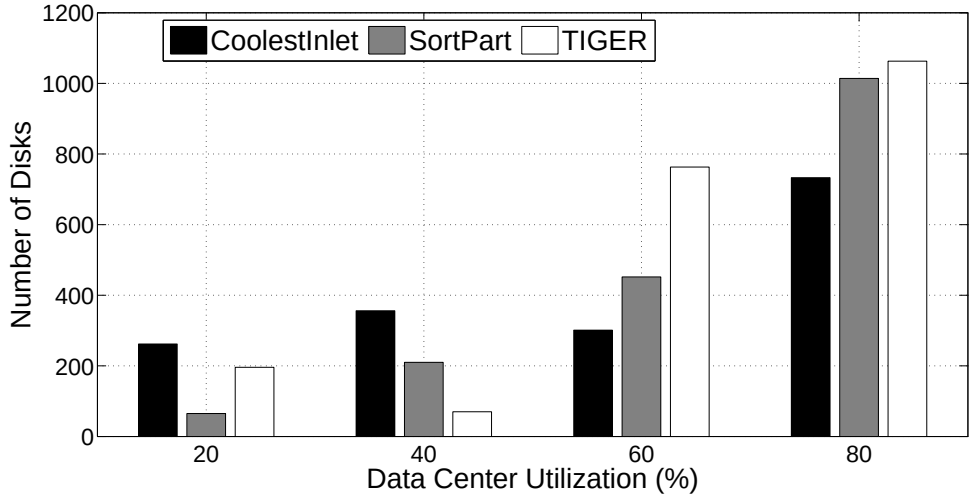
7.1.1 Anomalous Behavior of TIGER

In section 6.1.3, we evaluate the impact of energy-efficient disks on the performance of TIGER. We conduct the experiments for given setup and for data center utilizations ranging from 10% to 90%, increasing in the steps of 10% at each time. We observed that the performance of TIGER in terms of cooling efficiency is slightly lower than CoolestInlet

scheme when the data center utilization exceeds 70%. The same trend can be observed fro Fig. 6.2 and Fig. 6.3 and Fig. 6.7. Moreover, TIGER's performance in terms of response time is longer than CoolestInlet under the same utilization 6.9.



(a) Number of disks with minimum utilization



(b) Number of disks with maximum utilization

Figure 7.1: The number of disks with minimum utilization and the number of disks with maximum utilization. (a) Minimum utilization range: from U_{min} to (U_{min}+0.1) (b) Maximum utilization range: from U_{max}-0.1 to U_{max}

In order to investigate the aforementioned performance issue, let us take a closer look at the disk utilization distribution. We focus on two aspects of the utilization distribution. First, we measure the maximum and minimum disk utilization in a storage cluster. Second,

we quantify the number of disks with the maximum or close to the maximum (i.e., max - 10) utilization; we also consider the number of disks with the minimum or close to the minimum (i.e., min +10) utilization (see Fig. 7.1).We observe from this group of experiments that regardless of disk utilization, a handful of disks are sitting idle when TIGER or SortPart is employed. This trend is true for CoolestInlet when disk utilization ranges anywhere between 10% to 60%; however, when storage cluster's utilization becomes as high as 80%, we observe that the minimum disk utilization jumps to almost 60%. We conclude from this set of experiments that for all the test cases, the maximum disk utilization of CoolestInlet is always above 90%, which is greater than the maximum disk utilization of TIGER and SortPart.

Fig. 7.1(a) illustrates that regardless of the data center utilization, the number of disks with the minimum utilization in CoolestInlet is larger than those in TIGER and SortPart. Fig. 7.1(b) shows that when the data center's utilization is higher than 60%, the number of disks with the maximum utilization in CoolestInlet is smaller than those in TIGER and SortPart. This analysis confirms that I/O workload is more uniformly distributed by TIGER and SortPart than that by CoolestInlet. The disks with the minimum utilization in TIGER and SortPart are sitting idle even when the data center's overall utilization is very high (e.g., 80%). These idle disks result in performance degradation in TIGER (see Fig. 6.9).

7.1.2 HybridTIGER

Algorithm 3 HybridTIGER(file_info, node_info, U^{Th}_{TIGER})

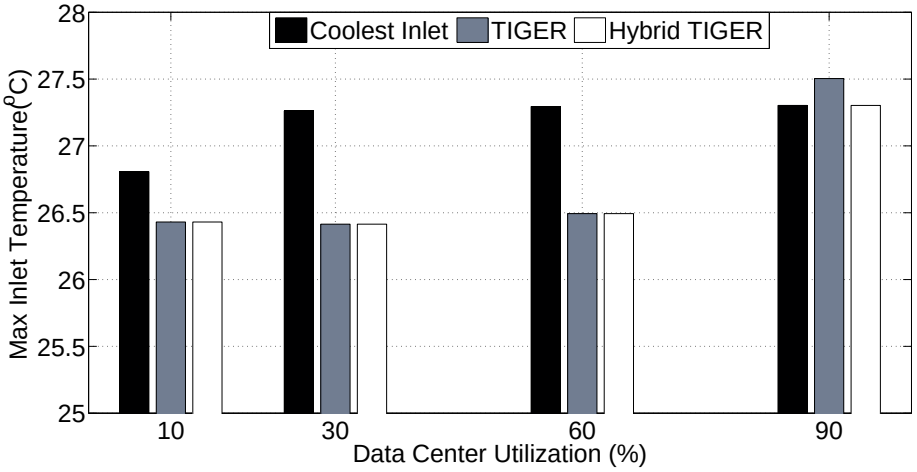
1: $U^{Th}_{total} \leftarrow 0$
2: **for** $f_i \in m$ **do**
3: $\quad U^{Th}_{total} \leftarrow U^{Th}_{total} + s_i \times \lambda_i$
4: **end for**
5: $U^{Th}_{avg} \leftarrow \frac{1}{D} U^{Th}_{total}$
6: **if** $U^{Th}_{avg} < U^{Th}_{TIGER}$ **then**
7: $\quad$ TIGER(file_info, node_info) //see (**??**)
8: **else**
9: $\quad$ Coolest_Inlet(file_info, node_info)
10: **end if**

To address the aforementioned performance problem, we advocate a simple yet wise idea of assigning files to idle disks under heavy workload. Our findings show that CoolestInlet performs better than TIGER under heavy workload. We develop a hybrid algorithm - HybridTIGER - that seamlessly integrates both TIGER and CoolestInlet by dynamically choosing one of these two schemes based on data center utilization. HybridTIGER takes file information, node information, and a threshold U_{TIGER}^{Th} as inputs. There are two major steps in HybridTIGER. First, the average utilization U_{avg}^{Th} is derived from the file and node information (see Lines 1-3 in Algorithm 3). Second, if the average utilization is below threshold U_{TIGER}^{Th}, TIGER will be invoked to assign files to nodes (see Lines 6-7 in Algorithm 3); otherwise, CoolestInlet will be in charge of the file assignment (see Lines 8-9 in Algorithm 3). The pseudo code of HybridTIGER is given in Algorithm 3.

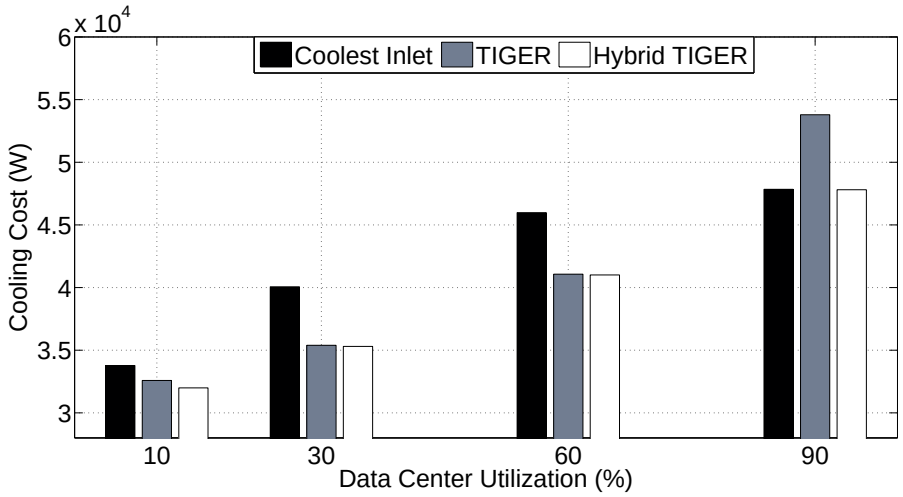
7.1.3 Evaluation of HybridTIGER

We evaluate performance of HybridTIGER in terms of cooling energy conservations. We use the same test described in Section 6.1.2, which contains 50 nodes in two rows of five racks each and each rack contains five nodes (see Fig. 6.1). We maintain initial supply temperature of 15℃ C. Sinice Hybrid TIGER takes advantages of TIGER and coolest inlet policies, we compare Hybrid TIGER's performace with that of TIGER and coolest inlet.

Fig. 7.2 reveals the inlet temperatures and cooling cost of CoolestInlet, TIGER, and HybridTIGER in scenario 1 (see section 6.1.3), where the idle disks are transitioned to sleep mode to offer energy savings. We observed from Fig. 7.2 that when storage cluster's utilization goes up to 60%, HybridTIGER and TIGER share similar performance, which is better than that of CoolestInlet. When utilization become higher than 60%, the performance of HybridTIGER is on par with that of CoolestInlet.



(a) Maximum T^{in}



(b) Cooling Cost

Figure 7.2: Inlet temperatures and cooling cost of CoolestInlet, TIGER, and HybridTIGER in scenario 1, where idle disks are transitioned into the sleep mode to offer energy savings.

7.2 TASH in MR1

Even though YARN is being adapted by most of the data centers, earlier version of the Apache Hadoop (a.k.a. MR1) is also widely in use. In this section, we provide a detailed guideline about how to taylor our thermal aware resource allocation approach for Apache Hadoop version 1.X. In the following section, we will first explain the internals of MR1 to point out the architectural differences between MR1 and MR2 (see section 7.2.1).

Section 7.2.2 provides an overview of the changes in our resource allocation policies to incorporate the architectural differences of MR1, followed by implementation guidelines in Section 7.2.3.

7.2.1 MR1 Internals

MapReduce is one of the most popular programming model for processing and generating large data sets. A MapReduce program is typically useful when simple operations are being performed on huge amount of data. A MapReduce program has a *map* function and a *reduce* function. A map function processes input data and generates some intermediate results in the form of $\langle key, value \rangle$ pairs. These intermediate results are sorted and all the records with same *key* are sent to the same reduce. The reducers then apply processing logic to merge the records and generate final output, which is also in the form of $\langle key, value \rangle$ pairs [9]. Apache Hadoop is an open source software that implements MapReduce programming model. Hadoop framework typically has two parts (1) Hadoop Distributed File System (HDFS), responsible for managing all the data in the Hadoop cluster (2) Hadoop runtime system, which is responsible for accepting jobs from authorized users, scheduling them on the cluster etc. Hadoop library is also designed to detect and handle failures at the application layer delivering high-availability service on the top of cluster of commodity machines.

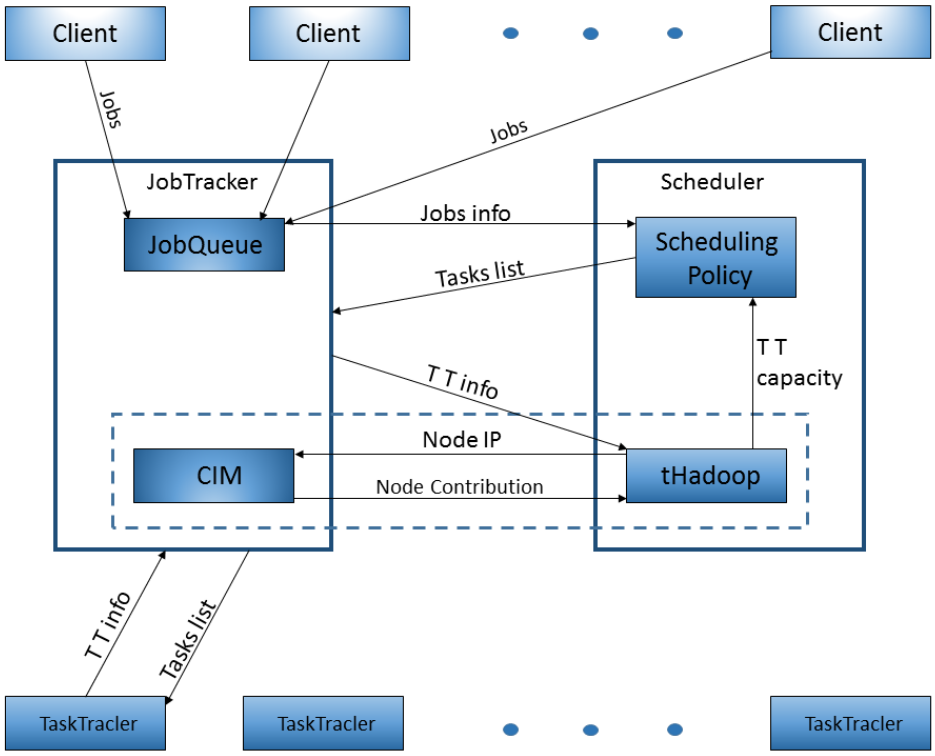
HDFS has two types of nodes and exhibits master-slave paradigm, where each file is partitioned into equal sized chunks and for each chunk, multiple replicas are stored. The master node of the HDFS is called as *Namenode*. It manages the file system namespace by maintaining the file system tree and the metadata for all the files and directories in the tree. It also stores the information of where all the blocks for a given file are located. *Datanodes*, on the other hand, work as slaves. They store and retrieve blocks when requested by client or namenode, and they report back to namenode periodically with list of blocks that they are storing.

Hadoop runtime system also works on master-slave paradigm with *Jobtracker* is a master and *Tasktrackers* as slaves. Jobtracker receives a job from the client, and manages the execution of the job whereas, tasktrackers actually execute the tasks assigned to them by the jobtracker. When a jobtracker receives a job from the client, it divides the job into multiple map tasks; one map task for each chunk; and reduce tasks. It maintains a queue of all the submitted jobs and all the tasks associated with each job. Then it distributes the tasks among all the available tasks based on the scheduling policy (see section). Tasktrackera are slaves and it periodically reports to jobtracker through Heartbeat. A heartbeat indicates that the tasktracker is alive and it also provides the important information such as whether tasktracker is ready to run a new tasks. Once the tasktracker is ready to execute a new task, jobtracker will allocate the tasks from the pool of tasks to be allocated.

7.2.2 Overview

tHadoop addresses the problem of minimizing the heat re-circulation through controlling power consumption of nodes in order to reduce the cooling cost of the data center. It controls the power consumption of the nodes in the data center by constraining the number of tasks to be run on the nodes. tHadoop utilizes the contribution of a node in the heat re-circulation to calculate the number of tasks to be assigned to the node.

tHadoop is a thermal aware scheduler, which can be integrated with any of the existing scheduler in Hadoop to facilitate the JobTracker to make thermal aware workload placement. tHadoop calculates the contribution of nodes in heat re-circulation by using cross interference matrix (see section 4.3.2). Based on the node's contribution and overall workload on the system, tHadoop determines the nummber of tasks to be assigned to particular node. It is important to note that tHadoop does not determines which tasks to schedule on the node, but how many tasks to schedule on the node. This fact allows the Hadoop administrator to pair tHadoop with any of the existing schedule (e.g., Fair Scheduler, Capacity Scheduler etc.) in order to reduce the cooling cost while enjoying the perks of existing algorithms.



architecture.png

Figure 7.3: tHadoop: System Overview

Fig. 7.3 outlines the tHadoop architecture, which has all the important entities of regular Hadoop framework such as clients, JobTracker, Scheduler, and TaskTracker. Although, scheduler is considered as a part of the JobTracker, it is implemented as a separate pluggable class; therefore; in Fig. 7.3, it is shown as a separate entity. The basic sequence of actions can be explained as follows: JobTracker maintains a queue of all the jobs submitted by the clients. It also maintains the information regarding the contribution of each node in the heat re-circulation. Whenever a TaskTracker requests new tasks through Heartbeat mechanism, JobTracker passes the TaskTracker information to the Scheduler. In the Scheduler, tHadoop first fetches the node's contribution and system workload from the JobTracker and calculates the number of tasks to be scheduled on the TaskTracker. Then, based on the underlying scheduling policy, the Scheduler will create the list of tasks to be scheduled on the TaskTracker. Finally, JobTracker will send the list of assigned tasks to the TaskTracker through HeartBeat response.

It is important to note that two building blocks of tHadoop; cross interference matrix and the scheduler; are divided into two entities (JobTracker and Scheduler) from implementation point of view. We will discuss this in detail in section 7.2.3. The parameters affecting scheduling decisions by tHadoop are contribution of the node in the heat re-circulation, maximum capacity of the TaskTracker to run the tasks, total workload on the system at the time, and the number of tasks currently running on the system. Recall from section 3.1, Cross-Interference Matrix depicts the fraction of outlet heat from each node that contributes to the inlet temperature of every other node. The TaskTracker capacity is nothing but the maximum number of tasks a TaskTracker can run simultaneously. This capacity is based on the hardware configuration of the TaskTracker. Finally, the total number map load or reduce load in the system is nothing but the number of tasks waiting in the JobQueue for being launched on the TaskTrackers.

7.2.3 Implementation

In order to implement our thermal aware scheduler in existing Hadoop framework, we need to address two main challenges. First, we have to incorporate our Heat Re-circulation model in the Hadoop Framework (see section 7.2.3). Second, we need to calculate the current TaskTracker capacity based on the contribution of the node in the heat re-circulation (see section 7.2.3).

Cross-Interference Matrix

Recall from section 3.1, Cross-Interference Matrix $A_{N\times N}$ depicts the fraction of outlet heat of each node contributing towards the inlet heat of every other node. As this fraction depends upon the physical layout of the data center, which is very unlikely to change over the long period of time, it is safe to assume that the contribution (fraction of outlet heat) of the node towards heat re-circulation is constant. Therefore, administrator can calculate the

Cross-Interference Matrix every time the physical layout changes, using procedure described in [29].

Table 7.1: Properties in core-site.xml

Sr. No.	Property	Value	Note
1	mapred.cim.path	path of the file containing cross-interference matrix	If not specified, causes fatal error
2	mapred.node.count	Number of nodes in the data center	If not specified, causes fatal error

The administrator has to pass this matrix as a text file, where first field of each line contains the ip address of the node and the remaining fields contain the values for that node. In order to facilitate the administrator to store this file anywhere on the master node, we added two properties; described in Table 7.1; in the configuration file *"core-site.xml"*.

Current TaskTracker Capacity

As mentioned in section 3.1, scheduler can access the node's contribution towards hear re-circulation and modifies the number of tasks assigned to the TaskTracker(s) running on the node. Apache Hadoop implementation provides three pluggable schedulers- FIFO, Fair, and Capacity scheduler. In this study we have implemented our strategy to FIFO scheduler. Before diving into design issues, lets get the internals of FIFO scheduler.

At any given time, the scheduler has access to the all the jobs (being scheduled, run and waiting in the queue) available in the system. Scheduler can also calculate the maximum capacity of all the TaskTrackers in the system. Based on this data, FIFO scheduler first calculates map load factor and reduce load factor for the TaskTracker. These two values depicts the map workload and reduce workload on the system. Map load factor of 1 indicates that the system is under 100% workload and all the available map slots should be used to run the map tasks. Depending upon the map load factor and empty map slots on the TaskTracker, the FIFO scheduler calculates the number of available map slots. It is assumed

that each map tasks requires one map slot. Since it is a FIFO scheduler, the scheduler then creates of a list of map tasks in round robin fashion until all the available map slots are filled. This list is then sent to the TaskTracker as a response to the heartbeat received from the TaskTracker.

It is important to note that to calculate the map load factor, scheduler divides the total number of map tasks yet to be assigned by the total capacity of the the cluster available (still free) at that time. Therefore, we can consider map load factor as current capacity *(C_i)* of the TaskTracker *i*. Therefore, available map slots can be calculated by using following equation:

$$AvailableMapSlots = C_i \times totalAvailableMapSlots \tag{7.1}$$

Therefore, we can use equations (5.7) or (5.8) to modify the map load factor and reduce load factor, which in turn will be used to calculate available map slots and available reduce slots. As mentioned in section 5.2.2, we are not modifying the which tasks to be scheduled, but we are modifying how many tasks to be scheduled. When used with FIFO scheduler, the tasks are selected in round robin fashion. When used with other schedulers, like Fair and Capacity scheduler, the tasks selection is done according to the algorithm provided by the scheduler.

Chapter 8

Conclusion and Future Work

In this dissertation, we have developed thermal-aware file and resource allocation policies to reduce the cooling cost of data centers. This chapter concludes the disseration by summarizing the contributions and describing future directions. The cpater is organized as follows: Section 8.1 highlights the main contributions of the dissertations. In Section 8.2,we concentrate on some future directions, which are extensions of our past and current research on thermal-aware file and resource allocation in data centers.

8.1 Main Contributions

Irrespective of the services offered and client-pool catered, data centers housing thousands of nodes and providing enormous computation and storage capacity have become the backbone of most of the businesses. Maintaining low operational cost of such data centers is one of the principle challenges faced by the designers of the data centers. Since cooling cost incurs significant and re-curring portion of the operational cost, reducing cooling cost plays an important role in maintaining low operational cost. To help achieve this goal, our research has investigated thermal-aware file and resource allocation policies to minimize the heat re-circulation in the data center. In what follows, we summarize the main contribution of this dissertation.

8.1.1 Theraml-Aware File Assignment

In the first phase of this study, we have investigated the impact of file assignment of power consumption and heat re-circulation of the storage clusters in data centers. We have proved that the linear power consumption model used for processors does not work well in

case of storage clusters where disks are major power consumers. In the second phase, we have proposed and implemented *TIGER*, a file assignment approach to reducing cooling energy requirements of data centers [74].

TIGER first decides disk utilization threshold based on contribution of each node towards the heat re-circulation in data center. TIGER then sorts the nodes according to their heat re-circulation contributions, and files are assigned to disks in each node provided that disk utilization is below the corresponding threshold. We applied cross-interference coefficients to estimate the re-circulation of hot air from the outlets to the inlets of storage nodes.

We have evaluated *TIGER*'s performance against existing policies that considers inlet temperature of nodes through simulation studies. Our experimental results confirmed that *TIGER* outperforms existing strategies in terms of cooling cost conservations. Furthermore, *TIGER* successfully in maintains performance penalties withing acceptable margines. More importantly, our research highlights the necessity of different approach to address thermal management issues in storage clusters.

8.1.2 Thermal-aware Resource Allocation

Most data centers now-a-days house some flavors of Hadoop frameworks to support big data analytics workload. In this study, we have addressed the probelm of thermal-management in Hadoop clusters by investigating the impact of resource allocation policies on the heat re-circulation in data center.

The proposed resource allocation approach called *TASH* calculates each node's contribution towards heat re-circulation using cross-interference matrix in a calibration phase. *TASH* then uses this contribution to judiciously allocate resources on the nodes, in order to minimize the heat re-circulation in data centers. The main goal is to minimize the maximum inlet temperature of the nodes in a data center in order to reduce the cooling cost of Hadoop clusters.

We have implemented *TASH* in Apache Hadoop 2.7.3 by modifying the FairScheduler to consider heat re-circulation while making container assignments. We have evaluated our implemented approach through extensive benchmarking using sixteen (16) node in-house cluster. The results showed that our proposed policy achieves significant cooling cost conservation over existing resource allocation policies.

8.2 Future Work

While addressing design challenges to develop a thermal-aware file and resource allocation policies for data centers, we came across serveral interesting issues that are still unresolved. This section overviews some of thse open issues that need further investigation. In addition, this section presents opportunities for future work by highlighting thermal management issues in other application domains that have to be studied in details.

8.2.1 Data Replication

Most of the services supported by big data clusters follow a typical access pattern, where data is written once and read multiple times. Also, there is strong possibility that some of noodes may become unavailable due to system failure. Therefore, most of frameworks supporting such big data analytics services utilizes multiple replicas of data. Data replication plays as important role in improving data avaiability, throughput, and reducing latency.

We are planning to investigate the ways *TIGER* can be extended to accommodate data replicas. Since *TIGER* sorts the files according to the service time and each replica will have the same service time, it is important to establish guidelines to differtiate between replicas of the same file. By differtiating between replicas of the same file, it can be assured that primary and backup copies of the file are placed on different nodes. We will conduct experinents to show that data replica offer *TIGER* ample opportunities to assign files in a away to further reduce cooling cost.

8.2.2 Machine learning in thermal management

Recall (see Section 4.1) that *TIGER* assumes that the access patterns of new files are known a priori. In the first phase, we inted to incorporate a prediction technique to predict file access patterns in a dynamic computing environment. In doing so, TIGER no longer relies on file access patterns given a priori. We firmly beleive the importance of using machine learning algorithms to analyze profiled data in order to make future predictions.

Most of the existing thermal management solutions either employ profiling techniques to predict systems workload or assume that users feed resource requirements beforehand. Although a number of algorithms have been developed to profile and predict the future workload, unfortunately, in a large number of cases workload may dramatically change due to a combination of reasons (e.g., data streaming and changing access patterns). Therefore, it is important to investigate approaches that not only proactively makes resource allocation decisions based on workload predictions but also improves over the time by learning from observed workload and their impact on thermal management.

Machine learning algorithms facilitate us to continuously analyze the data and improve our algorithms based on observed trends. We believe that it is worth investigating machine-learning algorithms to improve thermal-aware scheduling policies. Machine learning algorithms can also be used to extend existing workload-based power consumption models. In this future direction, we aim at developing a set of machine-learning-based file and resource allocation policies, which will enable systems to be self-configured in accordance to dramatically changing workload.

8.2.3 Thermal Management in Other Applications

Number of different big data frameworks(e.g., Spark [75] and Storm [76]) co-exists along with MapReduce framework. Each big data framework supports a specific type of workload (e.g., batch, real-time, inter-active) and offers optimized performnace for that workload

through smart scheduling and optimization policies. We plan to explore opportunities to integrate thermal awareness into such workloads (especially Spark and Storm).

Parallel and distributed databases have been widely used to store massive structured data and to perform data analysis. Databases systems use underlying storage clusters to store database tables and indexes, imposing diversified workload conditions on storage systems. It is our plan to delve into the impacts of parallel and distributed database systems on the thermal efficiency of data centers. Such investigation will lead to productive designs of thermal management in modern data centers. In a long term research, we inted to provide a ubiquitous thermal management strategy that can be self-configured based on the application framework.

8.3 Conclusions

This dissertation has presented thermal-aware file and resource allocation policies for data centers. The experimental results shown in the dissertation have shown that the proposed thermal-aware policies can deliver significant cooling cost reduction by minimizing the heat re-circulation in the data centers housing various types of clusters. In particular, the proposed policies improve cooling energy conservation over existing policies under light to medium-high workload conditions. In a scenario where the system workload is high, our policies can maintain the same level of performance in terms of cooling energy conservation. Furthermore, proposed policies maintains performace penalties within acceptable margine. We have also presented the guidelines to implement the proposed policies in existing big data frameworks - Hadoop.

References

[1] 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.

[2] B. Shi and A. Srivastava, "Thermal and power-aware task scheduling for hadoop based storage centric datacenters," in *Green Computing Conference, 2010 International*, pp. 73–83, 2010.

[3] C. L. Belady, "In the data center, power and cooling costs more than the it equipment it supports," *Electronics cooling*, vol. 13, no. 1, p. 24, 2007.

[4] P. Ranganathan and N. Jouppi, "Enterprise it trends and implications for architecture research," in *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pp. 253–256, IEEE, 2005.

[5] 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, 2008.

[6] J. Koomey, "Growth in data center electricity use 2005 to 2010," *A report by Analytical Press, completed at the request of The New York Times*, 2011.

[7] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the electric bill for internet-scale systems," in *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 123–134, ACM, 2009.

[8] R. Sawyer, "Calculating total power requirements for data centers," *White Paper, American Power Conversion*, 2004.

[9] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, Jan. 2008.

[10] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Job scheduling for multi-user mapreduce clusters," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-55*, 2009.

[11] M. Bhandarkar, "Mapreduce programming with apache hadoop," in *Parallel Distributed Processing (IPDPS), 2010 IEEE International Symposium on*, pp. 1–1, April 2010.

[12] D. Anderson, J. Dykes, and E. Riedel, "More than an interface-scsi vs. ata.," in *FAST*, vol. 2, p. 3, 2003.

[13] Ericsson, "Reliability aspects of power supplies," *Technical Report Design Note 002, Ericsson Microelectronics*, April 2000.

[14] C. Bash and G. Forman, "Cool job allocation: Measuring the power savings of placing jobs at cooling-efficient locations in the data center.," in *USENIX Annual Technical Conference*, vol. 138, p. 140, 2007.

[15] 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.

[16] A. Manzanares, X. Qin, X. Ruan, and S. Yin, "Pre-bud: Prefetching for energy-efficient parallel i/o systems with buffer disks," *Trans. Storage*, vol. 7, pp. 3:1–3:29, June 2011.

[17] Y. Kim, S. Gurumurthi, and A. Sivasubramaniam, "Understanding the performance-temperature interactions in disk i/o of server workloads," in *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, pp. 176–186, Feb 2006.

[18] 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.

[19] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 8, pp. 299–316, June 2000.

[20] A. R. D. Narayanan, A. Donnelly, "Write off-loading: Practical power management for enterprise storage," in *6th USENIX Conference on File and Storage Technologies*, 2008.

[21] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pp. 1–10, 2010.

[22] F. B. Schmuck and R. L. Haskin, "Gpfs: A shared-disk file system for large computing clusters.," in *FAST*, vol. 2, p. 19, 2002.

[23] R. B. Ross, R. Thakur, *et al.*, "Pvfs: A parallel file system for linux clusters," in *in Proceedings of the 4th Annual Linux Showcase and Conference*, pp. 391–430, 2000.

[24] R. Springer, D. K. Lowenthal, B. Rountree, and V. W. Freeh, "Minimizing execution time in mpi programs on an energy-constrained, power-scalable cluster," in *Proceedings of the Eleventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '06, (New York, NY, USA), pp. 230–238, ACM, 2006.

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

[26] Q. Zhu, A. Shankar, and Y. Zhou, "Pb-lru: a self-tuning power aware storage cache replacement algorithm for conserving disk energy," in *Proceedings of the 18th annual international conference on Supercomputing*, ICS '04, (New York, NY, USA), pp. 79–88, ACM, 2004.

[27] Z. Zong, X. Qin, X. Ruan, and M. Nijim, "Heat-based dynamic data caching: A load balancing strategy for energy-efficient parallel storage systems with buffer disks," in *Mass Storage Systems and Technologies (MSST), 2011 IEEE 27th Symposium on*, pp. 1–6, may 2011.

[28] J. Moore, J. Chase, and P. Ranganathan, "Weatherman: Automated, online and predictive thermal mapping and management for data centers," in *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, pp. 155 – 164, june 2006.

[29] Q. Tang, T. Mukherjee, S. K. S. 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, 2006.

[30] L. Yuan and G. Qu, "Analysis of energy reduction on dynamic voltage scaling-enabled systems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 12, pp. 1827–1837, 2005.

[31] I. Hong, M. Potkonjak, and M. B. Srivastava, "On-line scheduling of hard real-time tasks on variable voltage processor," in *Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pp. 653–656, ACM, 1998.

[32] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," in *Low Power Electronics and Design, 1998. Proceedings. 1998 International Symposium on*, pp. 197–202, IEEE, 1998.

[33] T. Pering, T. Burd, and R. Brodersen, "Voltage scheduling in the iparm microprocessor system," in *Low Power Electronics and Design, 2000. ISLPED'00. Proceedings of the 2000 International Symposium on*, pp. 96–101, IEEE, 2000.

[34] Y. Shin and K. Choi, "Power conscious fixed priority scheduling for hard real-time systems," in *Design Automation Conference, 1999. Proceedings. 36th*, pp. 134–139, IEEE, 1999.

[35] G. Quan and X. Hu, "Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors," in *Design Automation Conference, 2001. Proceedings*, pp. 828–833, IEEE, 2001.

[36] A. Wierman, L. L. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems," in *INFOCOM 2009, IEEE*, pp. 2007–2015, IEEE, 2009.

[37] A. Gandhi, M. Harchol-Balter, R. Das, and C. Lefurgy, "Optimal power allocation in server farms," in *Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems*, SIGMETRICS '09, (New York, NY, USA), pp. 157–168, ACM, 2009.

[38] B. Khargharia, S. Hariri, and M. Yousif, "Autonomic power and performance management for computing systems," in *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, pp. 145 – 154, june 2006.

[39] E. Pinheiro, R. Bianchini, E. V. Carrera, and T. Heath, "Load balancing and unbalancing for power and performance in cluster-based systems," 2001.

[40] A. Manzanres, X. Ruan, S. Y., M. Nijim, W. Luo, and X. Qin, "Energy-aware prefetching for parallel disk systems: Algorithms, models, and evaluation," in *Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on*, pp. 90 –97, july 2009.

[41] A. E. Papathanasiou and M. L. Scott, "Energy efficient prefetching and caching," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '04, (Berkeley, CA, USA), pp. 22–22, USENIX Association, 2004.

[42] Q. Li, J. Li, L. Shi, M. Zhao, C. J. Xue, and Y. He, "Compiler-assisted stt-ram-based hybrid cache for energy efficient embedded systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, pp. 1829–1840, Aug 2014.

[43] H. Zhang, P. V. Rengasamy, S. Zhao, N. C. Nachiappan, A. Sivasubramaniam, M. T. Kandemir, R. Iyer, and C. R. Das, "Race-to-sleep + content caching + display caching: A recipe for energy-efficient video streaming on handhelds," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, (New York, NY, USA), pp. 517–531, ACM, 2017.

[44] N. Nishikawa, M. Nakano, and M. Kitsuregawa, "Energy efficient storage management cooperated with large data intensive applications," in *2012 IEEE 28th International Conference on Data Engineering*, pp. 126–137, April 2012.

[45] 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.

[46] L. Wang, G. von Laszewski, J. Dayal, X. He, A. Younge, and T. Furlani, "Towards thermal aware workload scheduling in a data center," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, pp. 116 –122, dec. 2009.

[47] Z. Abbasi, G. Varsamopoulos, and S. K. Gupta, "Thermal aware server provisioning and workload distribution for internet data centers," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, (New York, NY, USA), pp. 130–141, ACM, 2010.

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

[49] W. Lang and J. M. Patel, "Energy management for mapreduce clusters," *Proc. VLDB Endow.*, vol. 3, pp. 129–139, Sept. 2010.

[50] R. Kaushik and M. Bhandarkar, "Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster," in *Proceedings of the 2010 international conference on Power aware computing and systems*, HotPower'10, (Berkeley, CA, USA), pp. 1–9, USENIX Association, 2010.

[51] R. T. Kaushik, L. Cherkasova, R. Campbell, and K. Nahrstedt, "Lightning: Self-adaptive, energy-conserving, multi-zoned, commodity green cloud storage system," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, HPDC '10, (New York, NY, USA), pp. 332–335, ACM, 2010.

[52] D. ÃĞavdar, L. Y. Chen, and F. AlagÃűz, "Green mapreduce for heterogeneous data centers," in *2014 IEEE Global Communications Conference*, pp. 1120–1126, Dec 2014.

[53] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-time tasks oriented energy-aware scheduling in virtualized clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 168–180, 2014.

[54] G. B. Barone, V. Boccia, D. Bottalico, and L. Carracciuolo, "Ecco: An integrated solution for environment compatible computing systems," in *2014 International Conference on Intelligent Networking and Collaborative Systems*, pp. 545–550, Sept 2014.

[55] I. n. Goiri, T. D. Nguyen, and R. Bianchini, "Coolair: Temperature- and variation-aware management for free-cooled datacenters," *SIGARCH Comput. Archit. News*, vol. 43, pp. 253–265, Mar. 2015.

[56] T.-C. Tang and Y.-S. Chen, "Thermal-aware mapreduce real-time scheduling in heterogeneous server systems," in *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, RACS '16, (New York, NY, USA), pp. 207–212, ACM, 2016.

[57] 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, pp. 71–84, Jan 2014.

[58] Í. Goiri, K. Le, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "Greenhadoop: leveraging green energy in data-processing frameworks," in *Proceedings of the 7th ACM european conference on Computer Systems*, pp. 57–70, ACM, 2012.

[59] I. n. Goiri, K. Le, M. E. Haque, R. Beauchea, T. D. Nguyen, J. Guitart, J. Torres, and R. Bianchini, "Greenslot: Scheduling energy consumption in green datacenters," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, (New York, NY, USA), pp. 20:1–20:11, ACM, 2011.

[60] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. Andrew, "Greening geographical load balancing," in *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '11, (New York, NY, USA), pp. 233–244, ACM, 2011.

[61] Y. Guo, Y. Gong, Y. Fang, P. P. Khargonekar, and X. Geng, "Energy and network aware workload management for sustainable data centers with thermal storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 2030–2042, Aug 2014.

[62] C. Chen, B. He, and X. Tang, "Green-aware workload scheduling in geographically distributed data centers," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pp. 82–89, Dec 2012.

[63] Z. Niu, B. He, and F. Liu, "Joulemr: Towards cost-effective and green-aware data processing frameworks," *IEEE Transactions on Big Data*, vol. PP, no. 99, pp. 1–1, 2017.

[64] R. Sharma, C. Bash, C. Patel, R. Friedrich, and J. Chase, "Balance of power: dynamic thermal management for internet data centers," *Internet Computing, IEEE*, vol. 9, pp. 42 – 49, jan.-feb. 2005.

[65] R. T. Kaushik and K. Nahrstedt, "T: a data-centric cooling energy costs reduction approach for big data analytics cloud," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, (Los Alamitos, CA, USA), pp. 52:1–52:11, 2012.

[66] L.-W. Lee, P. Scheuermann, and R. Vingralek, "File assignment in parallel i/o systems with minimal variance of service time," *Computers, IEEE Transactions on*, vol. 49, no. 2, pp. 127–140, 2000.

[67] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, "Apache hadoop yarn: Yet another resource negotiator," in *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, (New York, NY, USA), pp. 5:1–5:16, ACM, 2013.

[68] B. Dhruba, "Hdfs architecture guide," in *http://pristinespringsangus.com/hadoop/docs/hdfs_ design.pd* 2008.

[69] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The google file system," *SIGOPS Oper. Syst. Rev.*, vol. 37, pp. 29–43, Oct. 2003.

[70] X. Cheng, C. Dale, and J. Liu, "Statistics and social network of youtube videos," in *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pp. 229–238, 2008.

[71] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, (New York, NY, USA), pp. 15–28, ACM, 2007.

[72] S. Mitra, M. Agrawal, A. Yadav, N. Carlsson, D. Eager, and A. Mahanti, "Characterizing web-based video sharing workloads," *ACM Trans. Web*, vol. 5, pp. 8:1–8:27, May 2011.

[73] J. Summers, T. Brecht, D. Eager, and B. Wong, "Methodologies for generating http streaming video workloads to evaluate web server performance," in *Proceedings of the 5th Annual International Systems and Storage Conference*, SYSTOR '12, (New York, NY, USA), pp. 2:1–2:12, ACM, 2012.

[74] A. Chavan, M. I. Alghamdi, X. Jiang, X. Qin, M. Qiu, M. Jiang, and J. Zhang, "Tiger: Thermal-aware file assignment in storage clusters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, pp. 558–573, Feb 2016.

[75] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, (Berkeley, CA, USA), pp. 10–10, USENIX Association, 2010.

[76] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. Ryaboy, "Storm@twitter," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, (New York, NY, USA), pp. 147–156, ACM, 2014.