GREENHDFS: DATA-CENTRIC AND CYBER-PHYSICAL ENERGY MANAGEMENT SYSTEM FOR BIG DATA CLOUDS

BY

RINI KAUSHIK

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Professor Klara Nahrstedt, Chair
Professor Laxmikant Sanjay Kale
Professor Tarek Abdelzaher
Professor Remzi Arpaci-Dusseau, University of Wisconsin, Madison

ABSTRACT

Explosion in Big Data has led to a rapid increase in the popularity of Big Data analytics. With the increase in the sheer volume of data that needs to be stored and processed, storage and computing demands of the Big Data analytics workloads are growing exponentially, leading to a surge in extremely large-scale Big Data cloud platforms, and resulting in burgeoning energy costs and environmental impact.

The sheer size of Big Data lends it significant data movement inertia and that coupled with the network bandwidth constraints inherent in the cloud's cost-efficient and scale-out economic paradigm, makes data-locality a necessity for high performance in the Big Data environments. Instead of sending data to the computations as has been the norm, computations are sent to the data to take advantage of the higher data-local performance. The state-of-the-art run-time energy management techniques are job-centric in nature and rely on thermal- and energy-aware job placement, job consolidation, or job migration to derive energy costs savings. Unfortunately, data-locality requirement of the compute model limits the applicability of the state-of-the-art run-time energy management techniques as these techniques are inherently data-placement-agnostic in nature, and provide energy savings at significant performance impact in the Big Data environment.

Big Data analytics clusters have moved away from shared network attached storage (NAS) or storage area network (SAN) model to completely clustered, commodity storage model that allows direct access path between the storage servers and the clients in interest of high scalability and performance. The underlying storage system distributes file chunks and replicas across the servers for high performance, load-balancing, and resiliency. However, with files distributed across all servers, any server may be participating in the reading, writing, or computation of a file chunk at any time. Such a storage model complicates scale-down based power-management by making it hard to generate significant periods of idleness in the Big Data analytics clusters.

GreenHDFS is based on the observation that data needs to be a first-class object in energy-

management in the Big Data environments to allow high data access performance. GreenHDFS takes a novel data-centric, cyber-physical approach to reduce compute (i.e., server) and cooling operating energy costs. On the physical-side, GreenHDFS is cognizant that all-servers-are-not-alike in the Big Data analytics cloud and is aware of the variations in the thermal-profiles of the servers. On the cyber-side, GreenHDFS is aware that all-data-is-not-alike and knows the differences in the data-semantics (i.e., computational jobs arrival rate, size, popularity, and evolution life spans) of the Big Data placed in the Big Data analytics cloud. Armed with this cyber-physical knowledge, and coupled with its insights, predictive data models, and run-time information GreenHDFS does proactive, cyber-physical, thermal- and energy-aware file placement, and data-classification-driven scale-down, which implicitly results in thermal- and energy-aware job placement in the Big Data analytics cloud compute model. GreenHDFS's data-centric energy- and thermal-management approach results in a reduction in energy costs without any associated performance impact, allows scale-down of a subset of servers in spite of the unique challenges posed by Big Data analytics cloud to scale-down, and ensures thermal-reliability of the servers in the cluster.

GreenHDFS evaluation results with one-month long real-world traces from a production Big Data analytics cluster at Yahoo! show up to 59% reduction in the cooling energy costs while performing 9x better than the state-of-the-art data-agnostic cooling techniques, up to a 26% reduction in the server operating energy costs, and significant reduction in the total cost of ownership (TCO) of the Big Data analytics cluster. GreenHDFS provides a software-based mechanism to increase energy-proportionality even with non-energy-proportional server components.

Free-cooling or air- and water-side economization (i.e., use outside air or natural water resources to cool the data center) is gaining popularity as it can result in significant cooling energy costs savings. There is also a drive towards increasing the cooling set point of the cooling systems to make them more efficient. If the ambient temperature of the outside air or the cooling set point temperature is high, the inlet temperatures of the servers get high which reduces their ability to dissipate computational heat, resulting in an increase in server temperatures. The servers are rated to operate safely only with a certain temperature range, beyond which the failure rates increase. GreenHDFS considers the differences in the thermal-reliability-driven load-tolerance upper-bound of the servers in its predictive thermal-aware file placement and places file chunks in a manner that ensures that temperatures of servers don't exceed temperature upper-bound. Thus, by ensuring thermal-reliability at all times and by lowering the overall temperature of the servers, GreenHDFS enables data centers to enjoy energy-saving economizer mode for longer

periods of time and also enables an increase in the cooling set point.

There are a substantial number of data centers that still rely fully on traditional air-conditioning. These data centers can not always be retrofitted with the economizer modes or hot- and cold-aisle air containment as incorporation of the economizer and air containment may require space for ductwork, and heat exchangers which may not be available in the data center. Existing data centers may also not be favorably located geographically; air-side economization is more viable in geographic locations where ambient air temperatures are low for most part of the year and humidity is in the tolerable range. GreenHDFS provides a software-based approach to enhance the cooling-efficiency of such traditional data centers as it lowers the overall temperature in the cluster, makes the thermal-profile much more uniform, and reduces hot air recirculation, resulting in lowered cooling energy costs.

To my mother, for teaching me to never give up and to my father, for unconditionally loving and believing in me.

ACKNOWLEDGMENTS

I am grateful to my advisor Professor Klara Nahrstedt for her support, advices, and guidance throughout my Ph.D. process. Starting from the qualification exam time, she listened to and gave feedback on my presentations, patiently read through my poorly written initial paper drafts, and gave me detailed feedback on making my research better. I am thankful to my mentor and committee member Professor Remzi Arpaci-Dusseau for his guidance, mentoring, and advice starting from the one semester spent at University of Wisconsin, Madison. I am thankful to Professor YuanYuan Zhou for her mentoring that made a huge difference in my life and attitude. I am thankful to my committee members Professor Sanjay Kale and Professor Tarek Abdelzaher for their advices and suggestions. I had a great time in Professor Tarek Abdelzaher's Green Computing class and am thankful for the learnings I received in the class.

I am most grateful to my father for loving me unconditionally and for always believing in me. He wanted to see a doctor in front of my name, and this Ph.D. is his dream come true. He considered education to be more important than any other aspect in life and instilled in us a deep desire to learn. He was a researcher and an analytical thinker and I owe all my analytical skills to him. He was one of the most trustworthy, strong, dependable, and responsible person I have ever met. He is dearly missed!

I am thankful to my husband, Susheel, for his emotional support and motivation during my ups and downs in the Ph.D. phase. He made the major decision of relocating from California to the mid-west on my behalf just so that I could do my Ph.D. He left his friends, his hikes, and the wonderful climate of California to shiver in the cold weather of mid-west. He had to endure significant amount of grueling back-and-forth travel between Champaign and California every week in the process as well.

I am very grateful to my mother for her never give up attitude and for her immense support during the Ph.D. process. She stayed with me and helped out with my son whenever I had

a major deliverable like the qualification examination, preliminary examination, or some major paper deadlines. I am thankful to my son, Karun, for understanding even at a very young age of 4 years, my commitment to the Ph.D. process and the associated considerable workload. He made hand-made good luck cards for me for every important Ph.D. milestone such as paper deadlines, preliminary examination, and final defense which I will cherish forever. Karun even heard my entire final defense over skype at 1AM in the morning (he was in India then)! And, while I was working on the thesis write-up, he even made green tea to help me stay up.

I am grateful to my sisters who understand me better than I understand myself and for all their prayers and best wishes and efforts in supporting me. My sister, Priti, for motivating me to do the best I can, for encouraging me, for understanding me, and for praying constantly for my success. My sister, Anu, for believing in me, listening to me non-judgmentally, and supporting me. My brother-in-law, Kabir, for listening to my issues very patiently, for sharing his experiences, and for giving me excellent advice. My brother-in-law, Sanjeev, for positively motivating, and encouraging me. My sister, Tanu, for encouraging me to do Computer Science instead of Biomedical Engineering, for her help with filling out my applications, her guidance, support, and care. And, my brother-in-law, Praveen, for being there. Finally, I am thankful to my parents-in-law for their blessings.

I am thankful to Buntu Mama for listening to my concerns at a time when I was going through a low phase in the Ph.D. in 2011, for showing me the strengths of my research which gave me added motivation and boost to continue with the research. I am thankful to my friends Ramana Yerneni and Kumar Goswami for giving me guidance at a time when I was grappling with the major decision of pursuing a Ph.D.

I am thankful to my friend, Amrita Sehrawat, for hosting me for many days when I was working on the thesis writeup. I am glad to have found a life-long friend in her and am amazed by her strength of character. I am thankful to all my teachers: Mrs. Bhalla, Mrs. Laxmana, Mrs. Choudhary, Mrs. Raichoudhary, Mrs. Chandana, and Mr. Singh who believed that I was their best student and saw more potential in me than what I saw myself.

I am thankful to Milind Bhandarkar, Ryota Egashira, Arun Sunderasan, Ram Marti, Milton Smith, and Raj Merchia for the discussions, paper reviews and for the help with the approval process at Yahoo!. I am very thankful to KeeSiong Ng and his wife Ginny for reviewing the thesis, giving me valuable pointers on the same and for all their help.

I am grateful to Shan Lu for being a great mentor to me during my first semester at University of Illinois, Urbana-Champaign. I am thankful to my various colleagues in the MONET group such as Debish, Raoul, Arefin, Zixia, Huang, Thadpong, and Long for listening to my presentations, and providing me feedback prior to every conference and major event.

And, I am very glad to have met wonderful people in the academic office at University of Illinois, Urbana-Champaign who helped me tremendously throughout the Ph.D. process. Mary Beth Kelley, Rhonda McElroy, Kathleen Runck, Elaine Wilson, and Lynette Lubben were really helpful throughout and it was a pleasure interacting with them.

Finally, I am very grateful for the internship at Yahoo! which gave me access to all the data I needed for the GreenHDFS research. I had started working on GreenHDFS in 2009, and really wanted real-world data to show the efficacy of GreenHDFS. However, I was unable to find the data because of contractual issues in spite of trying for six-months. Then, the internship at Yahoo! came across, and lo and behold, I had access to all the data I needed. I burned lot of midnight oil to run as many experiments and analysis as I could using the real-world data at Yahoo!.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In today's digital world, Big Data is ubiquitous in myriad forms such as high-definition videos, movies, photos, scientific simulations, medical records and images, financial transactions, phone records, genomics data sets, seismic images, climatology and weather records, and geo-spatial maps. Social media sites such as Facebook and YouTube store billions of pictures and videos whose total size is already in excess of petabytes. Sensors in smart phones, human bodies, vehicles, smart energy meters, servers, and traffic controllers continuously generate significant amount of sensed data, which is then used to provide value-added services such as power usage monitoring and dynamic power pricing, location sensing, targeted advertising, traffic condition monitoring, patient management, and care for the elderly. International Data Corporation (IDC) predicts that the total digital data will exceed 35.2 zettabytes by 2020 [7]. And, this data is only expected to grow every year with the projected growth rate of 40% [101].

Explosion in Big Data [33] has led to a surge in Big Data analytics [48] which revolves around performing computations over extremely large data sets. There is a wealth of information hidden in the Big Data that can be leveraged for significant value add across technology frontiers. Internet services and e-commerce companies such as Google, Facebook, Yahoo!, Amazon, Microsoft, Twitter, and eBay mine petabytes of web logs daily to generate better business insights and to predict user interests for targeted advertising. Myriad use cases of Big Data analytics range from fraud detection in financial transactions, system log processing to identify errors, satellite image processing for astronomical discoveries, seismic data analysis for earthquake predictions, geo-spatial mapping for oil exploration, mail anti-spam detection, machine learning and data mining to build predictive user interest models for advertising optimizations, improving security through better intelligence gathering, portfolio risk management in real-estate, and all the way to genomics research for disease profiling, drug discoveries, and better health-care management. There are many Big Data platforms and analytical solutions available such as IBM's InfoSphere BigInsights [73], HP's Vertica [70], EMC's Greenplum [56], Oracle's Exadata [112], and open-source Hadoop

[144]. The data management and analytics software business is expanding significantly and is estimated to be almost $100 billion [54] already.

1.1 Motivation

With the sheer volume of the data that needs to be stored and processed, storage and computing demands of the Big Data analytics workloads are on a rapid increase. Big Data requires infrastructure that is capable of massive scale-out at economies-of-scale. This has led to an increase in big cloud data centers that provide Big Data analytics as software-as-a-service (SaaS). Gartner predicts that by 2015, world-wide data center hardware spending will surpass $126.2 billion and 71 percent of this spending will come from the big data center class [1]. The huge infrastructure brings in its wake burgeoning energy costs as over the lifetime of IT equipment, the operating energy costs are comparable to and may even exceed the initial equipment acquisition costs [32], and constitute a significant part of the total cost of ownership (TCO) of a data center [39,86,116]. Energy costs of powering and cooling the IT equipment are the dominant costs in the overall energy costs of the data centers [15, 59].

Energy efficiency of data centers is defined in terms of power usage effectiveness (PUE), which is the ratio of the total building power to the IT power. A PUE value of 1 is ideal as it entails that all the building power is consumed by the IT equipment and there are no power overheads. However, majority of the data centers still have a high PUE of 1.8 as per a 2011 survey of 500 data centers [2], and cooling system's power overhead is major culprit in the high PUE. Given the high power densities prevalent in today's data centers, a lot of heat is generated by the computing equipment and efficient removal of the heat is a necessity. The goal of cooling system in the data centers is to maintain the server temperatures within a safe operating range as increase in a server's temperature adversely impacts its reliability and lifetime. For every one watt of power consumed by the computing infrastructure, another one-half to one watt is consumed in powering the cooling infrastructure [113]. Environmental impact of cooling is also significant: a traditional 15 kilowatts water-chiller based data center uses a whopping 360,000 gallons of water a day for cooling [3]. Data centers in US alone emit an upwards of 170,000 metric tons of carbon dioxide annually and this number is only expected to quadruple in the coming years. Hence, reduction in overall (i.e., both server and cooling) operating **energy costs** and needs of the extremely large-scale, Big Data analytics clouds has become an urgent priority [40, 57].

Cooling energy costs are proportional to the difference in the temperature of the hot air exhausted by the servers in the data center and the cooling set point temperature of the cooling system. One way to curtail the cooling energy costs is to raise the cooling set point temperature of the cooling system which also has a beneficial side-effect of increasing the efficiency of the cooling system [27], further reducing the cooling energy costs. And, the other way is to use air-side or water-side economization (free-cooling) fully or in conjunction with traditional cooling systems [110]. In economization or free-cooling mode, outside air is used to cool the data center bypassing the traditional cooling systems such as chillers and air conditioners which are high cooling power consumers; the cooling energy costs of the data centers can be cut down if they are capable of supporting and operating for extended periods of time in the economizer mode.

The cooling-efficacy in the economizer mode depends on the temperature and the quality of the outside air. If the ambient temperature of the air is high in the economizer mode or if the set point temperature of the air conditioner is high, the inlet temperatures of the servers may get high, thereby reducing their cooling-efficiency, and resulting in their temperature rise. Now, the servers can safely operate only within their rated, allowable temperature range; if the server temperature goes out of the allowable range, failure rates increase. For example, the safe operating temperature range of Dell Blade is 50^oF - 95^oF, IBM Blade is 50^oF - 95^oF, and NetAppStorage is 50^oF - 104^oF. Thus, a thermal energy management technique is needed that ensures **thermal-reliability** of the servers in addition to reducing cooling energy costs. Ensuring thermal-reliability is much more important in the purely free-cooled data centers as they are totally at the mercy of the ambient temperatures and conditions.

While several companies such as Microsoft, Google, and Yahoo! have announced new data centers that rely only on free-cooling or air- and water-side economization, there are a substantial number of traditional data centers that still rely on air- or liquid-cooling either entirely or in-part coupled with economization. The existing data centers can not always be retrofitted with the economizer modes as incorporation of the economizers may require space for ductwork, and addition of heat exchangers which may not be available in the data center. Based on the location of the older data centers, free-cooling may or may not be an option as free-cooling is sensitive to outside ambient air temperature and climatic conditions; it is easier and more viable to use in locations where ambient air temperatures are low for most part of the year and humidity is in the tolerable range.

To make operation in economizer mode feasible for longer periods of time and many more

geographic locations, American Society of Heating, Refrigerating and Air-conditioning Engineers (ASHRAE) has even increased the allowable operating safety temperature range of the servers [27]; however, servers need to be designed to be able to support that range before the economization modes can be introduced. And, it just may be too expensive for some of the existing data centers to completely overhaul their existing servers in interest of new servers with higher thermal tolerance. Furthermore, a transition between the economizer and a traditional cooling mode should happen reliably during unfavorable climate conditions for the economizer mode. However, the reliability may not be as good in a retrofitted system, and hence, presents an additional concern to retrofitting economizer mode in the existing data centers. Thus, traditionally cooled data centers abound, and a cooling energy costs reduction and thermal-management technique that works for **both traditionally cooled, and new free-cooled** data centers is needed.

In addition to reducing energy costs and ensuring thermal-reliability, **energy-proportionality** is increasingly becoming an important consideration [28]. In an ideal, perfectly energy-proportional system, almost no power should be consumed when the system is idle and power consumption should increase in proportion to the increase in the activity level. In reality, instead of consuming negligible power, the server components consume almost 40%-70% [28] of the peak power during idle utilization. Main culprits are the memory systems, disk drives, and the networking equipment, and not the processors as their dynamic range (i.e., the range between the power draw at peak utilization vs. idle utilization) is much lower than that of the processor. In fact, processors that allow voltage and frequency scaling are the most energy-proportional server components. The dynamic range is approximately 2.0x for memory (DRAM DIMM consumes 3.5-5W at peak utilization and 1.8-2.5W at idle utilization), 1.2x for disks (Seagate Barracuda ES.2 1TB consumes 11.16W at peak utilization and 9.29W at idle utilization), and less than 1.2x for networking switches.

The ground reality is grim. A six-month study of 5000 servers in a compute cluster at Google shows that servers spend majority of the time in the 10%-50% utilization range. This is a very energy-inefficient range as the power consumption of the these servers can be as high as 70% of the peak power at 10% utilization, and as high as 84% of the peak power at 50% utilization [68]. Energy-proportionality cannot be achieved through processor optimizations such as dynamic voltage and frequency scaling (DVFS) alone (as has been the focus), and requires improvements across all server components. Some non-energy-proportional components such as the disks require greater innovation to be energy-proportional. Disk drives consume significant

amount of power simply to keep the platters spinning, possibly as much as 70% of their total power for high RPM drives [65]. Now, hardware-level innovations to make server hardware components energy-proportional may take significant time. Hence, there is an need to come up with software-based resource management policies to make Big Data analytics clusters energy-proportional even with non-energy-proportional server components.

1.2 Limitations of Existing Energy Management Techniques

The Big Data Analytics cloud data centers have very different compute, energy and storage models compared to those of traditional data centers. This brings forth many new and unique challenges to energy management [68] techniques as outlined below.

1.2.1 Cooling Energy Management

Majority of the existing research on run-time reduction of cooling energy costs relies on thermal-aware computational job placement/migration to reduce the cooling energy costs [16, 29, 106, 107, 113, 126, 134, 137]. These techniques either aim to place incoming computationally intensive jobs in a thermal-aware manner on servers with lower run-time temperatures or attempt to reactively migrate jobs from high run-time temperature servers to servers with lower run-time temperatures. These techniques are inherently *data-placement-agnostic* in nature and work very well when servers are state-less, data resides on a shared storage area network (SAN) or network attached storage (NAS) device, data sizes are small, and data can be sent to the computation without network bandwidth constraints.

Big Data analytics cloud mandates a different compute model which presents a significant challenge to the existing run-time cooling techniques. The network bandwidth constraints of the commodity network switches and the huge data sizes of Big Data render sending data to computations infeasible. Server-local bandwidth can be 8-20x higher than inter-rack bandwidth in these clusters [68]. Hence, *data-locality* is a really important consideration for high performance, and computations are sent to the servers where the data resides [49] instead of sending data to the computations as was done traditionally. An incoming computational job is split into sub-jobs,

and each sub-job is sent to the server hosting the target data in interest of data-locality.

Data-locality brings forth challenging performance and energy trade-offs as illustrated in the example in the Figure 1.1. Respecting data-locality in thermal-aware job placement constrains placement only to the servers that host chunks of the job's target file. Now, these servers may not all be thermally-conducive to the placement of the new job. In the example shown in the Figure 1.1, two of the chunks of the target file reside on the servers with the highest temperature in the cluster. When the sub-jobs follow the chunks, the resultant computational loads further increase the temperatures of the two servers. The example illustrates that non-thermal-aware, data-local job placement may result in an unbounded rise in server temperatures, resulting in thermal-unreliability, and an increase in the cooling energy costs.

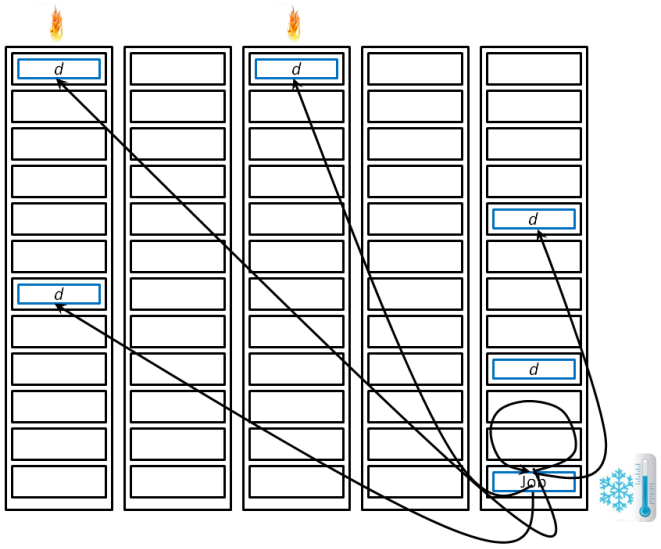

Figure 1.1: **Illustration of data-agnostic thermal-aware job placement limitation in Big Data analytic cloud.**

On the other hand, neglecting data-locality results in cooling energy costs savings at high performance cost. In Figure 1.1, the incoming new job is placed on the coolest run-time temperature server in a data-agnostic manner. Now, not all chunks of the target file of the jobs are server-local and some chunks reside on different servers on the same rack and some on servers in altogether different racks. Since the inter-rack and intra-rack bandwidths are much lower than the server-local bandwidth, such data accesses will suffer from significantly degraded performance and an increase in the overall job completion time. And, the increase in the job completion time will further reduce the cooling energy costs benefit of such a scheme. Thus, there is a need for new

cooling energy management techniques for Big Data analytics cloud that deliver cooling energy savings while allowing data-local high data access performance.

1.2.2 Compute Energy Management

There is significant amount of research on increasing energy-efficiency of the individual components of the server such as the processor [44, 47, 69, 105, 128, 138, 143], storage subsystem [51, 60, 65, 92, 118, 130, 141, 147], memory [17, 18, 50, 71, 89, 94, 95, 103, 131, 146] and networking [25, 83, 96, 109, 124]. However, no single server component (i.e., processor, memory or disks) contributes significantly to the overall power consumption in the commodity cloud servers as discussed in Section 2.1.5 and hence, an energy-management scheme, such as scale-down, that encompasses the entire server is needed.

Scale-down, that involves transitioning server components such as the processor, memory, and disks to an inactive, low-power sleep/standby power state during periods of idle utilization, is an attractive technique to conserve energy. A typical server consumes only 1%-10% of peak power in an inactive sleeping state vs. 40%-70% of peak power in active idle power mode. Given the non-energy-proportional nature of some of the state-of-the-art server components, scale-down is one of the most viable options for yielding energy-proportionality during idle periods. However, scale-down cannot be done naively. Scale-down mandates significant periods of idleness in the server components as energy is expended and transition time penalty is incurred anytime the components are transitioned back to an active power mode.

There is significant amount of research literature on scale-down [29, 87, 107, 108, 126]. A majority of these techniques tries to scale-down servers by manufacturing idleness by migrating and consolidating workloads and their corresponding state to fewer machines during periods of low load. This can be relatively easy to accomplish when using simple data models, when servers are mostly stateless, e.g., serving data that resides on a shared network attached storage (NAS) or storage area network (SAN). When the data resides on a shared NAS/SAN, the data access latency is the same between any server in the cluster and the storage system; hence, workloads can be moved around servers without any performance impact.

However, given the massive bandwidth requirements and the sheer amount of the data that needs to be processed, Big Data analytics clusters have moved away from NAS/SAN model

to completely clustered, commodity storage model that allows direct access path between the storage servers and the clients [61]. The underlying storage system distributes file chunks and replicas across the servers for high performance, load-balancing and resiliency. However, now with files distributed across all servers, any server may be participating in the reading, writing, or computation of a file chunk at any time. Thus, such a data placement complicates scale-down based power-management and makes it hard to generate significant periods of idleness in the Big Data analytics clusters and renders usage of inactive power modes infeasible [91].

Big Data analytics cloud mandates a different compute model which presents a significant challenge to the existing scale-down techniques. The network bandwidth constraints of the commodity network switches and the huge data sizes of Big Data render sending data to computations infeasible. *Data-locality* is a really important consideration for high performance as server-local bandwidth can be 8-20x higher than inter-rack bandwidth in these clusters [68]. To avail high server-local data access bandwidth, computations are sent to the servers where the data is residing [49]. This brings forth challenging performance and energy trade-offs: respecting data-locality makes workload consolidation very constrained or almost impossible; neglecting data-locality results in power savings at high performance cost. Figure 1.2, captures the above-mentioned

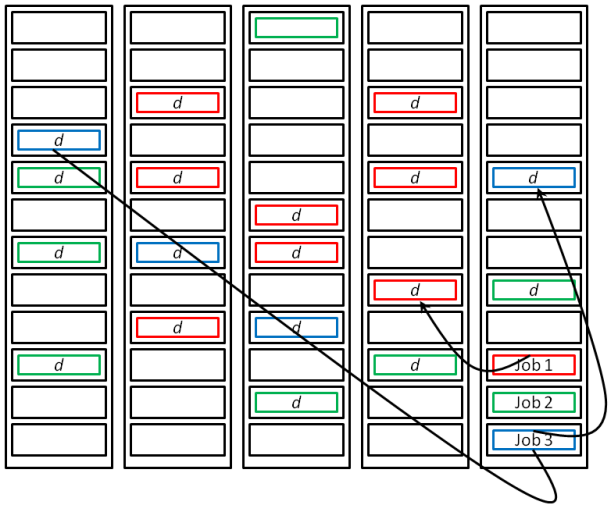

Figure 1.2: **Racks and servers in the cluster. Jobs 1, 2, and 3 are consolidated on few servers while the relevant data (marked in same color) for each job resides on other servers (same or different racks) in the cluster.**

challenges. In the example in the Figure 1.2, there are three incoming jobs during a period of low load. A job consolidation based scale-down technique consolidates three jobs on few servers

during periods of low load in the hope of scaling-down rest of the servers for saving energy costs. However, the rest of the servers actually host the very files chunks these jobs need to process and can not be scaled-down. And, such a data-placement-agnostic job consolidation comes at significant performance impact. The servers on which the jobs have been consolidated, don't host all the file chunks that these jobs need to process resulting in either a rack-local or even an inter-rack data access resulting in a significant performance degradation. Another possibility could be to consolidate both the jobs and the target files on few servers during periods of low load. However, such a technique will still suffer from performance impact as Big Data analytics relies on high parallel file access performance which happens only when the files are chunked and spread across the clusters.

Leverich et. al. [91] and Amur et. al. [23] have taken an energy-aware replica placement approach whereby primary replicas are consolidated on a covering set of servers that are always kept in a powered-on state, and secondary and tertiary replicas are placed on the rest of the servers which are then scaled-down. While being promising, these techniques suffer from degraded read and write data access performance. The covering set of servers containing the primary replicas of the files have all the data read and write accesses directed to them, resulting in an increase in queuing delays, and lower data read and write access performance. In Big Data analytics clusters, file writes happen mostly at the time of new file creation and a write is considered complete only when all the replicas of the file chunks have been written to the servers. Now, in order to avoid waking up scaled-down servers to store the secondary and tertiary replicas, these techniques rely on write off-loading technique, whereby write requests on scaled-down servers are temporarily redirected to persistent storage elsewhere in the data center [108]. The few write-offloading servers further serve as a performance bottleneck during the file writes. Write-performance is an important consideration in Big Data analytics production clusters as significant amount of new data is written daily on the cluster and is computed upon right afterwards. Reduce phase of Map Reduce, the most popular framework for Big Data analytics [49], writes intermediate computation results back to the cluster further mandating high write performance to ensure high overall performance of a Map Reduce task.

Lang and Patel propose an "All-In" strategy (AIS) for scale-down in Map Reduce clusters [85]. AIS uses all servers in the cluster to run a workload and then powers down the entire cluster. However, in production clusters, service level agreements (SLA), completion deadlines, and resource constraints may prevent batching the workloads together. Sharma et. al. [125] have extended upon Meisner et. al.'s work [102] to allow energy savings via blinking (i.e., transitioning

between high-power active state and low-power inactive state). However, both these approaches assume Solid State Drive (SSD) based clusters. Given the low capacities and high costs of the SSD drives, clusters comprising entirely of SSD drives are not feasible at the moment, especially with the petascale storage demands of production Big Data analytics clusters [36]. Disk-based clusters will suffer from significant performance degradation because of the transition penalty associated with frequent state transitions that happen under these schemes. In addition, hardware reliability may also get impacted as disks have limited start/stop cycles and disk longevity goes down with very frequent state transitions required by these schemes. Thus there is a need for new scale-down techniques that work in spite of the unique challenges posed by the Big Data analytics clouds, and yield compute energy costs savings and energy-proportionality without any or reduced performance impact.

1.3 GreenHDFS

GreenHDFS is based on the observation that to support the increase in digital data exhaust, the technology has to fundamentally adapt and change to meet the needs of the Big Data and that data needs to be a first-class object in energy-management in the Big Data environments. To reduce compute (i.e., server operating) energy costs and to provide energy-proportionality, GreenHDFS takes a *data-centric*, cyber-physical scale-down approach which is different from the state-of-the-art job-centric scale-down approaches. Instead of energy-aware consolidation of jobs or replicas as was done earlier, GreenHDFS focuses on energy-aware differentiation and consolidation of *files* to realize sufficient idleness to enable scale-down. On the cyber-side, GreenHDFS is cognizant that all-the-files-are-not-alike in the cluster and have different job arrival rates, sizes, popularity, and evolution life spans. On the physical-side, GreenHDFS is cognizant of uneven thermal-profile of the servers. Instead of treating the files and the servers alike, GreenHDFS differentiates between them on cost, performance, temperature, and power basis, and separates cluster servers and files into logical $Active$ and $Inactive$ zones. Zoning is done in an energy-, thermal-, and performance-aware manner that ensures high read and write performance of the newly arriving and active files in addition to allowing energy costs savings.

Because of compliance, government regulations around file retentions, and disaster recovery requirements, presence of dormant files, i.e., files are just lying dormant in the system without getting computed upon or accessed has increased significantly. A study of a production Big Data

analytics Hadoop cluster at Yahoo! [79] found that 56% of the storage in the cluster was lying dormant (i.e., had not been accessed in the entire one-month long analysis duration). A majority of this dormant data needs to exist for regulatory and historical trend analysis purposes and can not just be deleted [79]. IDC has also pointed out that up to 70% data in the data centers is dormant in nature and a study of Microsoft Big Data analytics cloud found 50% dormant data in their Big Data analytics cluster [64]. GreenHDFS consolidates the dormant files on the $Inactive$ zone, and spreads out the active and newly created files on the $Active$ zone.

Since computations exhibit high data-locality in Big Data analytics frameworks such as Map Reduce [144], energy-aware file placement translates into energy-aware computation placement. The computations flow naturally to the files in the $Active$ zone, resulting in maximal computational load in the $Active$ zone and minimal computational load in the $Inactive$ zone. GreenHDFS results in a number of servers in the $Inactive$ zone with very low utilization and guaranteed periods of idleness, making scale-down feasible.

To reduce cooling energy costs and to ensure thermal-reliability, GreenHDFS takes a different data-centric, cyber-physical cooling energy management approach which is fully data-placement-aware and doesn't involve energy savings and performance trade-offs. GreenHDFS is aware that the cooling and server power consumption depend on the load on the servers, and since jobs follow the files, the load on a server directly depends on the way the files are chunked and distributed across the servers in the cluster. Thus, file distribution affects the load distribution in the cluster, which in turn affects the cluster's energy costs. In GreenHDFS, files are placed first in the cluster in a thermal- and energy-aware manner so that the computational jobs can then automatically enjoy energy costs savings in addition to high data-local performance by following such proactively placed files. Various cyber-physical components such as controller, monitor, and actuator, and policies of GreenHDFS work in tandem to allow scale-down and cooling energy management.

The various challenges faced by the data-centric cooling energy management and scale-down approach in GreenHDFS are discussed next.

1.3.1 GreenHDFS Data-Centric Scale-Down Challenges

Scale-down cannot be done naively. Energy is expended and transition time penalty is incurred when the components are transitioned back to an active power mode. While inactive

sleep/standby power states of disks are very attractive as they consume negligible power compared to active idle mode which still consumes significant power, these inactive modes involve a wake-up latency as high as ten seconds for an inactive-to-active mode transition. A scale-down technique raises following questions to be effective:

- Is there a way to realize sufficient idleness in the servers to ensure that energy savings by scaling-down is higher than the energy spent in the inactive-to-active power state transition?

- Is it possible to ensure that there is no hardware reliability risk as some components (e.g., disks) have limited number of start/stop cycles (e.g., 50,000) and frequent power state transitions may adversely impact the lifetime of the disks?

- Is there a way to ensure that there is no performance degradation because of scaled-down servers? Disks take as high as ten seconds to transition from active-to-inactive power state. Hence, frequent accesses to the data residing on the scaled-down servers will result in frequent power state transitions which can lead to significant performance degradation. Thus, steps need to be taken to reduce the need to access the data residing on the scaled-down servers, and also, to amortize the performance penalty of the unavoidable power state transitions that do need to occur.

- Is there a way to ensure that there is no performance impact of load-unbalancing caused by scaling-down some servers in the data center? Steps need to be taken to ensure that load concentration on the remaining active state servers does not adversely impact overall performance (including data read and write access performance) of the system.

- GreenHDFS data-centric scale-down approach requires consolidation of the dormant files. This raises several additional questions/challenges:

  - How should dormant files get identified?

  - How should truly dormant files get distinguished from files that are simply experiencing a temporary lull in jobs arrival?

  - What are the repercussions of inaccurate dormancy determination?

  - Should the dormancy determination be coarse-grained at data-set level or fine-grained at file-level? What are the advantages and disadvantages of either approaches?

  - Is it possible to determine dormancy onset in the files predictively?
  - Should thresholds be used in the determination; if yes, what is the sensitivity of the system to threshold values?
  - How should the situation when a dormant file is no longer dormant be handled?
  - What support does dormancy determination require from the rest of the system? What is the performance repurcussion of this determination?
  - How often should the dormant files get moved to the $Inactive$ zone?
  - When should the dormant files get moved to the $Inactive$ zone?
- How should the $Inactive$ zone servers be selected to reduce the migration time of the dormant files?
- What trade-offs are involved in the selection of the $Inactive$ zone servers?
- How should the cluster get split into logical $Active$ and $Inactive$ zones in an energy, and performance-aware fashion?

1.3.2 GreenHDFS Data-Centric Cooling Energy Management Challenges

A naive way to do thermal-aware file placement is to place an incoming file $\tilde{f_j}$ on the server i that has the lowest temperature at that time instance. However, this naive file placement can result in higher server temperatures than anticipated, and may impact the cooling energy costs and thermal reliability as illustrated in the example in Figure 1.3. Let there be two servers S_1 and S_2 in the cluster. At file f_3's arrival at $t = 1$, server S_2's temperature $T_{2,t=1}$ is lower than server S_1's temperature $T_{1,t=1}$. The naive placement will choose server S_2 as the destination for file f_3 because of its lower temperature. However, S_2 is a bad choice for f_3. The cumulative load of files f_3 and f_2 (f_2 is already resident on the server S_2), may result in very high computational load at $t = 5$ on server S_2. As a result, S_2's temperature $T_{2,t=5}$ may even exceed the reliability-driven temperature upper-bound T_{max} (which is stated in the data sheet of the servers) at $t = 5$, resulting in an increase in the failure rate of the server.

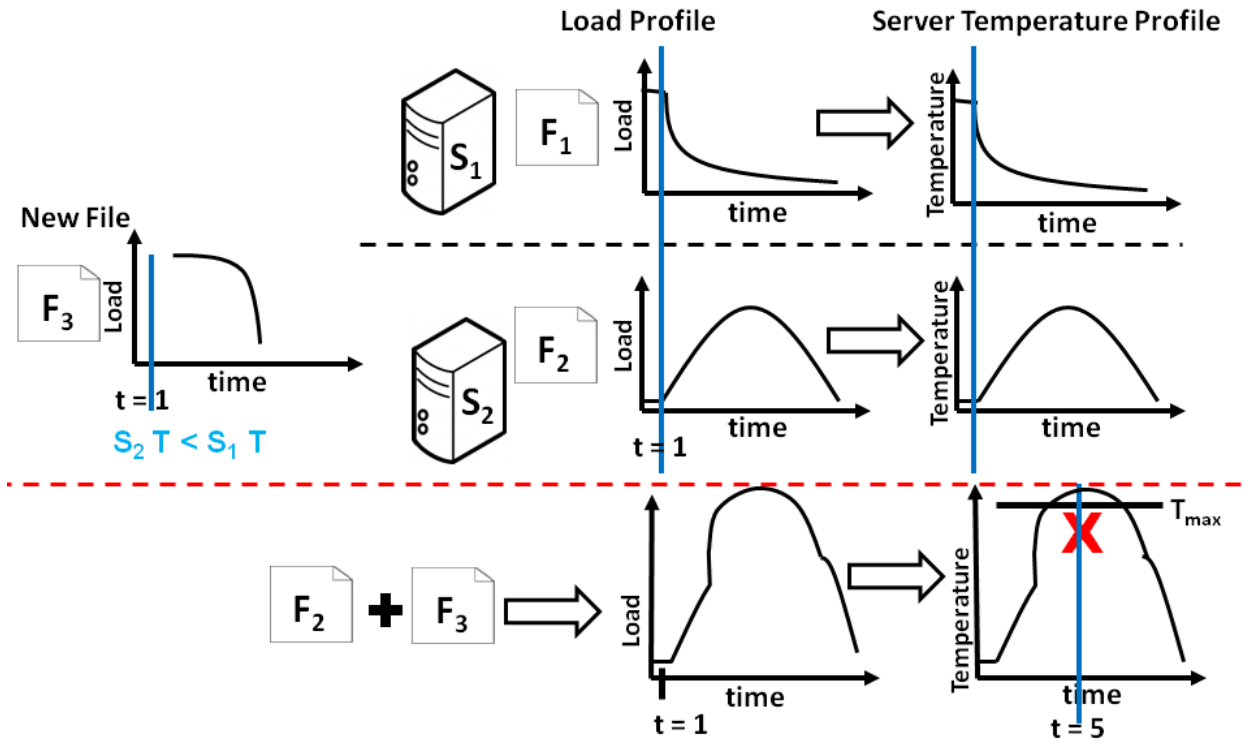

Figure 1.3: **Thermal-aware file placement challenge.**

On the contrary, f_3's expected load-profile is a much fit better fit for S_1's expected thermal-profile. The cumulative load-profile of f_3 and f_1 won't result in S_2 exceeding T_{max} at any point in time as the two load-profiles are complementary in nature. Thus, there is a need for more sophisticated and predictive thermal-aware file placement algorithms and heuristics that can somehow glean information about the "future" computational load profile of an incoming file and the cumulative "future" computational load profile of the servers in the cluster and then place the file "now" on the server that will be most thermally conducive in the "future". Informed thermal-aware file placement is further challenging and raises several *hard* questions:

- Is it even possible to have an optimal solution for thermal-aware file placement or else, is there a need to rely on approximate algorithms?
- How and what aspects of the computation job-profile of the file should be predicted to gain futuristic knowledge to guide the file placement?
- How should the files be placed on servers in a way that ensures that the cumulative com-

putational load of the files results in a thermal- and cooling-efficient temperature-profile of the server?

- There are various varieties of cooling techniques in today's data centers as elaborated in the Background Chapter. Is it possible to have a technique that can ensure thermal reliability, lower thermal-profile, and reduce cooling energy costs both in: 1) new data centers with air- or water-side economization and hot- and cold-aisle air containment, and 2) traditional air-conditioner or water-chiller cooled data centers without air containment or economizer modes and high PUE values?

- How sensitive is the technique to cluster, workload, and data set parameters such as file size skew, delta in inlet thermal-profile, utilization, file creation rate and sampling rate?

- How much run-time information should be incorporated in the file placement decision making process?

1.4 Contributions of the Dissertation

GreenHDFS's main contributions lie in introducing a *data-centric*, cyber-physical thermal and energy management Big Data analytics cloud [78–82]. To the best of our knowledge, this is the first dissertation that takes a data-centric approach to reduce cooling energy costs, ensure thermal-reliability of the servers, and to scale servers down in Big Data analytics cloud while allowing data-local high performance. Evaluation with one-month long real-world traces from a production Big Data analytics cluster (2600 servers, 5 petabytes, 34 million files) at Yahoo! shows that GreenHDFS significantly lowers cooling costs by up to 59%, and compute energy costs by 26%, while giving up to 9x times higher performance than state-of-the-art data-agnostic job placement centric cooling energy management techniques. GreenHDFS results in lower overall cluster temperature, more uniform thermal-profile, less thermal hot-spots, thermal-reliability assurance, and higher cooling-efficiency of the cooling system in both traditional and new data centers. The specific contributions are as follows:

- **Performance- and Energy-Aware Zone Partitioning** Zoning needs to be done in such a way that energy savings can be realized without performance impact. GreenHDFS uses zone partitioning optimization algorithm detailed in Section 4.5.1 to determine the opti-

mal number of servers to be assigned to the $Inactive$ zone subject to performance (i.e., throughput and response time) and capacity constraints. The zone partitioning algorithm ensures that throughput and response time is not impacted by the load-unbalancing caused by zoning.

- **Thermal-Aware Zoning** GreenHDFS is aware that in data centers uneven inlet thermal-profile is ubiquitous because of distance from and varying ability of the cooling system to cool different parts of the data center. In traditional air-cooled data centers, uneven thermal-profile is further aggravated because of hot air recirculation and air bypass. Due to the complex nature of airflow inside data centers, some of the hot air from the outlets of the servers recirculates into the inlets of other servers. The recirculated hot air mixes with the supplied cold air and causes inlets of some of the servers in the data center to experience a rise in inlet temperature. The higher the inlet temperature of a server, the lower is its cooling-efficiency (i.e., ability to remove the heat generated). Lower cooling-efficiency compromises the ability of a server to dissipate the heat generated by the computational load. GreenHDFS is cognizant of the uneven inlet temperatures (i.e., varying cooling-efficiencies) of the servers in the cluster and does thermal-aware server zone partitioning as illustrated in Section 4.5.2 which aids in cooling energy costs reduction in addition to compute energy costs reduction. Dormant files with low computation profile and thereby, low heat dissipation, are deemed a perfect fit for cooling-inefficient servers by GreenHDFS, and are placed on these servers accordingly. On the other hand, active files with high computation profile and high heat dissipation are deemed a perfect fit for cooling-efficient servers.

- **No Performance, Energy Costs Savings Trade-offs** GreenHDFS is fully cognizant of the importance of data-locality in the Big Data analytics cloud. GreenHDFS does energy- and thermal-aware file placement which naturally results in an energy-efficient job placement resulting in cooling energy costs reduction. The computational jobs enjoy server-local, low latency data accesses and hence, the performance is not impacted.

- **High Performance of Active Files** Big Data Analytics is data-intensive in nature and relies on high data access performance. GreenHDFS takes several steps to ensure high performance of the active data as shown below:

  - GreenHDFS takes benefit of zoned bit recording (ZBR) to place active data on the outer hard disk cylinder zones which have higher transfer bandwidth as shown in

Section 4.5.1. Such a placement reduces the service time of the data accesses, and hence, alleviates the load-unbalancing caused by scale-down.

  - All files are not alike and heterogeneity in file sizes and computational job profiles exists in Big Data analytics as well. As shown in Section 4.6.2, small files are present even in Big Data Analytic platforms and many a times suffer in data access performance because of line-blocking behind bigger files. GreenHDFS uses its predictive modeling to do fine-grained and fine-tuned proactive zone selection for an incoming file as shown in Section 4.6.2. Suitable small files are pro-actively placed in servers containing Solid State Drives; such a separation aids in performance. GreenHDFS also does proactive and self-adaptive replication and de-replication for files based on their predicted job-profile and active lifespan as shown in Section 4.6.4.

  - Majority of the servers are assigned to the $Active$ zone and minority to the $Inactive$ zone so that active files can be load-balanced in a fine-grained manner across the servers in the $Active$ zone which is very important for high parallel data access. Since there can be larger amount of dormant files than active files, measures need to be taken to reduce the server footprint of the $Inactive$ zone servers so that more servers can be assigned to the $Active$ zone. GreenHDFS allocates more disks to the $Inactive$ zone servers, and uses reliability mechanism as shown in Section 4.6.4, and space-efficient compression as shown in Section 4.6.5that reduces $Inactive$ zone's storage footprint.

- **Energy-Aware Data Management** Scale-down mandates long periods of idleness, less number of power state transitions, and ways to reduce the need to wake up servers to ensure energy savings, amortization of performance degradation, and hardware reliability. GreenHDFS governs the $Inactive$ zone with a totally different set of policies that are designed in a way to ensure effective scale-down as shown in Section 4.6.3 and 4.6.6. GreenHDFS policies are designed to curtail accesses to the scaled-down servers in order to limit the impact on power state transition penalty on performance.

- **File Dormancy Determination** GreenHDFS is aware that all-files-are-not-alike in the cluster. GreenHDFS relies on a policy called *File Migration Policy* covered in Section 4.6.7 to identify, move, and consolidate dormant files on the $Inactive$ zone. The policy is during periods of low load to ensure no performance impact to the rest of the system.

  GreenHDFS uses two variants of *File Migration Policy* to classify dormant files: reactive

and predictive that offer different energy costs savings, performance, and storage-efficiency trade-offs. Reactive GreenHDFS covered in Section 4.6.7.1 relies on insights gleaned reactively from the observed access patterns of the files to drive its energy-aware file policies. Reactive GreenHDFS doesn't possess any knowledge of the "future" and is coarse-grained in nature. Predictive GreenHDFS covered in Section 4.6.7.2 uses predictive models which it builds from supervised machine learning of historical traces to predict file attributes to guide file placement in a proactive and finely-tuned manner instead of relying on one-size-fits-all file policies allowing higher energy costs savings, storage-efficiency and performance than Reactive GreenHDFS. However, the predictive models are specific for workloads and different feature sets and training methods need to be identified for different workloads. On the other hand, Reactive GreenHDFS can work across multiple workloads.

- **Storage-Efficiency** The *File Migration Policy* covered in Section 4.6.7 not only aids in scale-down, but also results in much more available storage capacity in the $Active$ zone which can be used for better active file layout, and more aggressive replication for obviating performance hot-spots.

- **Energy-Proportionality** GreenHDFS presents opportunity to consolidate workloads on the $Active$ zone servers allowing them to be at a higher, more energy-efficient utilization than the 10%-50%, highly energy-inefficient utilization common in the Big Data analytics clusters. On the other hand, the $Inactive$ zone servers enjoy significant idleness under GreenHDFS and can be effectively scaled-down. Thus, GreenHDFS provides in a software-based mechanism to allow energy-proportionality even with non-energy-proportional server components.

- **Incorporation of Wimpy Nodes** The computational profile of the $Inactive$ zone servers makes them good candidates for low cost, low performance, and low power processors, and less memory. Incorporation of such wimpy nodes further aids in total cost of ownership reduction of the Big Data analytics cluster.

- **Self-Adaptive, Cyber-Physical System** Uneven thermal-profile and thermal hot spots are a ubiquitous issue in data centers because of complex air flow patterns and varying ability of the cooling system to cool different parts of the data center. GreenHDFS combines its data-semantics knowledge on the cyber-side with the thermal-profile knowledge of the cluster on the physical-side to do energy- and thermal-aware file placement. Cyber-physical control loops and components such as controller, monitor, and actuator discussed in Section

4.5 result in a self-adaptive energy-conserving system.

- **Thermal Reliability** GreenHDFS ensures that servers don't exceed the reliability upper-bound of temperature at any time by doing an asymmetric file placement which is cognizant of the differences in the maximum computational load that can be tolerated by a server. The difference in the load tolerance arises mainly from differences in the inlet temperatures of the servers in the cluster as detailed in Section 5.1.6.

- **Predictive Thermal-Aware File Placement** The thermal-aware file placement needs to be done in an informed fashion to realize cooling energy costs savings, and ensure thermal-reliability. GreenHDFS uses predictive models to predict file information which allows it to approximate the new, incoming file's computation profile to guide its thermal-aware file placement. The file placement is done in such a manner at the file creation time itself that it implicitly allows thermal-aware computational load placement in the future. Section 5.3.2.2 details the various predictive thermal-aware data placement heuristics in place in GreenHDFS.

- **Exploration of Various Heuristics** GreenHDFS compares predictive thermal-aware file placement with several other placement options: 1) non-thermal-aware random file placement in which files are randomly placed on the cluster with out any thermal-awareness, 2) thermal-aware non-predictive file placement which aims to place new incoming files either on servers with lowest run-time temperatures or on servers with lowest inlet temperatures, 3) thermal-aware computational job placement, and 4) thermal-aware replica placement to better understand the trade-offs.

- **Sensitivity Analysis** GreenHDFS evaluates the sensitivity of the cooling energy costs savings, thermal-reliability, load-balancing and storage capacity balancing possible with each heuristic on system variables such as monitoring service's sampling rate, job durations, file size skew, file creation rate, and delta in the cooling-efficiencies of the servers in the cluster.

- **Real-World Evaluation and Analysis** To understand the feasibility of the dissertation, significant analysis of the evolution life spans, jobs arrival rate, and sizes of the files is done using month-long real-world file system traces from a large-scale (2600 servers, 5 petabytes, 34 million files) production Big Data Analytics Hadoop cluster at Yahoo!. The analysis focuses on the clickstream dataset as clickstream processing, an example of log processing, is one of the most important use cases of Big Data analytics, has significant

business value and is critical to the Internet economy [54]. A clickstream shows when and where a person came in to a web site, all the pages viewed, and the time spent on each page. Clickstream analysis is used to predict whether a customer is likely to purchase from an e-commerce website [41], to improve customer satisfaction with the website and in effective content delivery (e.g., right bandwidth, resolution, etc.) and to assess the effectiveness of advertising on a web page or site. Internet services companies such as Facebook, Google, Twitter, LinkedIn, and Yahoo! [34] rely on clickstream processing [34, 52, 133] of huge web logs daily to calculate the web-based advertising revenues (source of majority of their income), and to derive user interest models and predictions. Every day 60-90TB uncompressed raw log data is loaded into Hadoop at Facebook [135], 100s of GB log data is loaded at Orbitz [123], and at Europe's largest ad targeting company [72]. Evaluation is performed using traces from the same production cluster to demonstrate the effectiveness and robustness of GreenHDFS's design and algorithms.

1.5 Structure of the Dissertation

The remainder of the dissertation is structured as follows: Chapter 2 provides the models and assumptions made in the dissertation and discusses relevant background information. Chapter 3 discusses related work in energy management. Chapter 4 discusses thermal-aware zoning, its partitioning schemes and algorithms, supporting observations from the real-world Big Data analytics cluster at Yahoo!, energy-aware policies, and evaluation of the Reactive, Predictive and Thermal-Aware policies demonstrating the effectiveness and robustness of GreenHDFS's design and algorithms. A detailed Total Cost of Ownership analysis is also presented for a cluster managed by GreenHDFS and for a cluster in which GreenHDFS is not deployed. Chapter 5 presents details of the motivation, challenges, algorithms and heuristics, predictive modeling, cyber-physical system, and evaluation related to thermal-aware file placement. Lastly, Chapter 6 summarizes GreenHDFS, the lessons learnt, and the future work.

CHAPTER 2

BACKGROUND

In today's digital world, Big Data is everywhere. Social media sites such as Facebook and YouTube store billions of pictures and videos whose total size is in excess of petabytes. E-commerce and Internet services companies collect petabytes of web logs daily to garner business insights from user behavior captured in the web logs. Sensors abound in smart phones, human bodies, vehicles, smart energy meters, servers, and traffic controllers and continuously generate significant amount of sensed data, which is then used for location sensing, targeted advertising, traffic condition monitoring, patient management, elder-care, and power usage and dynamic power pricing. Multimedia entertainment (e.g., high-definition videos and movies), surveillance, scientific simulations, medical records, financial transactions, phone records, genomics, seismology, climatology, geo-spatial mapping and so on are some other sources of very big data sets.

IDC predicts that the total digital data will exceed 35.2 zettabytes by 2020 [7]. And, this data is only expected to grow; the projected growth rate of data is 40% every year [101]. There is a wealth of information hidden in the Big Data that can be leveraged for better business insights, healthcare management, fraud detection, crime prevention and combat, national security, earthquake prevention and so on–the list of use cases is limitless. The data management and analytics software business is expanding significantly and is estimated to be almost $100 billion [54]. The sheer size of Big Data presents a challenge to traditional processing and storage tools and new analysis and storage mechanisms are needed.

2.1 Models and Assumptions

In this chapter, we provide relevant background, and detail the various models used and assumptions made in the dissertation. The Big Data analytics cloud data centers upon which this

dissertation is based have totally different compute, energy and storage models from traditional data centers as discussed next.

2.1.1 Data Center Model

Big Data analytics requires tremendous amount of storage and computing infrastructure. In this environment, scaling-up hardware by buying high-end, high-performance processors, or additional network attached storage boxes becomes exorbitant without giving the desired performance and capacity. Thus, vertical scale-up of the hardware is just not sufficient and horizontal scale-out is required to keep up with the burgeoning data. Cost-efficiency is a very important metric given the sheer scale-out required. Instead of using high-end and expensive components such as network switches with very high bisection bandwidth, infiniband interconnects, hardware RAID (Redundant Array of Inexpensive Drives) controllers, network attached storage (NAS) or large symmetric multiprocessors (SMP) servers, these data centers use low-cost, commodity hardware such as ethernet switches, non-enterprise-grade hard disks, and low-end, 1U or 2U server-class machines typically with dual quad core CPUs, 16-24GB RAM, and 4-12 directly attached serial advanced technology attachment (SATA) disks.

2.1.2 Network Model

The servers in these data centers are arranged in racks and are interconnected using low-end ethernet switches keeping cost-efficiency in consideration [68]. The rack-level switches have up-links connected to cluster-level ethernet switches. Each server in the racks is connected to the rack-switch with a 1-Gbps link. The rack-switch uses eight 1-Gbps links to connect to the cluster-level switch resulting in a over-subscription factor of five. This makes inter-rack data access bandwidth almost 5-10x times worse than the intra-rack bandwidth and 8-20x times worse than disk local access. Any software running on the system needs to be cognizant of the network bandwidth constraints and should make decisions accordingly.

2.1.3 Compute Model

The above-mentioned network model mandates software that is able to utilize the resources such as the network very efficiently by being cognizant of the network bandwidth limitations. Big Data further complicates the situation with its sheer size and movement inertia. It is no longer feasible or scalable to move data to the computations given the huge and ever-growing data set sizes, and network bandwidth constraints. Now, computations need to move to the data and efficient data access from disks is an essential part of the computation. As a result, *data-locality* is a very important consideration for high performance. Map Reduce, a highly scalable, parallel processing framework is widely used for Big Data analytics [49]. Map Reduce owes its high performance to its data-locality feature.

2.1.4 Storage Model

Traditional storage solutions such as Network Attached Storage (NAS) have a client/server model that is just not scalable enough in the Big Data environment. The gateways/filer heads in these systems (NFS, or CIFS) acts as performance and scalability bottleneck during the critical data access path as all the data accesses have to go through them. These systems are also very expensive (e.g. a NAS box may cost $400-$600 per terabyte) as they typically use expensive, high-end hardware such as infiniband interconnects, RAID controllers, and enterprise-grade hard drives. Thus, these solutions become prohibitively expensive when scaled-out to multiple petabytes range as required by the Big Data.

The specialized distributed file systems for data-intensive computing over Big Data such as Google File System (GFS) and the Hadoop Distributed File System (HDFS) follow a totally different storage and file system paradigm from the traditional systems. There are no gateways/filer heads and clients interact with a metadata server (MDS) for metadata operations and communicate directly with storage devices (OSDs) for file I/O (reads and writes). Such an architecture leads to a direct data access path between the clients and the servers hosting the data, resulting in a very scalable and high performance system. And, decoupling of the metadata from the main data aids in the scalability of the system. The direct data access path between the client and the servers hosting the data gives complete visibility on the data layout to the clients–something necessary to make data-local computing possible. The OSDs have their own local file system to

manage the data that has been placed on them resulting in overall system flexibility.

GFS and HDFS are built on top of low-end, inexpensive commodity hardware for economies-of-scale. Fault tolerance is a first class concern given the commodity hardware where failure is a norm rather than an exception. GFS and HDFS rely on file-level replication for resiliency, and fast failure recovery and *distribute* file chunks and their replicas over the *entire cluster*.

2.1.5 Energy Model

In a typical commodity server in a Big Data analytics cluster, no single server component (i.e., processor, memory, or disks) contributes significantly to the overall power consumption (CPU 33%, DRAM 30%, Disks 10%, other components 27% [68]). This energy model is different from the energy model of the high-end, high performance computing servers [35] where the processors are responsible for the highest power draw and hence, techniques such as dynamic voltage and frequency scaling (DVFS) work well in energy conservation. The feature sizes of CPUs are much smaller in today's world and core voltages at the highest frequency are 1.1V to 1.4V. The small feature sizes result in the static leakage power reaching or exceeding dynamic power, and the low core voltages reduce the voltage scaling window. Therefore, the potential to save energy via DVFS is reduced [88]. On the storage side, significant amount of energy management research happened in context of RAID systems. However, the cloud servers don't use expensive hardware RAID controllers in interest of cost-efficiency. Since, disks contribute just 10% to overall power consumption, techniques such as disk spin-down or disk-level energy management are not sufficient and there is a need for server-wide energy management.

2.1.6 File Model

Big Data analytics frameworks such as Map Reduce comprise of a distributed file system underneath, which divides the incoming file into multiple file chunks and then places these chunks into many servers distributed across the cluster as shown in Figure 2.1. Each file chunk is typically 64-128 MB in size [61, 84]. Let f_j^k = chunk k of file j, $1 \leq k \leq \hat{n}_j$, $1 \leq j \leq Z$ where Z is the number of files in the cluster and $\hat{n}_j$ is the number of chunks into which file j is divided. Denote

$\tilde{f}_j = \{f_j^1, f_j^2, \cdots, f_j^{\hat{n}_j}\}$ to be the set of all chunks of file j. $F = \{\tilde{f}_1, \tilde{f}_2, \cdots, \tilde{f}_Z\}$ to be the set of all the file chunks in the cluster. F is itself a set of individual file chunk sets $\tilde{f}_j$.

Let $c_{j,i}^k$ be the chunk k of file j assigned to server i. Let $C_{j,i}$ be a set of file j's chunks assigned to server $i = \{c_{j,i}^1, c_{j,i}^2, \cdots, c_{j,i}^{n_{j,i}}\}$, where $n_{j,i}$ is the number of file f_j's chunks assigned to server i. Let $\tilde{C}_i$ be the set of chunks of different files assigned to server i = a set of the set of file chunks assigned to server i $= \{C_{1,i}, C_{2,i}, \cdots, C_{N_i,i}\}$, where N_i is the number of the files whose chunks are assigned to server i. Let $\ddot{C} = \{\tilde{C}_1, \tilde{C}_2, \cdots, \tilde{C}_{\ddot{N}}\}$, where $\ddot{N}$ = number of servers in the cluster with chunks assigned to them. $\ddot{N}$ is equal to the total number of the servers S in the cluster usually. A mapping function $\pi : F \rightarrow \ddot{C}$ then takes the set of all file chunks F as its input and produces an ordered set $\ddot{C}$ as output. Thus, π assists in mapping the file chunks to the cluster servers.

In a Big Data analytics production clusters, a file is merely a container for the input data and the aggregated output data of the compute jobs. A file is predominantly only accessed by the compute jobs because of strict privacy, security, and business SLA concerns and not by ad hoc users. Each incoming job translates into one sub-job per file chunk f_j^k of the file $\tilde{f}_j$, thus each file chunk f_j^k also has a job arrival rate $\lambda_j(t)$ akin to that of its parent file f_j. Since, each file is computed upon in its entirety, each job results in all the chunks getting accessed.

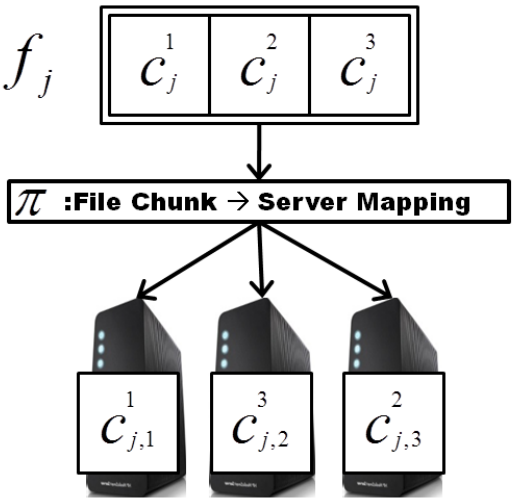

Figure 2.1: **File model in Big Data analytics**

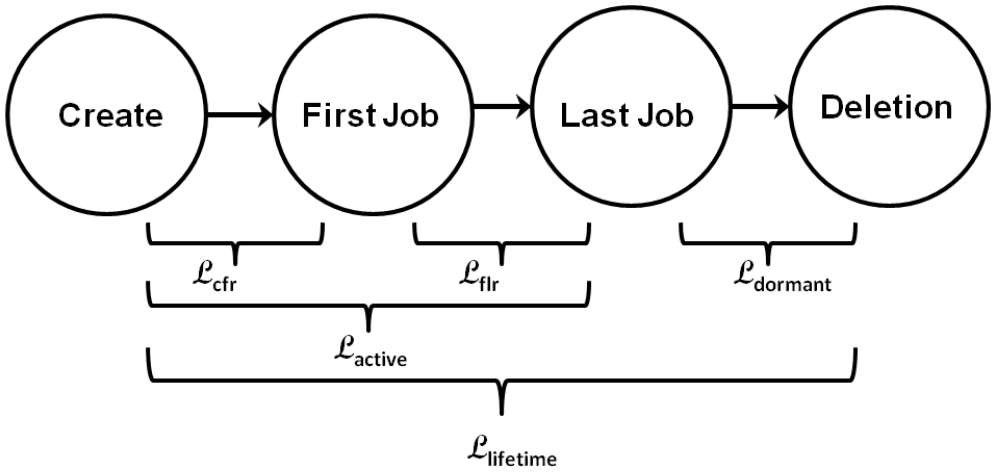

Figure 2.2: **Stages in a file's evolution.**

A file goes to several stages in its lifetime as shown in the Figure 2.2: 1) file creation, 2) *active* period during which the file is frequently being computed upon and hence, being frequently accessed, 3) *dormant* period during which file is not being computed upon and hence, not been accessed, and 4) *deletion*. We introduced and considered various lifespan metrics in our analysis to characterize a file's evolution. A study of the various lifespan distributions helps in deciding the energy-management policy thresholds that need to be in place in GreenHDFS.

- $\mathcal{L}_{cfr,j}$ metric is defined as the file lifespan between the file creation and first computational job directed to the file. This metric is used to find the clustering of the computational jobs around the file creation.

- $\mathcal{L}_{active,j}$ metric is defined as the file lifespan between creation and last computational job received by the file. This metric is used to determine the *active* profile of the files.

- $\mathcal{L}_{dormant,j}$ metric is defined as the file lifespan between last computational job directed to the file and file deletion. This metric helps in determine the *dormancy* profile of the files as this is the period for which files are dormant in the system.

- $\mathcal{L}_{flr,j}$ metric is defined as the file lifespan between first computational job and last compu-

tational job directed to the file. This metric helps in determining another dimension of the *active* profile of the files.

- $\mathcal{L}_{lifetime,j}$. This metric helps in determining the lifetime of the file between its creation and its deletion.

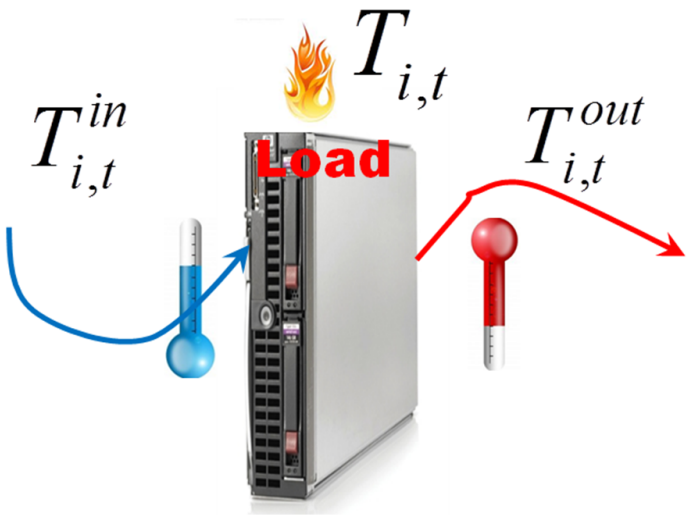

Figure 2.3: **Thermal model of a server**

2.1.7 Thermal Model

A server i in the cloud cluster can be modeled as a heat source (i.e., CPU, Disk, DRAM, Network), where $T_{i,t}^{in}$ is the inlet temperature, $T_{i,t}^{out}$ is the outlet temperature, and $T_{i,t}$ is the temperature of server i at time t as shown in Figure 2.3. The power $P_{i,t}$ consumed by a server i with computational load $L_{i,t}$ at time t can be stated as a linear function of $L_{i,t}$ [67]:

$$P_{i,t} = w_{i_1} \cdot L_{i,t} + w_{i_2}, \tag{2.1}$$

where w_{i_1} and w_{i_2} are coefficients expressed in joules/sec units. Coefficient w_{i_1} is the difference between the peak power consumption and the static power consumption (i.e., when the server is at idle utilization with no computational load), and coefficient w_{i_2} gives the static power

consumption. We assume a homogeneous cluster in this paper, whereby all servers have the exact same hardware. Hence, the values of w_{i_1} and w_{i_2} are the same for each server.

At steady state, the temperature of a server *i*, $T_{i,t}$ is a function of the inlet temperature $T^{in}_{i,t}$ of a server and the power $P_{i,t}$ being consumed at load $L_{i,t}$ [93]:

$$T_{i,t} = T^{in}_{i,t} + c_i \cdot P_{i,t} \tag{2.2}$$

where, constant c_i expressed in kelvin.secs/joules units, is a factor of server's heat exchange rate, it's air flow, and heat capacity density of the air. In a data center with perfect hot- and cold-aisle air containment, whereby there is no air recirculation from the hot aisle to the cold aisle, the differences in the inlet temperature of each server is predominantly dependent on the distance of the server from the vents or the cooling unit. We assume $T^{in}_{i,t}$ to be known empirically for each server. Thus, temperature $T_{i,t}$ can be modeled as a linear function of $L_{i,t}$, i.e.

$$T_{i,t} = T^{in}_{i,t} + c_i \cdot (w_{i_1} \cdot L_{i,t} + w_{i_2}) \tag{2.3}$$

It is important to ensure a server i's temperature $T_{i,t}$ remains less than T_{max} at all times *t*, where T_{max} is the reliability driven upper-bound on the temperature that can be tolerated by a server and is specified in its data sheet; exceeding T_{max} results in higher hardware failure rates.

At steady-state, the temperature of the hot air exhausted from the server i's outlet, $T^{out}_{i,t}$ is a function of the power $P_{i,t}$ being consumed at server i at load $L_{i,t}$, the temperature of the server $T_{i,t}$, and the heat exchange rate of the server i, θ_i expressed in joules/kelvin.secs units as shown below:

$$T^{out}_{i,t} = T_{i,t} - \frac{P_{i,t}}{\theta_i} \tag{2.4}$$

The hot air exhausted by the servers in the cluster enters the cooling subsystem which then cools the hot air to a temperature T_{ac} which is the air supply temperature set point of the cooling subsystem. The cooling power consumption is proportional to the heat removed by the cooling subsystem, which is proportional to the difference in the temperature $T^{out}_{c,t}$ of the hot exhausted air entering the cooling subsystem and the temperature T_{ac}. The cooling power consumption is

given by:

$$P_{c,t} = \frac{c_{air} \cdot f_{ac} \cdot \left(T_{c,t}^{out} - T_{ac} \right)}{COP}, \tag{2.5}$$

where, c_{air} is the heat capacity constant, f_{ac} is the flow rate of the cooling unit, and $|S|$ is the number of servers in the cluster. The efficiency of cooling unit is characterized by its coefficient of performance *(COP)* defined as the ratio of the amount of heat removed by the cooling device and the work required to do so.

2.2 Cooling in Data Centers

The electricity consumed by the servers almost entirely gets converted into heat. Given, the high power densities prevalent in today's data centers, a lot of heat is generated by the computing equipment and efficient removal of the heat is a necessity. The goal of cooling system in the data centers is to maintain the server temperatures within a safe operating range; a temperature outside the safe operating range adversely impacts server's reliability and lifetime. There are various cooling systems available for the data centers today:

- **Water-Cooling** In these systems, the refrigeration cycle happens inside a water chiller. Chiller's function is to produce chilled water which is then pumped through pipes to the computer room air handling unit (CRAH) located inside the data centers. The hot exhausted air from the IT equipment comes in touch with the chilled water circulating in the coils inside the CRAH, and an heat exchange takes places between the chilled water and the hot air. The heat from the hot exhausted air gets transferred to the chilled water and warmed up water is sent back to the chiller to get cooled down. While there are several advantages to chilled water system, it does have few drawbacks: 1) it introduces water inside the data center; any water leakage from the pipes can damage the IT equipment, and 2) the life expectancy of the water-based cooling is lowered by the corrosive affect of water flowing through the pipes.

- **Air-Cooling** The computer room air conditioner(CRAC) use a combination of compressor and condenser to cool the hot exhausted air from the IT equipment. The compressor and the condenser can be in the same unit (higher reliability) or split with compressor being

inside the building and condenser being outside. The refrigeration cycle is responsible for the movement of the heat energy from the inside of the data center to outside. First, hot exhausted air comes in contact with evaporator coils inside the CRAC which have chilled refrigerant flowing through them. The refrigerant absorbs the heat from the air and turns into gaseous state. It is then sent to the compressor which compresses the refrigerant making its temperature rise (a fundamental property of gases). The refrigerant then goes to the condenser which transfers heat from the refrigerant to the outside. The refrigerant is then expanded in the expansion valve which reduces the temperature of the refrigerant and is sent back to the evaporator coil. Air-cooling is the more pervasive in the data centers than water-cooling as it doesn't introduce liquid inside the data centers and hence, doesn't have to deal with leakages.

- **Free-Cooling** In the economizer mode, which is also refered to as free-cooling mode, outside air and/or water from natural sources such as lakes, or oceans are used to cool the data centers either in conjunction with air- and water-cooling systems covered earlier or in stand-alone fashion. The free-cooling can be done in two ways: air-side, or water-side. The air-side economizer uses outside air to cool the data centers while bypassing air conditioners to save cooling energy costs. Fans blow in cold air from outside through data center vents if the ambient temperature of the outside air is lower than certain temperature set points. The outside air is filtered to remove dust and other contaminants as much as possible. If the outside air is not as cold as desired, it is passed through evaporative coils which cools it down. However, this scheme introduces some extra-humidity in the system. In case, the ambient air temperature is much higher than the set point, backup air conditioning unit is used to cool down the data center. There is another variation of free-cooling, whereby, the cold air from outside is not directly blown into the data center. Instead, cold air from outside is used to cool down the exhaust air which is then passed back in. In case of water-side cooling, outside air is used to chill the water when its ambient temperature is conducive bypassing the chiller. Some data centers such as one from Google, use cold water from a lake as the source of chilled water, hence bypassing the water-chiller.

  Economization or free-cooling can significantly save cooling energy costs even though it is not exactly free as pumping, dehumidification, filtration, and fans do consume energy. To enable the usage of free-cooling for longer periods of time and in more locations, the American Society of Heating, Refrigerating and Air-conditioning Engineers (ASHRAE) has increased the allowable temperature ranges of the servers. Free-cooling is sensitive

to outside ambient air temperature and climatic conditions. If the ambient air is not very cold, the inlet temperatures of the servers rise, lowering their cooling-efficiency, and resulting in higher server temperature. Since hardware reliability gets impacted at higher server temperature, in order to make free-cooling viable, attention needs to be given to thermal-reliability and it is important to implement techniques in the cluster to safe-guard servers during periods of high load, and higher ambient temperatures.

While, several companies such as Microsoft, Google, and Yahoo! have announced new data centers that rely only on free-cooling or air- and water-side economization, there are a substantial number of traditional data centers that still rely on air- or liquid-cooling either entirely or in-part coupled with economization. The existing data centers can not always be retrofitted with the economizer modes as incorporation of the economizer requires space for ductwork, and addition of heat exchangers and the required space may not be available in the data center. Free-cooling is sensitive to outside ambient air temperature and climatic conditions; it is easier and more viable to use in locations where ambient air temperatures are low for most part of the year and humidity is in the tolerable range. Hence, based on the location of the older data centers, economization may or may not be an option. And, even though the American Society of Heating, Refrigerating and Air-conditioning Engineers (ASHRAE) has increased the inlet temperature guidelines, the servers need to be designed to be able to support that range before the economization modes can be introduced. And, it just may be too expensive for some of the existing data centers to completely overhaul their existing servers in interest of new servers with higher thermal tolerance. Finally, to be effective in cooling, free-cooling does require hot aisle containment whereby hot exhaust air coming out of the servers is separated out from the input air. Unfortunately, many a times it is hard to retrofit hot-aisle containment in existing data centers. Since a transition between the economizer and a refrigerant mode needs to happen because of unfavorable climate conditions for the economizer, the refrigerant mode transition should happen reliably. The reliability may not be as good in a retrofitted system, and hence, presents an additional concern to retrofitting in existing data centers. Power usage effectiveness (PUE) is the ratio of the total building power to the IT power. Majority of the data centers still have a high PUE of 1.8 as per a 2011 survey of 500 data centers [2]. The high PUE is because the power overheads such as 42% overhead of the CRAC and the chiller. GreenHDFS considers both traditional air-cooled data centers and free-cooled data centers and aims to reduce cooling energy costs and provide thermal-reliability in these data centers.

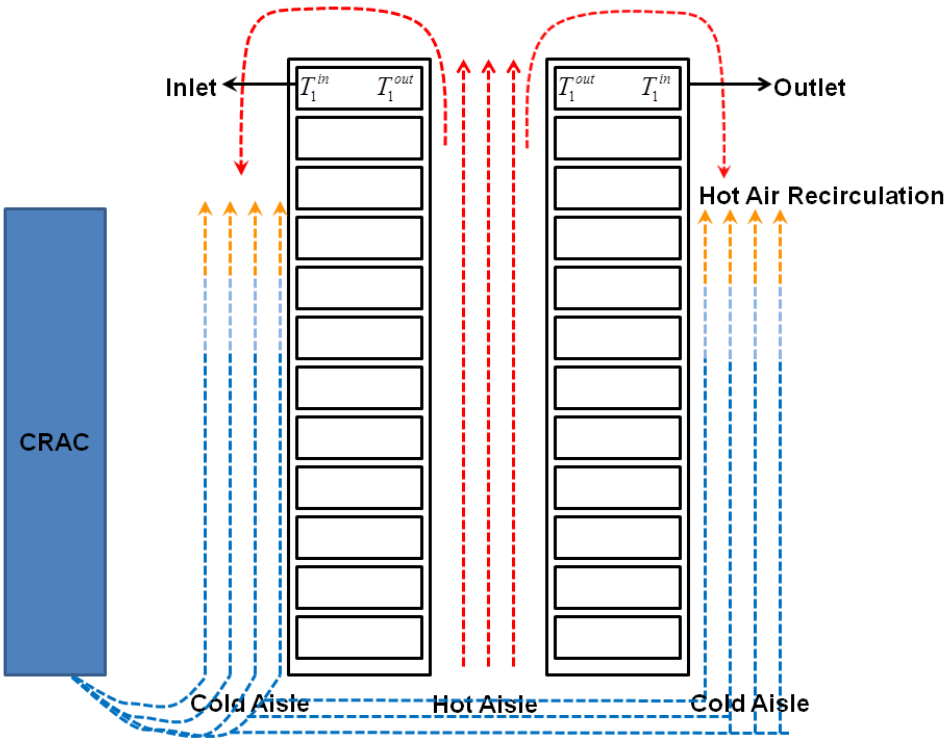

Figure 2.4: **Hot and cold aisle layout and hot air recirculation.**

A typical traditional data center is laid out with a hot-aisle/cold-aisle arrangement by installing the racks and perforated floor tiles in the raised floor [132] as shown in the Figure 2.4. CRAC delivers cold air under the elevated floor. The cool air enters the racks from their front side, picks up heat while flowing through these racks, and exits from the rear of the racks. The heated exit air forms hot aisles behind the racks, and is extracted back to the air conditioner intakes, which, in most cases, are positioned above the hot aisles. Each rack consists of several chassis, and each chassis accommodates several computational devices (servers or networking equipment).

Due to the complex nature of airflow inside data centers, some of the hot air from the outlets of the servers recirculates into the inlets of other servers. The recirculated hot air mixes with the supplied cold air and causes inlets of some of the servers in the data center to experience a rise in inlet temperature. This reduces the cooling-efficiency of these servers and results in an increase in their temperature which results in thermal hot-spots and overall higher temperature in the data center. The thermal hot-spots are the main cause for high cooling energy costs. To offset the recirculation problem, in majority of the real-world data centers, the temperature of the air supplied by the cooling system is set much lower than the red-line temperature, low enough to bring all the inlet temperatures well below the red-line threshold. Unfortunately, operating the

air conditioning units (CRACs) at lower temperature reduces their coefficient of efficiency (COP) and results in higher cooling costs. COP characterizes the efficiency of an air conditioner system; it is defined as the ratio of the amount of heat removed by the cooling device to the energy consumed by the cooling device. Thus, work required to remove heat is inversely proportional to the COP. New data centers have started using hot- or cold-aisle containment, as shown in Figure

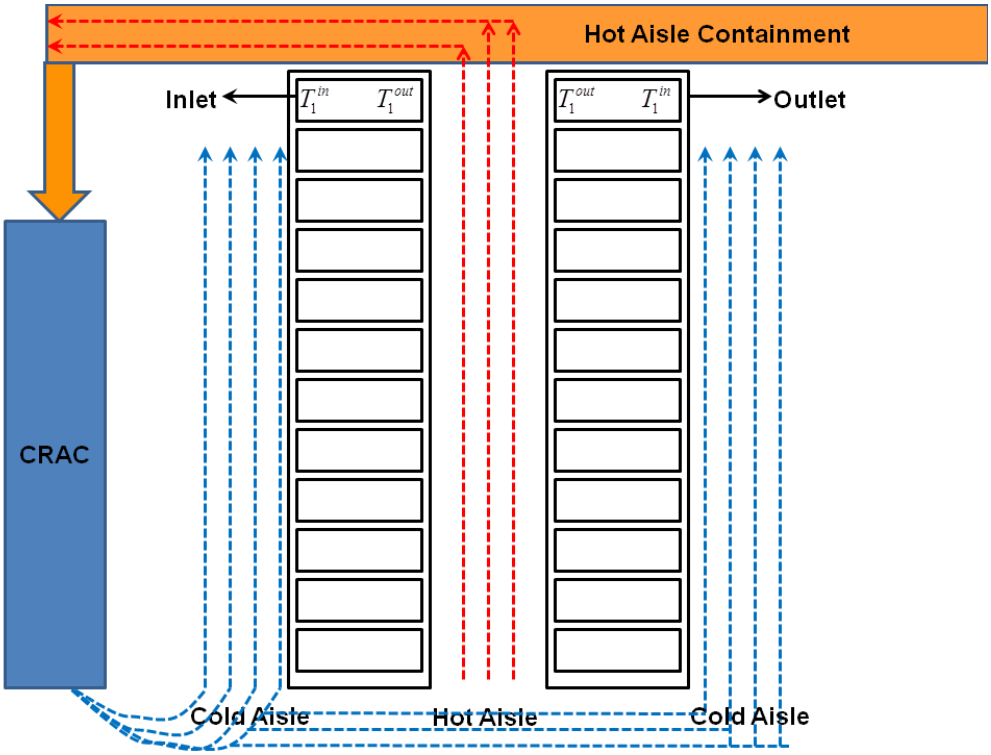

Figure 2.5: **Hot-aisle containment to reduce hot air recirculation.**

2.5, to limit the recirculation of the air and thereby, have increased the efficiency of cooling. The cold inlet and hot outlet streams are separated out. Such containment aids in a reduction of the thermal hot-spots. However, even with air containment, air leakage is possible and hence, it is possible that hot spots won't be removed entirely. While newer data centers are already being designed with containment in mind, there is still a vast majority of pre-containment era data centers which can not be retrofited with containment because of issues like data center layout, and lack of space. Furthermore, CRAC units still vary in their ability to cool different places in a data center (e.g., a corner of the room, farthest from the CRAC), and further aid in uneven thermal-profile [29]. Thus an uneven thermal profile and thermal hot spots are an ubiquitous issue in data centers.

One way to curtail the cooling energy costs is to raise the set point temperature of the air conditioner as that increases the efficiency of the air conditioner and reduces the cooling costs. And, the other way is to use air-side or water-side economization. Since both these techniques help lower the PUE, i.e, increase the data center energy efficiency, American Society of Heating, Refrigerating and in 2011, Air-Conditioning Engineers (ASHRAE) has increased the allowable temperature operating ranges of all the data center classes: A1 $(59^oF - 89.6^oF)$, A2 $(50^oF - 95^oF)$, A3 $(41^oF - 104^oF)$ and A4 $(41^oF - 113^oF)$ just to enable more geographic locations to have more hours in the economizer mode and to allow a raise in the air conditioner's set point temperature. However, higher set point temperature of the air conditioner means that the inlet temperatures at the servers would be higher, resulting in lower cooling-efficiency. Now, servers are rated to operate safely only within specified temperature range and thermal-reliability can not be traded-off for higher energy-efficiency. For example, the operating temperature of Sun Blade is $4^oF - 90^oF$, Dell Blade is $50^oF - 95^oF$, IBM Blade is $50^oF - 95^oF$, and NetAppStorage is $50^oF - 104^oF$. Also, higher temperatures also may mean higher server power consumption because of an increase in the server's fan power consumption and increase in leakage current with increase in temperature. Ensuring thermal-reliability is even more important in the purely air-cooled, chiller-less data centers who are at the mercy of the ambient temperatures and conditions.

GreenHDFS is cognizant of the variations in the thermal-profile of the servers and their varying cooling-efficiencies. It factors in this knowledge of the physical world in its cyber-world data placement decisions. GreenHDFS is not tied to any specific data center cooling model and since it aims to reduce the overall temperature in the cluster, it can be applied modern and traditional data centers. Increasing the inlet supply temperature of the servers increases the efficiency of the cooling unit. GreenHDFS lowers the overall temperature in the cluster, making it feasible to increase the inlet temperature without adversely impacting the reliability.

2.3 Map Reduce and Hadoop Background

Map Reduce is a programming model designed to simplify data processing [49] and is one of the most popular Big Data analytics framework. A Map Reduce application is implemented through two user-supplied primitives: Map and Reduce. Map tasks take input key-value pairs and generate intermediate key-value pairs through certain user-defined computation. The intermediate results are subsequently converted to output key-value pairs in the reduce stage with user-defined

reduction processing. A high-performance distributed file system such as Google File System (GFS) or Hadoop Distributed File System (HDFS), is used to store the input, intermediate, and the output data.

Map Reduce is fully cognizant of the network bandwidth constraints and the movement inertia of the Big Data. Data-locality of the computations is the most important feature of the Map Reduce framework and is responsible for its high performance. A performance test performed at Google showed that Map Reduce is capable of sorting 1TB on 1,000 computers in 68 seconds [62], thereby, illustrating its power. Instead of sending data to the computations, Map Reduce tries to colocate the computation with the data to allow for fast local data access [144]. Thus, every time computation needs to happen on a data, HDFS/GFS determines the nearest server containing a replica of the data and computation is then directed to that server.

Hadoop is an open-source implementation of Map Reduce [26]. It is logically separated into two subsystems: a highly resilient and fault-tolerant Hadoop Distributed File System (HDFS) [84] which is modeled after the Google File System [61], and a Map Reduce task execution framework. Hadoop runs on clusters of low-cost commodity hardware and can scale-out significantly. HDFS is an object-based distributed file system and is designed for storing large files with streaming data access patterns [144]. It aims to provide high throughput of data accesses and hence, it is more suitable for batch-processing applications than for interactive applications which require low latency data access. HDFS supports a single namespace architecture which consists of a hierarchy of files and directories.

HDFS cluster has two types of servers: one NameNode (master) server and a number of DataNode (slaves) servers. NameNode manages the filesystem namespace and metadata. Files and directories are represented on the NameNode by inodes, which contain permissions, modification and access times, user, owner, replication factor, size and absolute file path for these files and directories. Each file in HDFS is split into chunks of typically 64MB - 128MB in size. HDFS distributes the file chunks across the DataNodes for resiliency, high data access throughput, and fine-grained load-balancing. The NameNode maintains the namespace tree and the mapping of the file chunks to the DataNodes. The actual disk placement of the chunk on the DataNode is done by the local file system such as ext3 that is running on the DataNode.

Resiliency and fault tolerance are very important for these Big Data analytics clusters as they comprise of low-end commodity hardware, where failure is a norm. HDFS relies on replication as the means to provide resiliency and each file chunk is replicated n-way for resiliency. The default

replication factor is three in HDFS and at the time of file write, the second replica of a file chunk is written on a server in the same rack as the first replica and the third replica is written on a server in a totally different rack from the previous two replicas.

The file system operations supported by HDFS are "Open", "Create", "Rename", "Delete", "SetReplication", "SetOwner", "SetPermission", and "MkDir". The current implementation of HDFS doesn't support appends and the only writes to a file happen during file creation as part of the Create call. The files follow the write-once ready-many (WORM) pattern. In case of GFS, data can be appended to the files at anytime. The reads to a file happen as part of the Open call.

The clients access HDFS via a HDFS client called DFSClient. At the time of the file read, the DFSClient opens a stream to the file the user wishes to read. It internally contacts the NameNode via a RPC and gets the location of the DataNodes containing the first few file chunk. The client then contacts the DataNodes, reads the chunks and populates the stream. At the time of file writes, the DFSClient internally contacts the NameNode for DataNode mapping and writes chunks to a subset of DataNodes.

During normal operations, DataNodes send heartbeats to the NameNode at fixed interval. The heartbeats allow the NameNode to verify that a DataNode is alive. The DataNodes also send a block report to the NameNode where each block report contains blockid, generation stamp and length of each replica hosted by the DataNode, every 1 hour. The block reports keep the NameNode updated with the latest view of the block replicas on the cluster. Heartbeats also carry information about the total storage capacity, fraction of storage in use and number of data transfers currently in progress. If the NameNode doesn't receive a heartbeat from the DataNode in ten minutes, the NameNode marks the DataNode as dead and assumes that the block replicas hosted by the DataNode are unavailable. The NameNode then rereplicates the replicas on the DataNode marked as dead on other DataNodes.

A single JobTracker process running on the NameNode keeps track of the current jobs status and performs task scheduling. On each DataNode server, a TaskTracker process tracks the available execution slots. A DataNode can execute up to M Map tasks and R Reduce tasks simultaneously (M and R default to 2). A TaskTracker contacts the JobTracker for an assignment when it detects an empty execution slot on the machine.

CHAPTER 3

RELATED WORK

3.1 Existing Cooling Energy Management Techniques

Cooling strategies can be broadly divided into server-level, ensemble-level, and data-center-level strategies. They can be further classified as run-time and static strategies. Server-level strategies include active server fan tuning to cool down the servers [100]. Cohen et. al. propose control strategies via DVFS to enforce constraints on the chip temperature and on the workload execution [77]. At the ensemble level, Niraj et. al. rely on workload migration and location-dependent cooling-efficiency of the fans to manage the power and thermal characteristics of the ensemble. There is significant research on reducing cooling energy costs at the data-center-level [30, 31, 38, 46, 114, 115, 134, 140]. The research on cooling-efficient data center layouts, models and server and rack designs [113, 122, 132] is orthogonal to GreenHDFS.

The run-time strategies mostly rely on *thermal-aware job-placement* to reduce cooling energy costs [16,29,106,107,120,126,134,137]. For example, Moore et. al. [107] provide a mechanism to do temperature-aware workload placement. Sharma et. al. [126] present a framework for thermal load balancing whereby they show how an asymmetric, thermal-aware workload placement and migration can result in uniform temperature distribution in the data center. Bash et. al. [29] attempt to place heavy computational workloads on servers in cooling-inefficient locations in the data center. Sarood et. al. do thermal-aware load balancing [120]. Parolini et. al. and Tang et. al. present a cyber-physical systems approach for data center modeling and control for energy-efficiency which is again relies on thermal-aware job-placement [97, 134].

As we show in the evaluation, thermal-aware job-placement results in significant *performance* impact in the Big Data environment where data-locality is extremely important for low data access latency. Data-locality consideration and significant server state *limit* thermal-aware task

placement and task migration based cooling techniques. Given the explosion in Big Data, data needs to become a first-class object in computing and the computing paradigms need to change accordingly. GreenHDFS takes a data-centric thermal and energy management approach and does proactive, thermal-aware data placement which in turn leads to thermal-aware computation placement.

Recent research on scale-down in MapReduce GFS and HDFS managed clusters seeks to exploit the replication feature of these file systems and proposes energy-aware replica placement techniques for server scale-down [23]. Lang and Patel propose an "All-In" strategy (AIS) for scale-down in MapReduce clusters [85]. These techniques are not thermal-aware and focus only on the computing energy costs savings.

sectionExisting Compute Energy Costs Reduction Techniques There is significant amount of research on increasing energy-efficiency of the individual components of the server such as the processor [44, 47, 69, 105, 128, 138, 143], storage subsystem [51, 60, 65, 92, 118, 130, 141, 147], memory [17, 18, 50, 71, 89, 94, 95, 103, 131, 146] and networking [25, 83, 96, 109, 124]. However, in a typical commodity server in a compute cluster, no single server component (i.e., CPU, DRAM or HDD) contributes significantly to the overall power consumption and hence, an energy-management scheme that encompasses the entire server such as server scale-down is needed.

3.2 Existing Scale-Down Approaches

The scale-down based techniques to reduce the server operating energy costs can be broadly classified into four categories: 1) *Workload migration/placement* based techniques, 2) *Replica placement* techniques, 3) *Workload scheduling* techniques. Techniques 2), and 3) have been proposed specifically for MapReduce based data-intensive compute clusters.

3.2.1 Workload Migration/Placement

One technique to scale-down servers is by manufacturing idleness through migrating workloads and their corresponding state to fewer machines during periods of low activity [42,43,45,107,126,137].

Workload migration is viable when servers are state-less (i.e., serving data that resides on a shared NAS or SAN storage system). Big Data analytics' highly scalable, shared-nothing architecture brings with it unique scale-down challenges. Servers in a Big Data analytics clusters such as Hadoop or other Map Reduce clusters are not state-less. Data-locality is an important feature in Big Data analytics which is responsible for the high performance of data processing in Big Data analytics. Computations are colocated with the data and hence, computation migration is limited to only the servers hosting a replica of the data that needs to be processed. Furthermore, with data distributed across all nodes, any node may be participating in the reading, writing, or computation of a data-block at any time. Such data placement makes it *hard* to generate significant periods of idleness in the Big Data analytics clusters even during low activity periods.

3.2.2 Replica Placement

Big Data Analytics clusters such as Hadoop clusters maintains replicas (default replication factor is three-way) of each data chunk in the system for resiliency, and reliability. Recent research on scale-down compute clusters such as Hadoop clusters [23, 91] seeks to exploit the replication feature of these file systems and proposes energy-aware replica placement techniques. Leverich et. al. [91] propose maintaining a primary replica of the data on a "Covering Subset" (CS) of nodes that are guaranteed to be always on. The rest of the servers can be then scaled-down for energy savings. However, using just the CS servers for all the data accesses may result in degraded data access performance (response time will increase because of the increase in the queuing delays in the disks of the CS servers).

Amur et. al. [23] extend Leverich et. al.'s work by providing ideal power-proportionality in addition by using an "equal-work" data-layout policy, whereby replicas are stored on non-overlapping subsets of nodes. Their proposed file system Rabbit is capable of providing a range of power-performance options. The lowest power, lowest performance option in Rabbit is achieved by keeping just the servers with the primary replicas on. More servers are powered up as performance needs increase.

While promising, these solutions do suffer from degraded write-performance as they rely on write off-loading technique. Write off-loading allows write requests on spun-down disks to be temporarily redirected to persistent storage elsewhere in the data center [108] to avoid server

wakeups at the time of file writes. Write-performance is an important consideration in Hadoop. Reduce phase of a MapReduce task writes intermediate computation results back to the Hadoop cluster and relies on high write performance for overall performance of a MapReduce task.

Furthermore, a study of a production Hadoop cluster at Yahoo! observed that the majority of the data in the cluster has a news-server like access pattern [78]. Production clusters are the truly large-scale compute clusters [36] and are the ones that will benefit the most from the energy-conservation techniques. Predominant number of computations happens on newly created data; thereby mandating good read and write performance of the newly created data. Given, the huge data set sizes, good write performance happens when the incoming data chunks are written in parallel across a large number of DataNodes in the cluster. Write off-loading is just not a performant and a scalable option for such writes. If these techniques do try to wakeup servers to absorb the new writes, they will still suffer performance degradation due to the power state transition penalty. Furthermore, given the significant number of file writes that happen in a day on a production cluster, waking up servers to absorb the writes will also adversely impact the lifetime of the components such as the disks.

3.2.3 Workload Scheduling

Lang and Patel propose an "All-In" strategy (AIS) for scale-down in MapReduce clusters [85]. AIS uses all nodes in the cluster to run a workload and then powers down the entire cluster. The advantages of the AIS technique are that 1) It is a simple approach and does not need any code changes or over provisioning of the storage on a subset of servers, and 2) It offers same data access throughput as the baseline Hadoop and does not need any data layout changes.

This technique makes an underlying assumption that all the workloads happen simultaneously on the system. However, a typical production cluster has several workloads running on the system with varying start and stop times. Given, the globalization rampant today, significant number of clusters are in use 24/7 and hence, such a technique may not see enough idleness in the system to justify a scale-down. The authors also propose batching intermittently arriving jobs and then, submitting all the jobs in the batch simultaneously. Some of the jobs may have Service Level Agreements (SLA) considerations and it may not be acceptable to batch and execute such jobs at a later time. Resource contention may also arise if jobs are all invoked simultaneously on the

cluster. For example, the cluster may not have enough map/reduce compute slots available to be able to service all the jobs simultaneously.

Sharma et. al. [125] have extended upon Meisner et. al.'s work [102] to allow energy savings via blinking (i.e., transitioning between high-power active state and low-power inactive state). However, both these approaches have assumed non-hard disk clusters. Disk-based clusters may suffer from significant performance degradation and impact on disk longetivity with frequent state transitions. Given, the low capacities and high costs of the SSD drives, clusters comprising entirely of SSD drives are not feasible at the moment, especially given the petascale storage demands of a single production compute cluster [36].

3.2.4 Wimpy Nodes

In addition to the scale-down techniques, Vasudevan et. al. [139] and Hamilton [66] have proposed data-intensive clusters built with low power, lower performance processors (Wimpy Nodes) that aim to reduce the peak power consumption of the cluster. While promising, Lang et. al. [86] do point out that for more complex workloads, the low power, lower performance clusters result in a more expensive and lower performance solution.

3.3 Predictive Modeling

Existing highly scalable file systems such as Google file system [61] and HDFS [144] do not do any energy management nor incorporate machine learning based predictive data management cluster policies. To the best of our knowledge, GreenHDFS is the first system to provide a mechanism to predict file attributes for a data-intensive compute framework such as Hadoop and to use the predictions in driving energy and data management policies.

On the predictive data and file management side, some hints-based file systems have been proposed in the past which allow applications to supply hints about the access patterns to the file system to aid in prefetching and selective caching of data [63, 111]. However, these systems either require application or compiler-level changes. Other predictive techniques in file system

involve run-time, computationally intensive analysis and may not scale in the current trend of petascale storage systems [21, 99]. For example, Akyürek et. al. require in-memory counters per disk block [21].

Ellard et. al, propose a file classification scheme in which they use a combination of several attributes such as file name, owner, group identifiers, permissions and creation time as the parameters in their classifier to predict file access patterns [55]. The dataset used by Ellard et. al. is based on Network File System traces. Big Data analytics storage model discussed in 2.1.4, has moved away from NFS based NAS systems because of the inherent performance bottlenecks present in the client/server model of NFS. Hence, there is a need to study Big Data analytics data sets to figure the feature sets that would make sense for that data set.

Since, not all file attributes are relevant and using too many attributes in the feature set can result in over-fitting issues, we exhaustively examined real-world traces from a large-scale production Big Data analytics Hadoop cluster at Yahoo! and realized that the directory hierarchy of a file in itself is a strong predictor for the clickstream data set (dataset with the most business value). Ellard et. al., aim to create classes of data and hence, use a classification technique. On the other hand, Predictive GreenHDFS needs to predict numeric values of the various file attributes and hence, requires a regression technique.

On the predictive energy management side, Essary and Amer used predictive data grouping to reduce the energy consumption of hard disks [58]. However, in a typical commodity server in a compute cluster, no single server component (i.e., CPU, DRAM or HDD) contributes significantly to the overall power consumption and hence, an energy-management scheme that encompasses the entire system is needed. Furthermore, Energy-proportionality is increasingly becoming an important consideration [59]. In an energy-proportional system, almost no power is consumed when the system is idle and power consumption increases in proportion to the increase in the activity level. Predictive GreenHDFS relies on server scale-down to conserve energy. Scale-down, which involves transitioning server components such as the CPU, DRAM, and disks to an inactive, low power-consuming sleep/standby state during periods of idle utilization, is an attractive technique to conserve energy. Given the non-energy-proportional nature of some of the state-of-the-art server components such as hard disks, scale-down is one of the most *viable* options for yielding energy-proportionality during idle periods.

3.4 File and Storage Systems

Majority of the existing file systems treat all the files in the system alike [61, 84, 98, 142] and do not differentiate between the files. TierFS file system does differentiate between files and uses multiple storage tiers to enhance the recoverability of the system [20]. However, TierFS systems doesn't do any thermal- or energy-management. GreenHDFS, on the other hand, is built on the observation that all-file-are-not-alike in the cluster, and files differ in their computational jobs arrival rate, popularity, size, and evolution life spans. GreenHDFS then leverages the knowledge of the differences in the files to guide its energy conservation policies.

All the existing file systems are purely at the cyber-side and have no awareness of the physical-side of the cluster. GreenHDFS, on the other hand, is a cyber-physical file system and is fully aware of the thermal-profile of the servers in the cluster. GreenHDFS leverages its knowledge of the unevenness in the temperatures of the servers and the differences in the files in its thermal-aware file placement policies. The thermal-aware file placement allows GreenHDFS to reduce cooling energy costs and ensure thermal-reliability of the servers in the cluster. None of the other file systems or storage systems look at thermal or cooling energy management.

Gunda et. al. [64], do data-differentiation between the data sets in the data center. They opt for an approach whereby they garbage-collect dormant datasets. However, a significant amount of dormant data is deliberately retained in the system for various business reasons such as regulatory compliance, historical trend analysis and business continuity and can not just be deleted prior to the end of its retention period. AutoRaid [145] also does differentiate between the data; however, only in the context of a single disk-array controller whereby one level uses mirroring and the other RAID-5.

Existing multi-tiered storage systems leverage different tiers to make trade-offs between cost, reliability and performance. GreenHDFS has introduced an additional dimension of *power* and *temperature* to the set of the trade-offs. Furthermore, the zones in GreenHDFS are just a logical abstraction at the file system level unlike physically distinct storage tiers in the hierarchical storage systems. Movement of the files to different zones and metadata management of the files is much easier in case of GreenHDFS compared to the hierarchical storage management systems which typically rely on file stubs or reparse points to do metadata management as they are not integrated with the file system.

CHAPTER 4

DATA-CENTRIC COMPUTE ENERGY MANAGEMENT

4.1 Motivation for Scale-Down

There is significant amount of research on increasing energy-efficiency of individual components of servers such as the processor [44, 47, 69, 88, 105, 127, 128, 138], the storage subsystem [51, 60, 65, 92, 118, 130, 141, 147], the memory system [17, 18, 50, 71, 89, 94, 95, 103, 131, 146] and the networking equipment [25, 83, 96, 109, 124]. However, in a typical commodity server in a Big Data analytics cluster, no single server component contributes significantly to the overall power consumption and hence, an energy-management scheme that encompasses the entire server is needed.

Energy-proportionality is increasingly becoming an important consideration [28]. In an ideal, perfectly energy-proportional system, almost no power is consumed when the system is idle and power consumption increases in proportion to the increase in the activity level. In reality, instead of consuming negligible power, the server components consume almost 40%-70% [28] of the peak power during idle utilization. The main culprits are memory, hard disk drives, and networking system as their dynamic range (i.e., the range between the power draw at peak utilization vs. idle utilization) is much lower than that of the processor. In fact, processors that allow voltage and frequency scaling are the most energy-proportional server components. The dynamic range is approximately 2.0x for memory (DRAM DIMM consumes 3.5-5W at peak utilization and 1.8-2.5W at idle utilization), 1.2x for disks (Seagate Barracuda ES.2 1TB consumes 11.16W at peak utilization and 9.29W at idle utilization), and less than 1.2x for networking switches. This suggests that energy-proportionality cannot be achieved through processor optimizations such as dynamic voltage and frequency scaling (DVFS) alone, and requires improvements across all server components.

Some non-energy-proportional components such as the disks require greater innovation to be energy-proportional. Disk drives consume significant amount of power simply to keep the platters spinning, possibly as much as 70% of their total power for high RPM drives [65]. Energy-proportionality incorporation in disks may require smaller rotational speeds, or smaller platters. Since, the hardware-level innovaton may take significant time, it is important to explore software-driven energy-proportionality mechanisms.

Scale-down, that involves transitioning server components such as the processor, memory, and disks to an inactive, low power-consuming sleep/standby state during periods of idle utilization, is an attractive technique to conserve energy. Given the non-energy-proportional nature of some of the state-of-the-art server components, scale-down is also one of the most *viable* options for yielding energy-proportionality during idle periods as inactive, sleep power states consume negligible power compared to idle-mode power states. A typical server consumes only 1%-3% of peak power in an inactive sleeping power state vs. 40%-70% of peak power in active idle power state. This behavior can be explained by examining the behavior of components such as disks. While the disk arms don't need to move during active idle-mode power state conserving some power, the disk does keeps its platters spinning which can consume as high as 70% of the peak power. On the other hand, in the inactive power states such as standby/sleep, the heads are parked and the spindle is at rest; thereby, consuming negligible power. Thus, scaling-down a server during idle utilization is a much better option for energy conservation and energy-proportionality than using active idle-mode power states.

4.2 Scale-Down Challenges

Scale-down cannot be done naively. Energy is expended and transition time penalty is incurred when the components are transitioned back to an active power state. While inactive sleep/standby power states of disks are very attractive as they consume negligible power compared to active idle-mode power state which still consumes significant power, these inactive power states involve a wake-up latency as high as ten seconds for an inactive-to-active power state transition. Scale-down can be justified only if the idleness interval meets the criterion illustrated in the following equations:

The power consumed by a server i with computational load $L_{i,t}$ at time t can be stated as a

linear function of $L_{i,t}$ [67]:

$$P_{i,t} = w_{i_1} L_{i,t} + w_{i_2}, \tag{4.1}$$

Coefficient w_{i_1} is the difference between the maximum power draw (at 100% utilization) and the static power, and coefficient w_{i_2} gives the static power consumption when the machine is at idle utilization with no computational load [67].

In any idle period t_{idle}, whereby $L_{i,t}$ is zero, there are two power state options for the servers: 1) scale-down, i.e., transition to low-power consuming inactive power state, 2) remain in active power state.

$$E_{sleep} = P_{sleep} \cdot t_{idle} + E_{wake} \tag{4.2}$$

$$E_{nosleep} = (w_{i_2}) \cdot t_{idle} \tag{4.3}$$

The energy expenditure for option 1) is given by Equation 4.2 where, P_{sleep} is the power draw of the inactive power state, E_{wake} is the energy expended upon a server wakeup, and E_{sleep} is the total energy consumed by scaling-down the server for the t_{idle} duration and then, transitioning the server back to active power state at the end of t_{idle} interval. The energy expenditure for option 2) is given by Equation 4.3 where, $E_{nosleep}$ is the energy consumed if the server just stays in the active power mode for the entire t_{idle} duration.

To ensure energy savings, scale-down should be done during an idle period t_{idle}, only if $E_{nosleep} - E_{sleep} >> 0$. This happens when t_{idle} meets the time duration requirements illustrated below:

$$t_{idle} > \frac{E_{wake}}{w_{i_2} - P_{sleep}} \tag{4.4}$$

An effective scale-down technique mandates the following:

- *Sufficient idleness* to ensure that energy savings are higher than the energy spent in the transition as shown in Equation 4.4.

- *Less number of power state transitions* as some components (e.g., disks) have limited number of start/stop cycles (e.g., 50,000) and too frequent transitions may adversely impact the lifetime of the disks.

- *No performance degradation*. Disks take significantly longer time to transition from active to inactive power mode (as high as 10 seconds). Frequent power state transitions will lead to significant performance degradation. Hence, steps need to be taken to reduce the power state transitions and also, to amortize the performance penalty of the unavoidable power state transitions that do need to occur.

- *No performance impact of load-unbalancing*. Steps need to be taken to ensure that load concentration on the remaining active state servers does not adversely impact overall performance (including data read and write access performance) of the system.

4.3 Scale-Down Challenges in Big Data Analytics Cluster

There is significant amount of research literature on scale-down [29, 87, 107, 108, 126]. A majority of these techniques try to scale-down servers by manufacturing idleness by migrating and consolidating workloads and their corresponding state to fewer machines during periods of low activity. This can be relatively easy to accomplish when using simple data models, when servers are mostly stateless (e.g., serving data that resides on a shared NAS or SAN storage system). When the data resides on a shared networked attached storage (NAS) such as Netapp NAS box or storage area network (SAN), the data access latency is the same between any server in the cluster and the storage system; hence, workloads can be moved around servers without any performance implications.

However, given the massive bandwidth requirements and the sheer amount of the data that needs to be processed, Big Data analytics clusters have moved away from NAS/SAN model to completely clustered, commodity storage model that allows direct access path between the storage servers and the clients [61]. The underlying storage system distributes file chunks and replicas across the servers for high performance, fine-grained load-balancing and resiliency. With files distributed across all servers, any server may be participating in the reading, writing, or computation of a file chunk at any time. Such a data placement *complicates* scale-down based

power-management and makes it hard to generate significant periods of idleness in the Big Data analytics clusters and renders usage of inactive power modes infeasible [91].

Furthermore, Big Data analytics cloud mandates a different compute model which presents a significant challenge to the existing scale-down techniques. The network bandwidth constraints of the commodity network switches and the huge data sizes of Big Data, render sending data to computations infeasible. *Data-locality* is a really important consideration for high performance as server-local bandwidth can be 8-20x higher than inter-rack bandwidth in these clusters [68]. To avail high server-local data access bandwidth, computations are sent to the servers where the data resides [49]. This brings forth *challenging* performance and energy trade-offs: respecting data-locality makes workload consolidation very constrained or almost impossible; neglecting data-locality results in power savings at high performance cost.

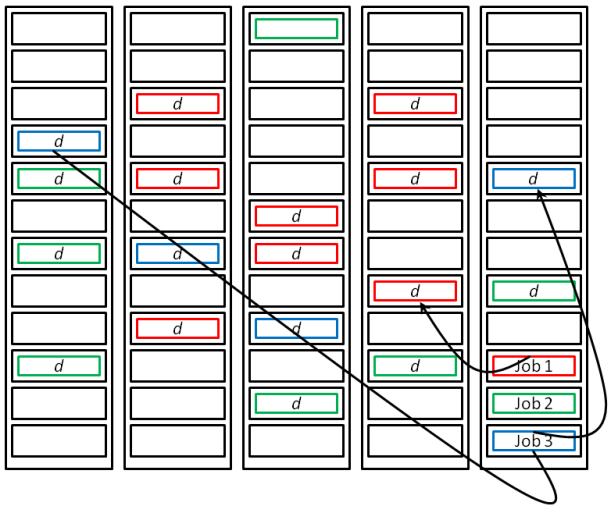

Figure 4.1: **Racks and servers in the cluster. Jobs 1, 2, and 3 are consolidated on few servers while the relevant data (marked in same color) for each job resides on other servers (same or different racks) in the cluster.**

Figure 4.1, captures the above-mentioned challenges. In the example in the Figure 4.1, let there be three incoming Big Data analytics jobs during period of low load. A job consolidation based scale-down technique consolidates the three jobs on few servers during periods of low load in the hope of scaling-down rest of the servers for saving energy costs. However, the rest of the servers are actually hosting the very files chunks which these jobs need to process. And, in the

worst-case, if the target files of the jobs to be consolidated are really huge, they may even span the entire cluster, rendering scale-down of the remaining servers infeasible. Furthermore, such a data-placement-agnostic job consolidation comes at significant performance impact. The servers on which the jobs have been consolidated, may or may not host the file chunks that these jobs need to process. This results in either a rack-local or even an inter-rack data access. Inter-rack bandwidth can be 8-20x times lower than server-local bandwidth in the Big Data analytics clusters, thereby, resulting in a significant performance degradation.

Recent research on increasing energy-efficiency in GFS and HDFS managed clusters [23, 91] propose maintaining a primary replica of the data on a small covering subset of nodes that are guaranteed to be on and which represent the lowest power setting. The remaining replicas are stored in larger set of secondary nodes which are scaled-down to save energy. In a Big Data analytics storage model, a file chunk is considered written once all the replicas of the file chunk have been written to the respective servers. While the primary replica of the chunk can be written to the covering set of servers without requiring any server wake-ups, the secondary and tertiary replicas (three-way replication is the norm) may require server wake-ups of the secondary servers on which these replicas need to be placed. Since, wake-ups have a significant transition latency, these secondary and tertiary replica writes suffer from degraded performance. Other option is to offload the secondary and tertiary replica writes to dedicated servers using write-offloading technique. However, write-offloading technique comes at performance impact caused by queuing delays on the few write-offloading servers. The reads and writes of the primary replica of files also suffer from degraded performance because of the increase in queuing delays on the covering set of servers. In production Big Data analytics, thousands of files are created and written to even at an hour granularity. Since, covering replica set solution suffers from degraded write-performance and increased code complexity [24], it is not a good fit for the production clusters. Thus, there is a need for a *new* scale-down based energy management technique for Big Data analytics cloud that delivers energy savings while allowing data-local high performance.

4.4 GreenHDFS

GreenHDFS takes a *data-centric*, cyber-physical scale-down approach which different from the state-of-the-art *job-centric* scale-down approaches. GreenHDFS is based on the observation that data needs to be a first-class object in energy-management in the Big Data environments.

Instead of energy-aware consolidation of jobs or replicas as was done earlier, GreenHDFS focuses on energy-aware consolidation of *files* to realize sufficient idleness to enable scale-down. On the cyber-side, GreenHDFS is cognizant that not all-files-are-created-alike and files differ in their job arrival rates, sizes, popularity, and evolution life spans. On the physical-side, GreenHDFS is aware that even in a data center with hot- or cold-aisle containment or free-cooling, uneven server inlet thermal-profile exists because of distance from the cooling system or the vents, hot air leakage, and varying ability of the cooling system to cool different parts of the data center. In traditional air-cooled data centers without air containment (hot- and cold-aisle containment can not always be retrofitted in existing data centers because of space and other architectural limitations), uneven inlet thermal-profile is further aggravated because of hot air recirculation and air bypass; due to the complex nature of airflow inside data centers, some of the hot air from the outlets of the servers recirculates into the inlets of other servers. The recirculated hot air mixes with the supplied cold air and causes inlets of some of the servers in the data center to experience a rise in inlet temperature. The higher the inlet temperature of a server, the lower is its cooling-efficiency (i.e., ability to remove the heat generated). Lower cooling-efficiency compromises the ability of a server to dissipate the heat generated by the computational load.

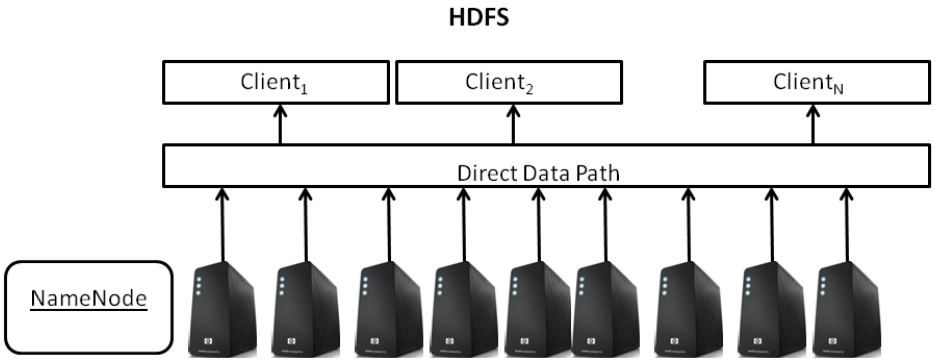

Figure 4.2: **All the files and servers in the cluster are treated alike in baseline Big Data analytics cluster with no GreenHDFS deployment.**

Files in the cluster can be in different evolution phases: some files may be in their active phase; whereby, the files are actively being computed upon and accessed, and other files may be in their dormant phase; whereby, the files are past their active phase and are now lying dormant in

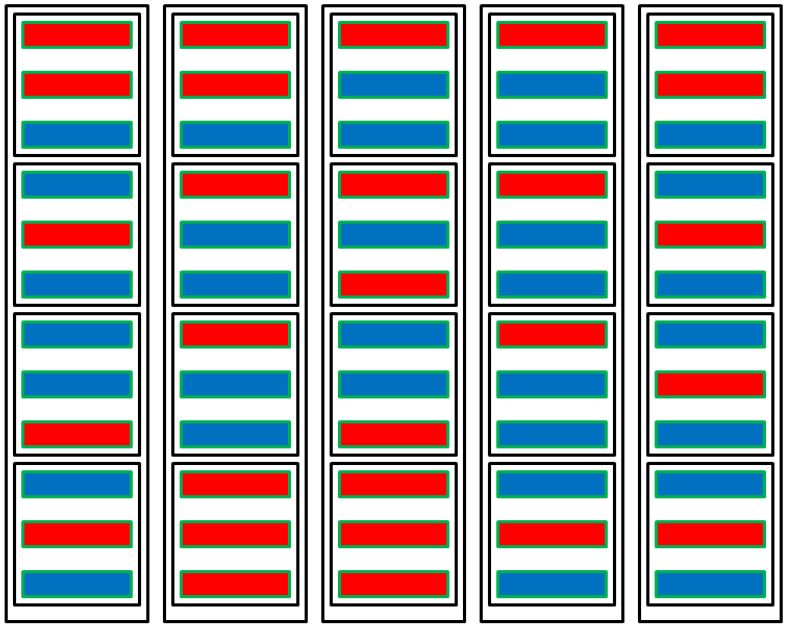

Figure 4.3: **Active and dormant file chunks are inter-mixed across servers. Since all servers contain some share of active file chunks, servers are rarely idle and any server can be reading, writing, or computing at any time.**

the system without receiving any computations or accesses. State-of-the-art Big Data analytics clusters such as Hadoop clusters treat the servers and the files in the cluster alike and do not differentiate between them in their policies and decision making process as shown in the Figure 4.2. The active and dormant file chunks are intermixed on the servers in the cluster as shown in the Figure 4.3. In this storage model, active file chunks may reside on any server; as a result, servers are rarely idle as any server can be reading, writing, or computing at at time, rendering scale-down infeasible.

File- and Server-Differentiation: On the other hand, GreenHDFS is cognizant of the file and server differences and does differentiate between both the servers and files in its policies and decisions. GreenHDFS trades cost, performance, temperature, and power by separating cluster servers and files into logical $Active$ and $Inactive$ zones as shown in Figure 4.4. $Active$ zone servers are used to host files that are actively being computed upon and the new incoming files, and the $Inactive$ zone servers are used to host dormant files, i.e., files that are no longer in their active life span, but, are still being retained in the system for regulatory, compliance,

disaster recovery, or historical trend analysis reasons. These files have none or very infrequent computations targeted to them. Our study of a production Big Data analytics Hadoop cluster at

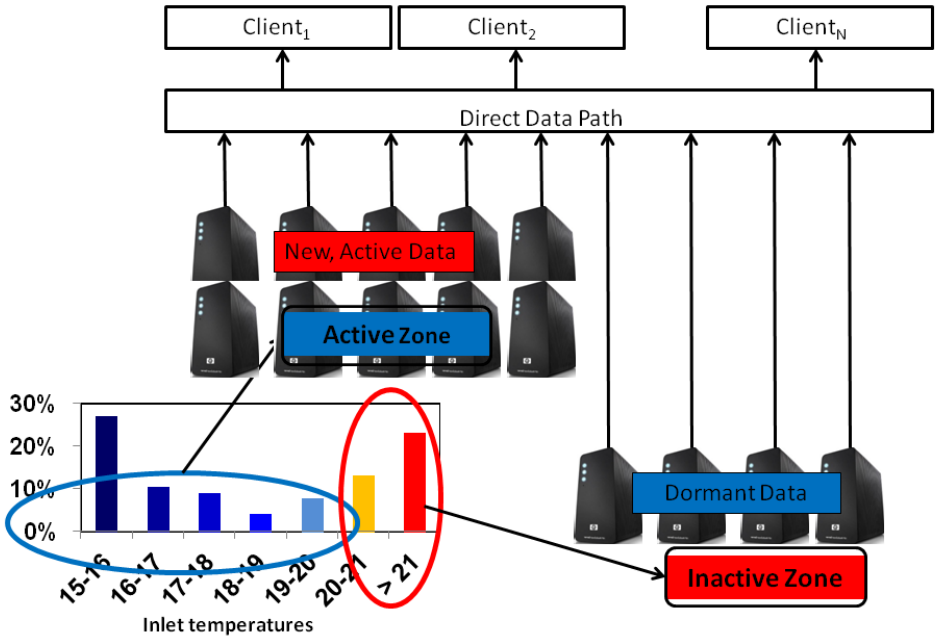

Figure 4.4: **GreenHDFS is aware that files and servers in the cluster are not-all-alike and separates them into thermally- and data-differentiated** $Active$ **and** $Inactive$ **zones.**

Yahoo! [79] found that 56% of the storage in the cluster was dormant (i.e., was not accessed in the entire one-month long analysis duration) as shown in Figure 4.5. A majority of this dormant data needed to *exist* for regulatory and historical trend analysis purposes and could not just be deleted [79]. IDC has also pointed out that up to 70% data in the data centers is dormant in nature and a study at Microsoft also found 50% dormant data in their Big Data analytics cluster [64]. There has been an increase in the dormant data in the production clusters due to several government regulations around data retention, compliance regulations, and disaster recovery concerns.

Since, computations exhibit high data-locality in Big Data analytics frameworks such as Map Reduce [144], energy-aware file placement translates into energy-aware computation placement. The computations flow naturally to the files in the $Active$ zone, resulting in maximal computational load in the $Active$ zone and minimal computational load in the $Inactive$ zone. GreenHDFS

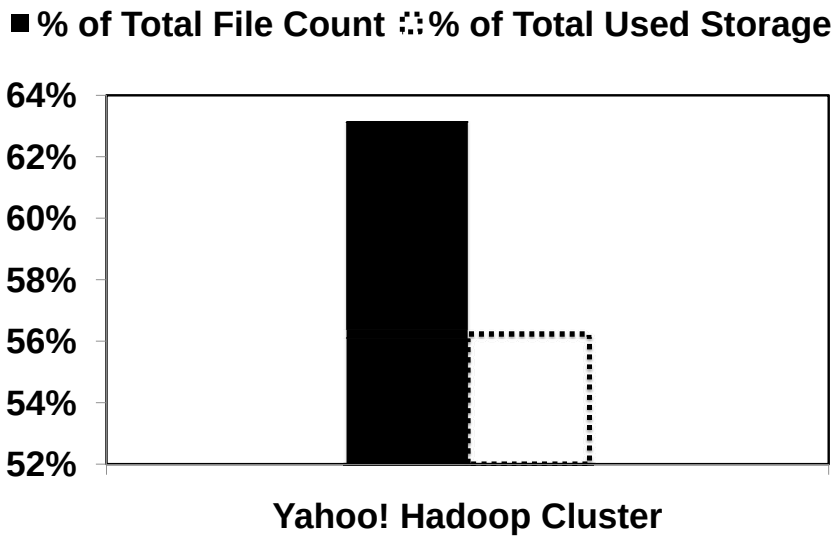

Figure 4.5: **56% of used storage capacity is dormant in the Yahoo! production hadoop cluster.**

data-differentiation-driven, dormant files consolidation storage model shown in Figure 4.6 results in a number of servers in the $Inactive$ zone with very *low* utilization and guaranteed periods of idleness, making scale-down feasible as scale-down mandates sufficient idleness to amortize the energy expenditure and performance penalty of power state transitions. The CPU, DRAM and disks on these servers can then be transitioned to inactive power states resulting in substantial energy savings.

Thermal-Aware Zone Partitioning: GreenHDFS does zoning in a thermal-aware manner and assigns the most inherently cooling-inefficient servers in the cluster to the $Inactive$ zone as $Inactive$ zone hosts dormant files whose low or negligible computational load-profile is a great fit for the cooling-inefficient servers given their impaired heat dissipation capability. Such a placement ensures that the cooling-inefficient servers receive negligible computations; as a result, the cooling-inefficient servers don't generate much heat, and their exhaust temperature remains bounded. Thus, a thermal-aware server zone partitioning reduces the thermal hot-spots in the cluster leading to an overall lower temperature in the cluster which in turn reduces the cooling energy costs.

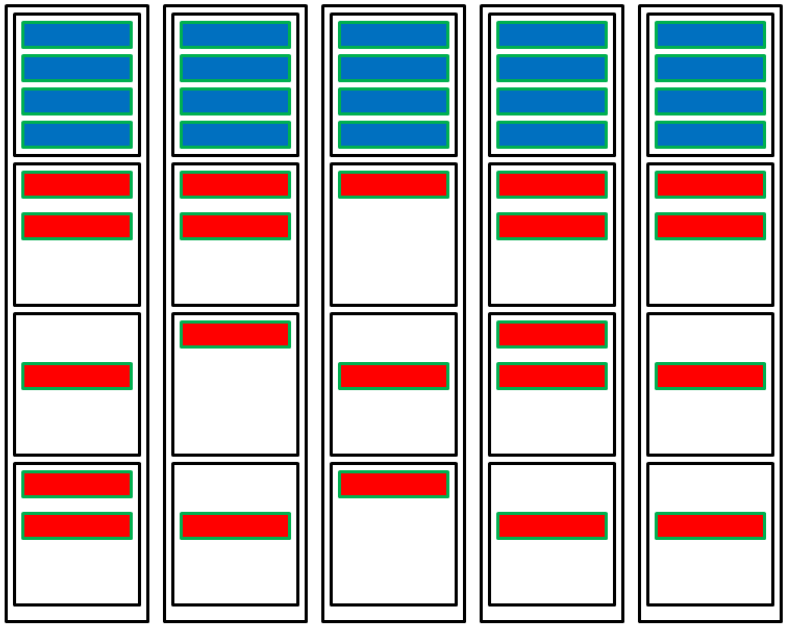

Figure 4.6: **Active and dormant file chunks are separated. Dormant files are consolidated on few servers allocated to the $Inactive$ zone and active file chunks are spread out across the rest of the servers. In this split storage model, the servers in the $Inactive$ zone experience significant periods of idleness.**

Data-Differentiated Per-Zone Policies: GreenHDFS governs the $Active$ and $Inactive$ zones with policies most conducive for the class of data residing in the zone. GreenHDFS makes different energy savings vs. performance trade-offs in the per-zone policies as shown in the Table 4.1. Since, $Active$ zone is used to host active files and newly created files, which may have strict service level agreement (SLA) requirements, and completion deadlines; performance is of the greatest importance. GreenHDFS trades-off power savings in interest of higher performance in the $Active$ zone and no energy management is done in the servers. Majority of the servers in the cluster are assigned to the $Active$ zone upfront so that the files can be chunked across as many cluster servers as possible for fine-grained load-balancing and high parallel data access performance. Since, actively used storage capacity can actually be less than half of the total used storage capacity as per the studies of these cluster, the active data can enjoy fine-grained load-balancing even with slightly reduced number of servers.

The $Inactive$ zone consists of dormant files, i.e., files that are no longer in their active life

span and hence, are not being computed upon. Hence, performance is not as important and GreenHDFS trades-off performance in interest of higher energy-conservation in the $Inactive$ zone. GreenHDFS uses aggressive power management policies in the $Inactive$ zone to scale-down servers. Minority of the servers in the cluster are assigned to the $Inactive$ zone in order to minimize the impact of load-unbalancing caused by scale-down on active data accesses. In order to be able to allocate only few servers to the $Inactive$ zone, it is important to make each server in the $Inactive$ zone storage-heavy to reduce the server footprint of the $Inactive$ zone as dormant files may actually consume more storage capacity than the active files on these clusters. GreenHDFS uses twelve disks per server in the $Inactive$ zone vs. four disks in the $Active$ zone.

Table 4.1: **Description of the associated striping, power, reliability and data classification policies in the predictive GreenHDFS zones.**

	$Active$ **zone (SSD)**	$Active$ **zone (HDD)**	$Inactive$ **zone**
Storage Type	SSD	SATA	SATA
Number/Capacity of Disks	256GB	4, 1TB	12, 1TB
File Chunking Policy	None	Performance-Driven [61]	Energy-Efficiency Driven,None
Server Power Policy	Always-on	Always-on	Aggressive, Scale-down
Replication Policy	None	n-way	n-way
Data Classification	Small-Sized Files	Active and Newly Created Files	Dormant Files
Power Transition Penalty	None	None	High
Energy Savings	Medium	None	High
Performance	Very High	High	Low

Incorporation of Wimpy Servers: Since, $Inactive$ zone servers are used to host dormant files with very low to negligible computational load-profile, and performance requirements, GreenHDFS uses low cost, low performance and low power processors in the servers in the $Inactive$ zone. Recently, several low-power processors have been introduced in the market. A class of Intel Atom introduced in 2010 called Z560 [74] is a single-core processor which consumes only 2.5W (0.01W when idle), has a clock speed of 2.13GHz, and costs $144. GreenHDFS uses Intel Atom in the $Inactive$ zone servers. On the other hand, high power, high performance processors such a quad-core Xeon 5400 are used in the servers in the $Active$ zone as done in the state-of-the-art Big Data analytics clusters. A quad-core Xeon 5400 consumes 80-150W of power while offering clock speeds ranging from 1.86-3.50GHz and its costs range from $209.00 - $1493.00 [76]. GreenHDFS

also differentiates between the zones in the memory allocation based on the class of files residing on a zone. The $Active$ zone servers are deployed with eight DRAMs for high performance. On the other hand, GreenHDFS reduces the number of DRAMs from eight to two in the $Inactive$ zone servers which further aids in reducing power consumption. The low power $Inactive$ zone servers can still be used for computations in situations where the $Active$ zone servers are not sufficient to absorb the entire workload such as in periods of heavy, peak demand. Usage of low-cost, low-power processors significantly lowers the total cost of ownership of the cluster.

Solid State Drive Incorporation: To handle the inherent heterogeneity in the sizes, and computational jobs arrival rates of the files, GreenHDFS introduces an additional $Active$ zone layer, called $ActiveSSD$, which consists of few number of servers with Solid State Drives (SSDs) instead of Hard Disk Drives (HDDs). SSDs are known to have much higher random read access rate (IOPs), lower access latency, higher bandwidth, lower power and higher reliability compared to Hard Disk Drives (HDD) [19]. However, SSDs can not be incorporated in the system in an ad hoc manner or used naively. SSDs are much more expensive than HDDs and their incorporation has a direct affect on the total cost of ownership of the cluster. SSDs also suffer from wear-leveling and hence, can support only limited write cycles. GreenHDFS is cognizant of the limitations and advantages of the SSDs and has policies in the place that utilize the strengths of the SSDs while minimizing their limitations. GreenHDFS does total cost of ownership (TCO) analysis to guide its SSD incorporation by figuring out the number of SSDs that can be incorporated in the cluster without adversely impacting the TCO.

Energy Proportionality: In the Big Data clusters, the lowest energy-efficiency region corresponds to their most common operating mode, as shown by a six-month average CPU utilization study done at Google [28]. Per this study, servers operate most of the time between 10%-50% of their maximum utilization levels. Thus, there is a significant opportunity to consolidate computational load on the $Active$ zone and push the servers to operate closer to their energy-efficient range (i.e., 70%-80% of peak utilization). In the $Inactive$ zone, on the other hand, scaling-down servers aids in deriving energy-proportionality during idle utilization. Since, the power draw in the inactive power states is very close to zero, by transitioning the servers to inactive power state, GreenHDFS provides a mechanism to have an energy-proportional behavior in data centers built with non-energy-proportional components during periods of average load. And, the compute capacity of the $Inactive$ zone zone can always be harnessed under peak load scenarios by waking up the sleeping servers.

The rest of the chapter does a deep-dive into GreenHDFS, its various policies, components, algorithms, and cyber-physical architecture. Section 4.5.1 describes the zoning algorithm used to determine the N number of servers to assign to the $Inactive$ zone. The algorithm aims to maximize the energy savings possible via scale-down subject to performance and capacity constraints. Next, Section 4.5.2 elaborates the zone partitioning schemes, which figure out the actual N physical servers to assign to the $Inactive$ zone in a thermal- and performance-aware manner. The cyber-side monitor, controller, and actuator, energy-aware file management and placement policies, and per-zone policies and managers are given in Section 4.6. The predictive modeling in place in GreenHDFS is elaborated in Section 4.6.1. The physical-side monitor, controller, and actuator are described in Section 4.7. Lastly, the evaluation of the Reactive and Predictive variants of GreenHDFS is presented in Sections 4.9 and 4.10.

4.4.1 Scale-Down Challenges in GreenHDFS

GreenHDFS data-centric scale-down approach requires consolidation of the dormant files. This raises additional questions/challenges on top of the scale-down challenges discussed earlier:

- How should dormant files get identified?
- How should truly dormant files get distinguished from files that are simply experiencing a temporary lull in jobs arrival?
- What are the repercussions of inaccurate dormancy determination?
- Should the dormancy determination be coarse-grained at data-set level or fine-grained at file-level? What are the advantages and disadvantages of either approaches?
- Is it possible to determine dormancy onset in the files predictively?
- Should thresholds be used in the determination; if yes, what is the sensitivity of the system to threshold values?
- How should the situation when a dormant file is no longer dormant be handled?
- What support does dormancy determination require from the rest of the system? What is

the performance repurcussion of this determination?

- How often should the dormant files get moved to the $Inactive$ zone?
- When should the dormant files get moved to the $Inactive$ zone?
- How should the $Inactive$ zone servers be selected to reduce the migration time of the dormant files?
- What trade-offs are involved in the selection of the $Inactive$ zone servers?
- How should the cluster get split into logical $Active$ and $Inactive$ zones in an energy, and performance-aware fashion?

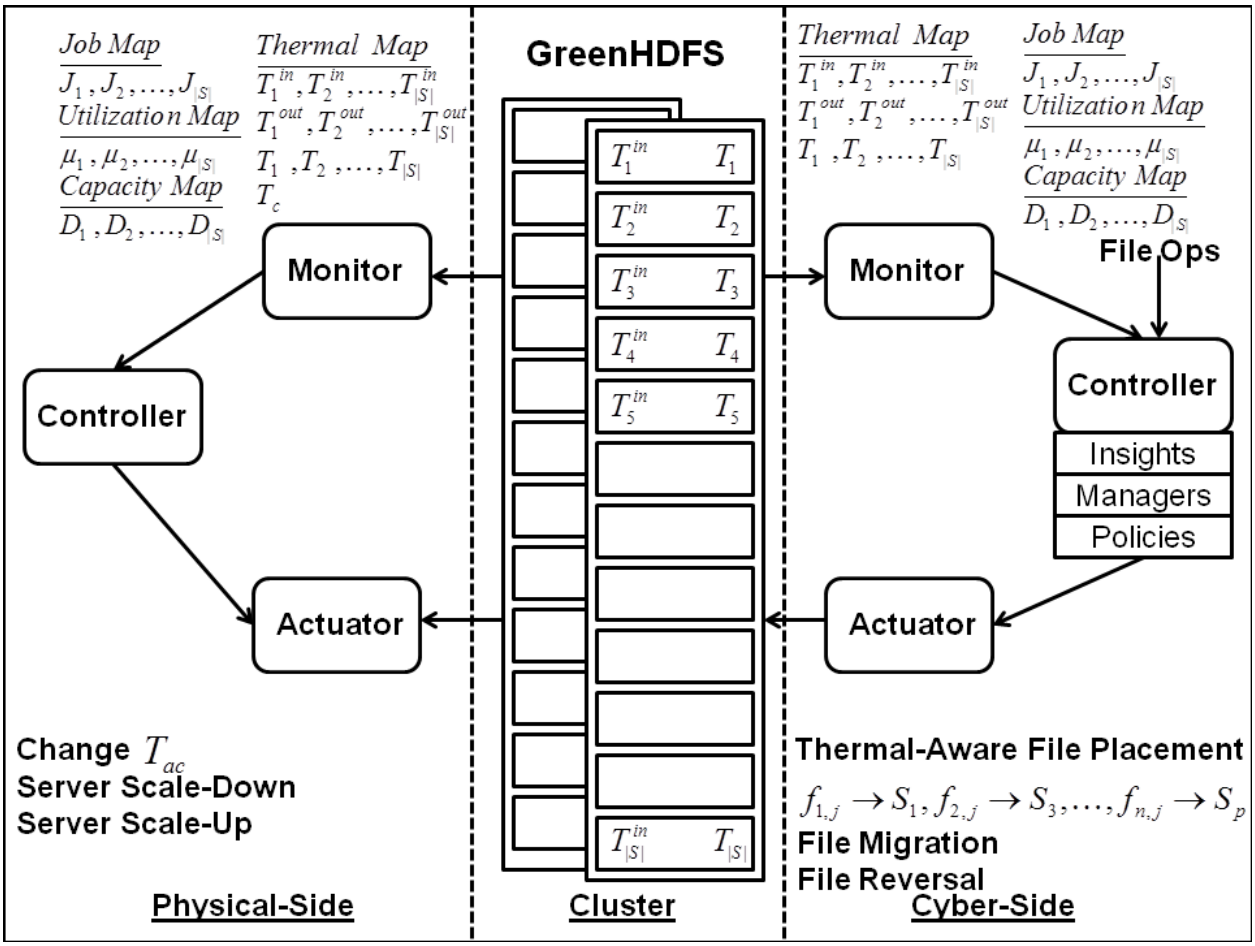

Figure 4.7: **Cyber-Physical** $GreenHDFS$**.**

4.5 Cyber-Physical System

GreenHDFS is designed as a cyber-physical system and consists of a physical-system, and physical-side and cyber-side controllers, actuators, and monitors as shown in the Figure 4.7. The cyber- and physical control loops make GreenHDFS a self-adaptive, energy-conserving system. The physical-system in GreenHDFS comprises of $|S|$ servers in the cluster c and their associated temperature sensors. Each server *i* in the cluster has a temperature sensor at its inlet to measure the temperature $T^{in}_{i,t}$ of the cold air coming into the server, a temperature sensor at its exhaust to measure the temperature $T^{out}_{i,t}$ of the hot air exhausted from the server, and a sensor to measure the steady-state temperature of the server $T_{i,t}$. And, finally, a temperature sensor measures the overall temperature $\mathcal{T}^{out}_{c,t}$ of the cluster. During the run-time, the physical- and cyber-monitors collect information from the physical system and send the relevant information to the physical- and the cyber-controller respectively. The controllers then use the run-time information to guide their policies and decision making process.

GreenHDFS has an initial bootstrapping phase in which the following two steps are performed: 1) zone partitioning algorithm is used to figure out the optimal N number of servers to assign to the $Inactive$ zone. The algorithm aims to minimize the energy costs subject to performance and capacity constraints, and 2) thermal-aware zone partitioning schemes are used to select the servers for the $Inactive$ and $Active$ zones keeping energy costs, and performance trade-offs in mind.

Table 4.2: **Table of Notations Used in the Zone Partitioning Algorithm.**

Variable	Description	Units
t	Time interval	Seconds
$P_{i,t}$	Power consumption of server i at time t	Watts (W)
$\|S\|$	Total number of servers in the cluster	
N	Number of servers in the $Inactive$ zone	
$\|S\| - N$	Number of servers in the $Active$ zone	
b_{max}	Data access bandwidth in the outer-most hard disk cylinder zone	Megabytes/Second (MBs-1)
b_{avg}	Average data access bandwidth of the disk	Megabytes/Second (MBs-1)
d_{active}	Disks in $Active$ zone server i	
c_{active}	Capacity of the disks in $Active$ zone server i	Terabytes (TB)
$d_{inactive}$	Disks in $Inactive$ zone server i	
$c_{inactive}$	Capacity of the disks in $Inactive$ zone server i	Terabytes (TB)
U_{max}	Maximum storage capacity of the cluster	Terabytes (TB)
$\bar{\lambda}_{overall}$	Average overall jobs arrival rate in the cluster	
$\bar{\lambda}_{GreenHDFS,i}$	Average jobs arrival rate to a server i in the GreenHDFS managed cluster	
$\bar{\lambda}_{baseline,i}$	Average jobs arrival rate to a server i in a cluster with no energy management	
$\bar{r}_{overall}$	Overall number of jobs in the cluster	
$\bar{p}_{overall}$	Average parallelism of jobs in the cluster	
$r_{sub-job}$	Data size computed upon by a job's sub-job	
$t_{serv,i}$	Job service time of server i	
$t_{seek,i}$	Seek time of data access at server i	
$t_{rot,i}$	Rotation time of data access at server i	
$\rho_{baseline,i}$	Utilization of server i in $Active$ zone in GreenHDFS cluster	
$\rho_{GreenHDFS,i}$	Utilization of server i in $Active$ zone in Baseline cluster	
$E[t_{serv,i}]$	Expected service time of a job's sub-job running on server i	
$E[t_{resp,i}]$	Expected response time of a job's sub-job running on server i	
$E[T_{resp}]$	Expected overall response time of job	
$\hat{\mathfrak{t}}$	Ratio of active data to overall data in cluster	

4.5.1 Zone Partitioning Algorithm

The zone partitioning algorithm determines the optimal N number of servers to assign to the $Inactive$ zone. The value is determined in a way that maximizes the energy costs savings while ensuring that the throughput and response time of the overall system is not impacted. There is a possibility of some overall performance impact in the $Active$ zone in GreenHDFS because of the load-unbalancing caused by the scale-down of the $Inactive$ zone servers. Load-unbalancing may increase queuing delays in the $Active$ zone as it now has lower $|S| - N$ number of servers instead of $|S|$ servers. While the performance impact is limited as $N << |S|$ and amount of active data is less than the amount of dormant data, GreenHDFS still takes measures to alleviate the queuing delays possible in $Active$ zone during data access by reducing the service time of each data access request in the $Active$ zone.

A study of 5000 Google compute servers, showed that the servers spent most of the time within the 10% - 50% CPU utilization range [28] as Big Data analytics is a data-intensive class of workload. Hence, there is no concern of load-unbalancing of the CPUs in GreenHDFS; instead, there are ample of opportunities to increase the CPU utilization of the servers in the $Active$ zone without the danger of exceeding the performance-driven, load provisioning guidelines. Hence, the main focus of GreenHDFS is on reducing the load-unbalancing impact of scale-down on data accesses.

Modern disks use zoned bit recording (ZBR) whereby the disk cylinders are divided into zones based on their distance from the center of the disk as shown in the figure 4.8. The outer-most cylinder zones $zone0$ and $zone1$ have many more sectors per track than the inner-most cylinder zones. This results in a factor of difference in the transfer bandwidth available across zones. Since, the rotational speed of the disk is the same, more data can be read from the outer cylinder zones of the disk than from the inner cylinder zones in the same time leading to a much higher data transfer rate in the outer cylinder zones. The difference in the transfer rate between the inner and the outer cylinder zones can be greater than two for some of the enterprise disks available today.

GreenHDFS aims to take advantage of the much higher data transfer bandwidth available on

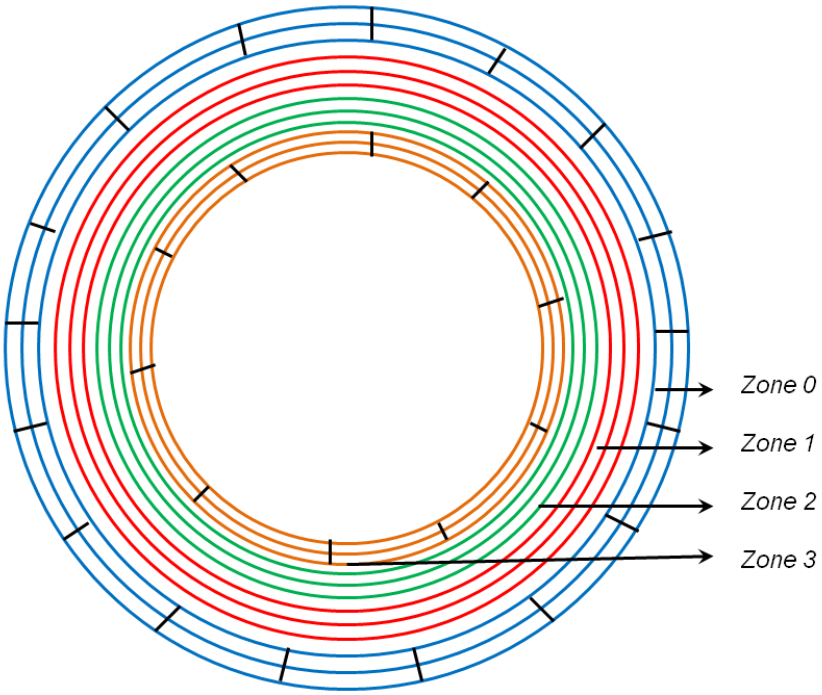

Figure 4.8: **Cylinder zones in modern disks with zoned bit recording. The outer cylinder zones have more sectors per track than the inner cylinder zones, resulting in higher data transfer bandwidth in the outer cylinder zones such as $zone0$ or $zone1$.**

the outer cylinder zones and places active files, i.e., files that are still being actively computed upon and have not yet reached the end of their active lifespan $\mathcal{L}_{active,j}$ on the $zone0$ (outer-most cylinder zone) of the disks to avail higher transfer bandwidth. The log processing workloads such as clickstream processing have a very skewed, news-server-like access pattern whereby the new incoming files get a much larger share of the daily incoming jobs. GreenHDFS places the new incoming files on the outer-most cylinder zones as well in addition to the active files. Higher transfer bandwidth reduces the data transfer time. Now, the file chunks are typically 64MB-128MB on the production Big Data analytics clusters. The large chunk size is chosen specifically to amortize the affect of seek and rotational times on the overall disk access time. Thus, the overall disk access time is predominated by the transfer time; hence, reducing the transfer time reduces the disk access time.

If the file accesses are heavily skewed to the new and active files, as is the case with the log processing workloads such as clickstream processing, there is an added performance advantage of consolidating the active and new incoming files on the outer-most disk cylinders zones. The

advantage arises from a reduction in the disk seek time as all the data that being accessed is coalesced on the outer disk tracks. The disk head doesn't have to seek all over the disk for reading/writing the data; both reads and writes are fulfilled from the outer disks tracks. And, of course, there is a power advantage of the reduced seek times as well on the overall disk power consumption.

Now, the effectiveness of such an active file layout is contingent upon having enough space in the outer cylinder zones for placing the new incoming files and active files. If the place is not automatically available, there would be a need to reorganize the files on the disk and move the older, still active files to the inner disk cylinder zones, so that space is created for hosting the new files on the outer disk cylinder zone. Such a reorganization may be expensive depending on the extent of reorganization required. Fortunately, an analysis of the clickstream processing workload (most popular and important Big Data analytics workload) of a production cluster at Yahoo! shows that the majority of the used storage capacity has a very short active lifespan $\mathcal{L}_{active,j}$, as shown in the Figure 4.9. Majority of the files are actively computed upon for a short while and then, they are either deleted or are retained in the system for regulatory or compliance reasons. Now, GreenHDFS already has a policy *File Migration Policy* running in the $Active$ zone which monitors the dormancy of the files, and moves newly dormant files to the $Inactive$ zone as discussed in Section 4.6.7. Thus, the *File Migration Policy* will automatically move the files that are no longer active to the $Inactive$ zone everyday, thereby automatically freeing up space on the outer disk tracks for the new incoming files. Hence, the files don't need much reorganization across the disk cylinder zones in GreenHDFS.

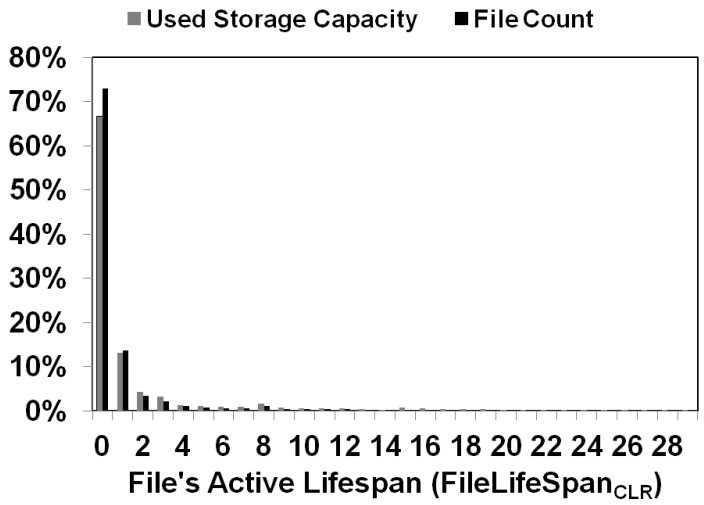

Figure 4.9: **File's active life span histogram of production real-world Big Data analytics Hadoop cluster at Yahoo!. Majority of the files and the used storage capacity have a very short active life span.**

4.5.1.1 Feasibility Study

The outer disk cylinder zone, $zone0$, typically is capable of hosting 25% of the overall disk data storage capacity. It is very important to do a feasibility study to evaluate if the active and new incoming files can all be fit together on $zone0$ or $zone1$. Assume each server in the $Active$ zone has d_{active} number of disks and each disk has a storage capacity c_{active}. The percentage $\mathfrak{p}$ of active data stored in each disk of a server in the $Active$ zone can be estimated by the Equation 4.5 shown below:

$$\mathfrak{p} = \frac{\hat{\mathfrak{r}} \cdot U}{(|S| - N) \cdot c_{active} \cdot d_{active}} \tag{4.5}$$

where U is the total used storage capacity on the cluster, $\hat{\mathfrak{r}}$ is the percentage of the currently active data in the cluster, $|S|$ is the number of servers in the cluster, and N is the number of the servers in the $Inactive$ zone. If the value of $\mathfrak{p}$ is low, the active data can easily fit in the outer-most cylinder zones of the disks.

It is equally important to ensure that the dormant files will indeed fit on the N servers in the $Inactive$ zone. In the $Inactive$ zone, each server has $d_{inactive}$ number of disks, and each disk

has a storage capacity $c_{inactive}$. The dormant files will fit only if $\mathfrak{p}_{inactive} \leq 1$ in the Equation 4.6. The condition ensures that the $Inactive$ zone servers are able to host the dormant files without exceeding 100% storage capacity of the $Active$ zone servers.

$$\mathfrak{p}_{inactive} = \frac{(1 - \hat{\mathfrak{r}}) \cdot U}{N \cdot c_{inactive} \cdot d_{inactive}} \tag{4.6}$$

Example: Let's consider the production Big Data analytics Hadoop cluster at Yahoo! which has been extensively used in the evaluation and analysis of this dissertation. This cluster has total number of servers $|S| = 2600$ and used storage capacity $U =$ 5PB. In the baseline case with no GreenHDFS deployment, all the servers in the cluster store the entire data without any data-differentiation and $N = 0$. In the baseline cluster, $d_{active} = 4$ and $c_{active} = 1$ TB. As per Equation 4.5, $\mathfrak{p} = 0.48$ which means that storage capacity of the disks in the servers is 48% utilized assuming a uniform file chunks distribution over the servers in the cluster.

Now, we observed that out of 5PB used storage capacity, only 44% of the data in the cluster was actually being actively accessed, and the remaining 56% of the data was lying dormant without getting accessed (and, still needed to exist for regulatory or compliance reasons). Thus, $\hat{\mathfrak{r}} = 0.44$. Now, let GreenHDFS be deployed in this cluster and let 20% of the servers in the cluster be assigned to the $Inactive$ zone, making $N = 520$ and the servers assigned to the $Active$ zone are $|S| - N =$ 2080. GreenHDFS does data-differentiation and now, only the active files amounting to $(\hat{\mathfrak{r}} \cdot U)$ storage capacity are placed on the $Active$ zone's $|S| - N = 2080$ servers and the dormant files amounting to $((1 - \hat{\mathfrak{r}}) \cdot U)$ storage capacity are placed on the $N = 520$ $Inactive$ zone servers. Plugging in these values in Equation 4.5 yields $\mathfrak{p} = 0.26$. Thus, the disks in the $Active$ zone servers are only 26% full with the active data. This data can easily fit on the outer-most cylinder zones of the disks, hence showing the feasibility of using the outer cylinder zones of disks with zoned bit recording for active file placement in the cluster.

In the $Inactive$ zone, each server has $d_{inactive} =$ 12 number of disks and each disk has a storage capacity $c_{inactive} =$ 1TB. Typically, the low-end servers used in the commodity clouds can only support up to twelve disks and hence, $d_{inactive}$ cannot exceed twelve. Plugging in the values in Equation 4.6 yields $\mathfrak{p}_{inactive} =$ 44%. Hence, we see that the $Inactive$ zone disks can easily fit the dormant files without exceeding storage capacity allocated to them.

GreenHDFS does zone partitioning in a way that ensures performance while resulting in energy

savings. In Big Data analytics cluster, there are two main performance metrics of the data access: throughput and response time.

Throughput: The throughput of a cluster is defined as the total data that can be transferred per second. In case of a baseline cluster with no GreenHDFS deployment, the throughput is given by the following equation:

$$|S| \cdot b_{avg} \tag{4.7}$$

where b_{avg} is the average data transfer rate of the disk. The average is taken over the transfer rates of the various cylinder zones present in the disk.

In case of GreenHDFS, courtesy of cluster servers' zone partitioning, the throughput that is achieved by the placing the active files on the outer cylinder zones of disks on the $|S| - N$ $Active$ zone servers is given by:

$$(|S| - N) \cdot b_{\max} \tag{4.8}$$

where $b_{\max}$ is the data transfer bandwidth for the outer-most cylinder zone in the disks with zoned bit recording. Both $b_{\max}$ and b_{avg} are available from disk data sheets.

For zone partitioning to not have any impact on the throughput, the throughput of the active files residing in the $Active$ zone in GreenHDFS should be at least equal to, if not greater than the throughput of the baseline cluster. Thus:

$$(|S| - N) \cdot b_{\max} \geq |S| \cdot b_{avg} \tag{4.9}$$

Average Response Time: It is important to ensure that the average response time of active files in a cluster managed by GreenHDFS is at-least the same as the response time in a state-of-the-art cluster with no GreenHDFS deployment. We used paper by Scheuermann et. al. [121] to guide our average response time calculation and made changes to the assumptions and definitions as needed by GreenHDFS.

Assume, $\bar{\lambda}_{overall}$ is the average overall jobs arrival rate in the cluster in an epoch t. $\bar{r}_{overall}$ is the average data size of the data to be computed upon by the jobs in the cluster in epoch t and $\bar{p}_{overall}$ is the average parallelism of the jobs, i.e., the average number of sub-jobs that the jobs get split into based on the number of chunks in the target files.

With a good predictive *File Migration Policy* active in GreenHDFS, all the Big Data analytic jobs

will be targeted only to the files residing in the $Active$ zone. Assuming perfect load balancing and an uniform file chunk distribution across the servers in the cluster, the jobs arrival rate to a server i in a set of $|S| - N$ $Active$ zone servers in the cluster, denoted as $\lambda_{GreenHDFS,i}$ is computed as:

$$\lambda_{GreenHDFS,i} = \frac{\bar{\lambda}_{overall} \cdot \bar{p}_{overall}}{|S| - N} \tag{4.10}$$

In case of baseline cluster with no GreenHDFS deployment, the files are distributed across the $|S|$ servers in the cluster. Hence, the jobs arrival rate to a server i in a set of $|S|$ servers in the cluster, denoted as $\lambda_{baseline,i}$, is computed as:

$$\lambda_{baseline,i} = \frac{\bar{\lambda}_{overall} \cdot \bar{p}_{overall}}{|S|} \tag{4.11}$$

Average sub-job size, $\bar{r}_{sub-job}$, can be derived as:

$$\bar{r}_{sub-job} = \frac{\bar{r}_{overall}}{\bar{p}_{overall}} \tag{4.12}$$

In case of Map Reduce based Big Data analytics clusters such as Hadoop clusters, every file is chunked and replicated across the cluster. The default chunk size of a file is 64MB - 128MB. A Map Reduce job directed to a file $\tilde{f}_j$, spawns Map tasks equal to the number of chunks $\hat{n}_j$ in the file $\tilde{f}_j$. In interest of high data-local data access performance, each Map task (i.e., sub-job) is sent to a server hosting a chunk of the file $\tilde{f}_j$. Thus typical value of $\bar{r}_{sub-job}$, i.e., the data accessed by a sub-job is the default chunk size of the cluster, i.e., 64MB - 128MB. The value of $\bar{p}_{overall}$ is equivalent to the average number of chunks in the files in the clusters.

The service time denoted as $t_{serv,i}$ for an individual sub-job to server i is computed as follows:

$$t_{serv,i} = max\left(t_{seek,i}\right) + max\left(t_{rot,i}\right) + t_{xfer,i} \tag{4.13}$$

where $t_{seek,i}$ and $t_{rot,i}$ are the seek and rotation time involved in servicing the request at a server i. We make a simplifying assumption that the maximum seek time is twice the average seek time

rotation latency $t_{rot,i}$ is the average latency mentioned in the data-sheet of a disk instead of using distributions to model the seek and rotation time.

The utilization $\rho_{baseline,i}$ of a server in baseline cluster, i.e., a cluster with no GreenHDFS deployment is given by the following equation:

$$\rho_{baseline,i} = \lambda_{baseline,i} \cdot t_{serv,i} \tag{4.14}$$

The utilization $\rho_{GreenHDFS,i}$ of an $Active$ zone server in GreenHDFS cluster is given by the following equation.

$$\rho_{GreenHDFS,i} = \lambda_{GreenHDFS,i} \cdot t_{serv,i} \tag{4.15}$$

We assume a M/G/1 queuing model to represent a server and the expected value of the response time of a sub-job $r_{sub-job}$ denoted as $t_{resp,i}$ can be given by [121]:

The expected value of the sub-job's response time in case of baseline cluster is given by:

$$E[t_{resp,i}] = E[t_{serv,i}] + \rho_{baseline,i} \cdot E[t_{serv,i}] \frac{1 + m_i^2}{2 \cdot (1 - \rho_{baseline,i})} \tag{4.16}$$

The expected value of the sub-jobs's response time in case of GreenHDFS cluster is given by:

$$E[t_{resp,i}] = E[t_{serv,i}] + \rho_{GreenHDFS,i} \cdot E[t_{serv,i}] \frac{1 + m_i^2}{2 \cdot (1 - \rho_{GreenHDFS,i})} \tag{4.17}$$

m_i stands for the coefficient of variance of the service time of server i and is computed as follows:

$$m_i^2 = \frac{VAR[t_{serv,i}]}{E[t_{serv,i}]^2} \tag{4.18}$$

We assume the variance in the $t_{serv,i}$ to be zero as we are assuming fixed and maximum possible values of seek time and rotational latency. The size of a sub-job is assumed to to be the default chunk size of 64MB-128MB and hence, the transfer time is also fixed. Thus, $m_i = 0$ in our case.

The overall response time of the sub-jobs of a Big Data analytics job is the equivalent to the maximum value of $t_{resp,i}$ seen by the servers on which sub-jobs corresponding to the Big Data analytic job are running, as the overall response time of a Big Data analytic job is equivalent to the completion time of the longest running straggler sub-job.

$$E[T_{resp}] = max_i \left(t_{resp,i}\right) \tag{4.19}$$

The value of $E[T_{resp}]$ can be approximated as follows:

$$E[T_{resp}] = E[t_{resp,i}] + \sqrt{VAR[t_{resp,i}]} \cdot \sqrt{2 \cdot \bar{p}_{overall}} \tag{4.20}$$

The variance in case of GreenHDFS is calculated as follows

$$VAR[t_{resp,GreenHDFS,i}] = VAR[t_{resp,i}] + \frac{\lambda_{GreenHDFS,i} \cdot E[(t_{serv,i})^3]}{3 \cdot (1 - \rho_{GreenHDFS,i})} + \frac{(\lambda_{GreenHDFS,i})^2 \cdot E[(t_{serv,i})^3]}{4 \cdot (1 - \rho_{GreenHDFS,i})^2} \tag{4.21}$$

$$VAR[t_{resp,baseline,i}] = VAR[t_{resp,i}] + \frac{\lambda_{baseline,i} \cdot E[(t_{serv,i})^3]}{3 \cdot (1 - \rho_{baseline,i})} + \frac{(\lambda_{baseline,i})^2 \cdot E[(t_{serv,i})^3]}{4 \cdot (1 - \rho_{baseline,i})^2} \tag{4.22}$$

The servers in the $Active$ zone draw power given by Equation 4.23.

$$P_{active} = \sum_{i=1}^{|S|-N} P_i \tag{4.23}$$

We make a simplistic worst-case assumption that all the servers in the $Active$ zone are running at full utilization and thereby, are drawing peak power. This is the worst-case scenario for any cluster. In reality, the servers are at a much less utilization and draw much less power than the peak power. Thus, P_i in the above equation can be substituted by the peak power draw of the servers in the cluster. Since the servers in the $Inactive$ zone are scaled-down, they don't consume any power and all the power consumption in the cluster comes from the $Active$ zone servers. Increasing N by allocating more servers to the $Inactive$ zone has the potential to save lot more energy; however, larger the N, lower the throughput and higher the response time of

the cluster.

To decide on the optimal way to partition a cluster into an $Inactive$ zone consisting of N servers and an $Active$ zone consisting of $|S| - N$ servers, we seek the N that minimizes the power consumption in the cluster (i.e., cooling and server power), subject to capacity and performance constraints as follows:

$$\min_{N} = P_c + \sum_{i=1}^{|S|-N} P_i \tag{4.24}$$

subject to:

$$(|S| - N) \cdot b_{\max} \geq |S| \cdot b_{avg} \tag{4.25}$$

$$(|S| - N) \cdot c_{active} \cdot d_{active} \geq \hat{\mathfrak{r}} \cdot U_{max} \tag{4.26}$$

$$N \cdot c_{inactive} \cdot d_{inactive} \geq (1 - \hat{\mathfrak{r}}) \cdot U_{max} \tag{4.27}$$

$$E[T_{resp,GreenHDFS}] \leq E[T_{resp,baseline}] \tag{4.28}$$

Where, $E[T_{resp,GreenHDFS}]$ is given by Equation 4.21, $E[T_{resp,baseline}]$ is given by Equation 4.22, and U_{max} is the maximum storage capacity possible in the cluster. Since, the above objective function is monotonically increasing in N, the above optimization can be solved easily by trying out increasing values of N until the constraints are violated.

4.5.2 Zone Partitioning Schemes

The thermal-aware zone partitioning scheme is run during initial bootstrapping phase to perform the following tasks: 1) rank the cooling-efficiencies of the servers, and 2) partition the servers in the cluster into cooling-efficiency differentiated server zones. The bootstrapping phase creates a thermal-profile of the cluster, with all servers kept at same utilization, to identify the inherently cooling-inefficient servers. The bootstrapping phase uses MentorGraphics floVENT [6], a computational fluid dynamics (CFD) [115] simulator, to simulate the cluster under consideration. floVENT simulates a cluster with great accuracy including the geometry, layout, and configuration of the compute equipment. floVENT has been extensively used and validated in several research papers in the past [29, 107, 126].

At the conclusion of the simulation, floVENT provides the inlet and the exhaust temperature for each server and the CRACs in the cluster. The bootstrapping phase then ranks the servers in the cluster in decreasing order of their inlet temperatures (i.e., cooling-efficiency). It also creates a per-rack ranking of the inlet temperature (i.e., cooling-efficiency) of the servers in the racks. The bootstrapping phase needs to be rerun if there are physical changes to the hardware or layout in the data center. Past research has shown that changes in the physical layout of the data center can result in significant changes in the air flow in the data center and hence, result in a change in the thermal-profile of the servers [114].

Figure 4.10, shows a histogram of the range of the inlet temperatures present in a traditional air-cooled data center. MentorGraphics floVENT was used to simulate the cluster under consideration. There is significant difference in the inlet temperatures of the servers as illustrated in several other studies of real-world data centers in the past as well.

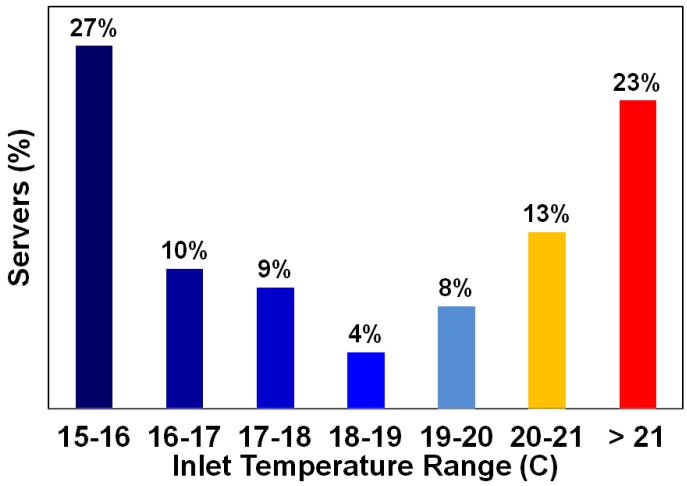

Figure 4.10: **The histogram of the inlet temperatures in a Big Data analytics cluster with no energy management.**

The zone partitioning schemes use the thermal-efficiency ranking to partition the servers in the cluster into cooling-efficiency differentiated zones. The zone partitioning algorithm covered in Section 4.5.1 is used to determine the N number of servers to assign to the $Inactive$ zone.

The $Inactive$ zone servers are used to store dormant file class, i.e., file class with very low or negligible data access-profile (cyber), and thereby low computation-profile, generating low-server energy, hence requiring low cooling energy. The $Inactive$ zone servers experience significant

idleness and can be scaled-down. The $Active$ zone servers are used to store the rest of the file classes with high/medium/low data access-profile, generating high/medium-server energy, and hence requiring high/medium cooling energy.

GreenHDFS assigns the most inherently cooling-inefficient servers in the cluster to the $Inactive$ zone. Thus, GreenHDFS ensures that the cooling-inefficient servers receive negligible computations, don't generate much heat and their exhaust temperature remain bounded. Such a thermal-aware data zone partitioning reduces the hot spots in the cluster leading to an overall lower temperature in the cluster which in turn reduces the cooling energy costs. The rest of the more cooling-efficient servers are assigned to the $Active$ zone. Since $Active$ zone servers receive majority of the computational load, they dissipate more heat, and require more cooling to keep the server temperatures in check. Thus, the cooling-efficient servers are a good fit for the active files. GreenHDFS uses the following two zone partitioning schemes to split the servers in the cluster into $Inactive$ and $Active$ zones:

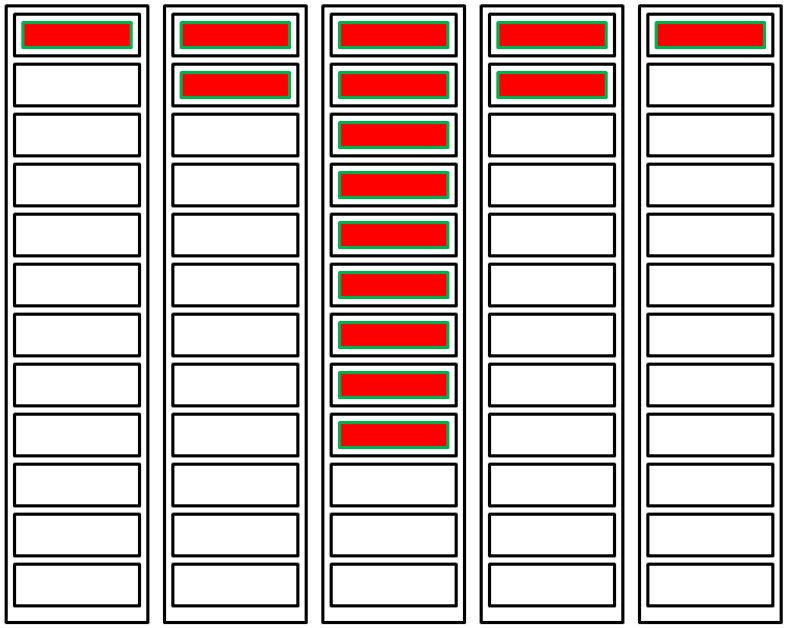

Figure 4.11: **The most cooling-inefficient servers cluster-wide are assigned to the $Inactive$ zone.**

4.5.2.1 Cluster-Level Zone Partitioning

In this scheme, GreenHDFS uses cluster-wide ranking of the cooling-efficiencies of the servers and assigns the N number of most cooling-inefficient servers in the cluster to the $Inactive$ zone. Typically the racks in the center rows of the cluster have higher inlet temperatures than the racks in the outer-most rows as center rows experience more hot air recirculation and air bypass. Thus, the center racks have a much higher share of cooling-inefficient servers than the outer racks. With the cluster-wide scheme, the racks in the center rows will have more servers assigned to the $Inactive$ zone than the outer-most racks as shown in the Figure 4.11. Some of the outer racks may even not have any servers assigned to the $Inactive$ zone. Let's consider the impact of the cluster-wide zone partitioning scheme on cooling energy costs savings, compute energy costs savings, and performance.

4.5.2.1.1 Cooling Energy Costs The cluster-wide zone partitioning has the most potential to save cooling energy costs in GreenHDFS. For example, if N is as large as 30%, then with an inlet temperature distribution as shown in the Figure 4.10, all the servers with temperature higher than 21 will get assigned to the $Inactive$ zone. Since, $Inactive$ zone is used to host inactive data with very low computational load, and servers in the $Inactive$ zone are scaled-down aggressively, this leads to significant alleviation of thermal hot-spots in the cluster. Thermal hot-spots are the primary cause of high cooling energy costs in the clusters. The servers with impaired cooling-efficiency are unable to dissipate heat generated by a server's computational load effectively. This results in an increase in the server's temperature. It is important not to exceed the maximum temperature tolerance of the servers T_{max} as hardware failure rates start increasing with an increase in the temperature beyond T_{max}. In interest of thermal-reliability of the servers, the cooling systems are run at much lower set point temperature T_{ac} to alleviate the hot-spots. The efficiency of the cooling system goes down with a reduction in T_{ac} resulting in higher cooling energy costs. If the hot-spots are reduced courtesy of a technique like cluster-wide zone partitioning, the cooling systems need not be run at very low temperatures. Increasing the set point temperature of the cooling system increases its operating efficiency, thereby resulting in even more cooling energy costs savings. However, the cooling energy costs savings may come with some performance trade-offs as illustrated below.

4.5.2.1.2 Server Energy Costs The *File Placement Manager* discussed in Section 4.6.6 keeps some servers in an active power state in the $Inactive$ zone to act as the target for the dormant files that are getting migrated from the $Active$ zone to the $Inactive$ zone by the *File Migration Policy*, while the rest of the servers in the $Inactive$ zone are scaled-down to conserve energy. These servers are kept in active power state till they are filled to their capacity, and are then scaled-down. Next set of $Inactive$ zone servers are chosen at that point to host the next batch of dormant files. In case of the cluster-level partitioning, only one server cluster-wide needs to be in an active state, thereby, allowing majority of the $Inactive$ zone servers to be scaled-down, increasing energy savings.

Now, to evaluate the energy costs savings, it is important to consider the number of wake-ups scaled-down servers in the $Inactive$ zone are subjected to when a file residing on the $Inactive$ zone is accessed. If such accesses happen soon after the file is moved to the $Inactive$ zone, i.e., there is some temporal correlation between the time a file is migrated and the accesses, than number of server wake-ups are less with cluster-level zone partitioning scheme as the files that become dormant on the same day most likely reside on the same server.

4.5.2.1.3 Write Performance Big Data analytics framework such as Map Reduce uses rack-awareness while writing data onto the cluster as intra-rack bandwidth is higher than inter-rack bandwidth. Two replicas of each chunk are written on servers in the same rack to take advantage of the intra-rack bandwidth for reduction in the writing time. If a rack (e.g., one of the central racks), has a large number of servers assigned to the $Inactive$ zone servers, then it would have less number of servers in the $Active$ zone and thereby, the replication pipeline at the time of the writes may not be able to find servers on the same rack to write the replica. This may result in an increase in the write latency.

4.5.2.1.4 File Migration Performance The *File Migration Policy* discussed in Section 4.6.7.2, utilizes rack-awareness while migrating files from the $Active$ zone to the $Inactive$ zone, to take advantage of the higher intra-rack network bandwidth compared to the lower inter-rack network bandwidth; and, aims to migrate dormant files from a $Active$ zone server to a $Inactive$ zone server residing on the same rack. If there are no servers assigned to the $Inactive$ zone in some racks, then the *File Migration Policy* has to resort to inter-rack migration, thus taking longer to migrate the data.

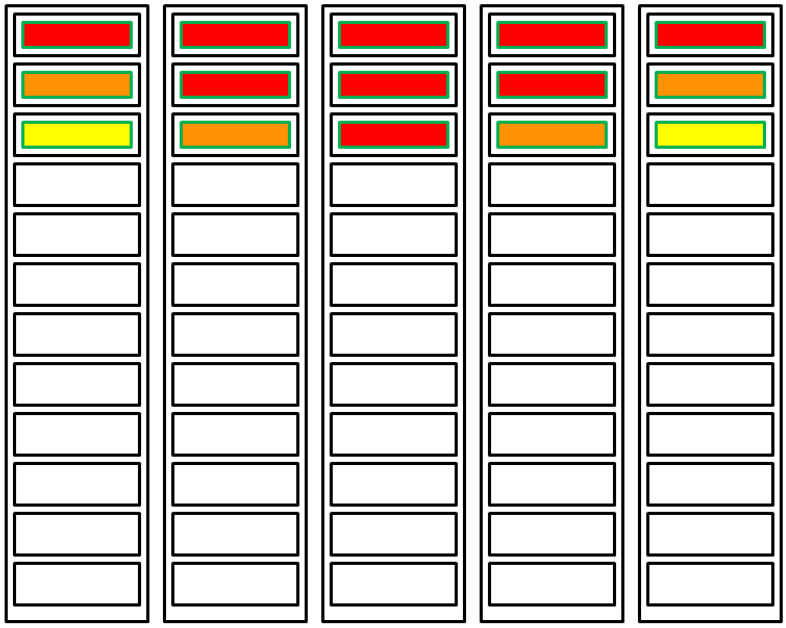

Figure 4.12: **The most cooling-inefficient servers in every rack are assigned to the $Inactive$ zone.**

4.5.2.2 Rack-Level Zone Partitioning

In this scheme, GreenHDFS uses per-rack cooling-efficiency ranking to assign servers to the $Inactive$ zone and the $Active$ zone. The $N/(|R| \cdot \mathfrak{N})$, N is the number of servers to be assigned to the $Inactive$ zone as determined by the zone partitioning algorithm, $|R|$ is the number of servers in each rack, and $\mathfrak{N}$ are the number of total racks in the cluster. Hence, in this scheme, most cooling-inefficient servers per rack are assigned to the $Inactive$ zone, and the rest of the servers in the rack to the $Active$ zone. This partitioning results in a uniform split of servers per rack to the zones as shown in the Figure 4.12. Let's consider the impact of the cluster-wide zone partitioning scheme on cooling energy costs savings, compute energy costs savings, and performance.

4.5.2.2.1 Cooling Energy Costs Since, this scheme uses rack-level cooling-efficiency ranking instead of cluster-wide cooling-efficiency ranking, it misses including some of the servers

that rank high in the cluster-wide cooling-inefficiency ranking in the $Inactive$ zone. Now, these cooling-inefficient servers get assigned to the $Active$ zone, and have active files stored on them. When the computational load does happen on the files stored on these cooling-inefficient servers, these servers aren't as effective in dissipating the heat generated from load, resulting in a temperature rise in the servers. As a result, thermal hot-spots don't get alleviated completely in the data center unlike the cluster-wide zone partitioning scheme and cooling energy costs reduction may not be as low as that in the cluster-level partitioning scheme.

4.5.2.2.2 Server Energy Costs The *File Placement Manager* discussed in Section 4.6.6 keeps one server per-rack in the cluster in an active power state in the $Inactive$ zone to act as the target for the dormant files that are getting migrated from the $Active$ zone to the $Inactive$ zone by the *File Migration Policy*. Keeping one server from each rack in an active power state leads to more number of servers than in case of cluster-wide zone partitioning scheme. In case of rack-level zone partitioning, $\mathfrak{N}$ servers need to be in an active state, where $\mathfrak{N}$ is the number of racks in the cluster, as opposed to one server in active power state in case of cluster-level zone partitioning scheme. $\mathfrak{N}$ can be as high as eighty-six in a cluster with number of servers $|S|$ $=$ 4000, which is very typical of production Big Data analytics clusters. Since less number of servers are in a scaled-down state, the server energy costs savings are lower with rack-level zone partitioning scheme.

Another aspect to consider is the number of server wake-ups that happen when files residing on the $Inactive$ zone are accessed. If such accesses happen soon after a file is moved to the $Inactive$ zone, i.e., there is some temporal correlation between the time a file is migrated and the accesses, than number of server wake-ups are less with cluster-level zone partitioning scheme as the files that become dormant on the same day most likely reside on the same server. Whereas, in case of rack-level zone partitioning, incoming, newly dormant files get split across the racks and hence, resulting in more server wake-ups which in turn reduce the energy savings.

4.5.2.2.3 Write Performance Since, servers are assigned to the $Inactive$ zone from each rack uniformly in the rack-level zone partitioning scheme, no rack runs into the lack of $Active$ zone servers issue experienced by cluster-level partitioning. Hence, replication pipeline at the time of the writes is able to find servers on the same rack to write the replicas at intra-rack bandwidth. Thus, write performance is not impacted in case of rack-level partitioning.

4.5.2.2.4 File Migration Performance Only the $Inactive$ zone servers in the same rack are chosen as the target of the newly dormant files in that rack. Thus, file migration enjoys intra-rack bandwidth which is higher than the inter-rack bandwidth. The files get migrated faster, thereby reducing the impact of file migration.

4.6 Cyber-Side

The cyber-side system comprises of a monitor, a controller, and an actuator. The cyber-controller is the brain of GreenHDFS and takes all energy- and file-management policy decisions. Cyber-controller gets file system events such as file create, read, and write from the file system clients and utilization, thermal, and capacity maps from the monitoring service running in the physical-system. Cyber-controller makes all the major cyber-side data placement and management decisions based on its own state, models, accumulated insights, and run-time utilization, thermal, and capacity maps from the physical-system.

The cyber-monitor garners information from the cluster and the gathered information is piggybacked on the heartbeat mechanism which is always in place in large-scale distributed file systems [84], to ensure that there is no additional performance overhead of monitoring. The cyber-monitor sends the following information to the cyber-controller:

- Thermal-map which comprises of the server outlet temperatures $\{T_1^{out}, T_2^{out}, \cdots, T_{|S|}^{out}\}$ of all the servers in the cluster.
- Thermal-map which comprises of the server inlet temperatures $\{T_1^{in}, T_2^{in}, \cdots, T_{|S|}^{in}\}$ of all the servers in the cluster.
- Thermal-map which comprises of the server temperatures $\{T_1, T_2, \cdots, T_{|S|}\}$ of all the servers in the cluster.
- Free storage capacity map which includes the details on the free space $\{\hat{\mathcal{D}}_1, \hat{\mathcal{D}}_2, \cdots, \hat{\mathcal{D}}_{|S|}\}$ available on each server in the cluster.
- Utilization map of the processors, and disks of each server: $\{\mu_1^{cpu}, \mu_2^{cpu}, \cdots, \mu_{|S|}^{cpu}\}$, and $\{\mu_1^{disk}, \mu_2^{disk}, \cdots, \mu_{|S|}^{disk}\}$

- Job count map which comprises of the count of the jobs that were performed on each server $\{J^1_{n,t}, J^2_{n,t}, \cdots, J^N_{n,t}\}$

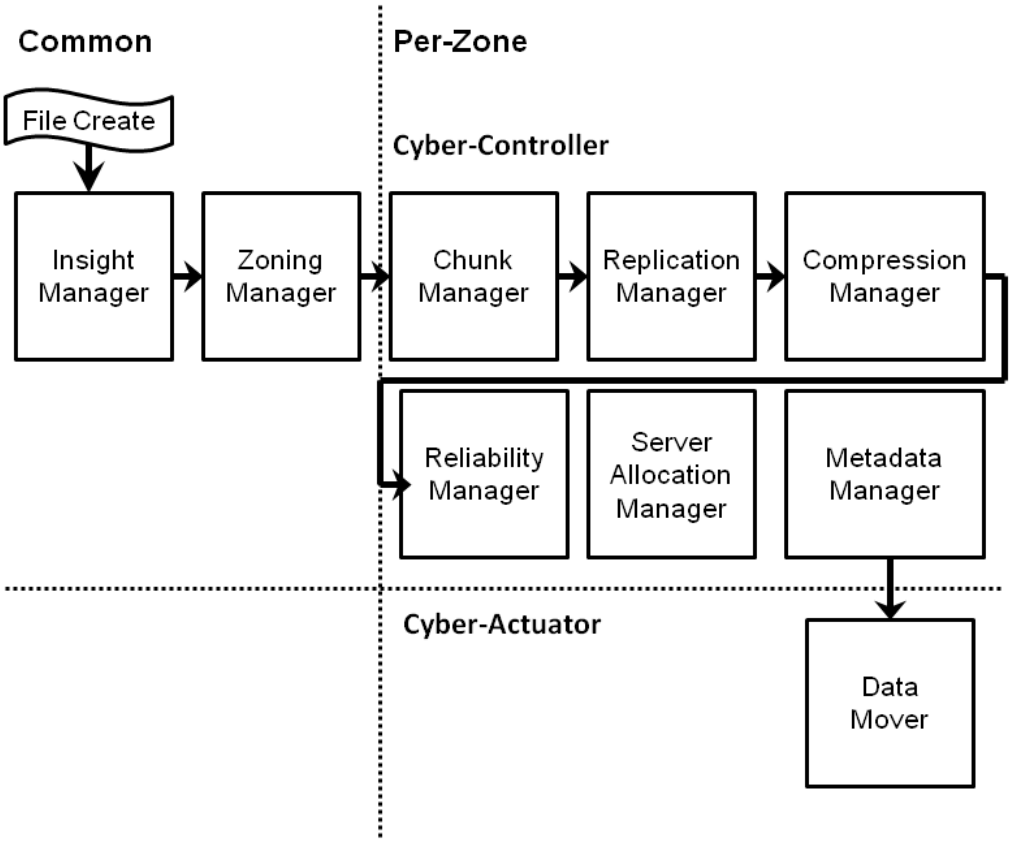

Figure 4.13: **Flow of a new file in GreenHDFS.**

The controller comprises of several managers at global and per-zone level as shown in the Figure 4.13. At file creation time, the *Insight Manager* takes as input incoming file's information such as its absolute hierarchical path, and any associated user-level hints. If the *Insight Manager* has any predictive models stored for the incoming file, then these models are used to make predictions about the file such as file's jobs arrival rate, file's size, and file's evolution life spans. If the predictive models don't exist, *Insight Manager* checks to see if any hints were given by the user and then, uses these hints to guide the rest of file-specific policies. If neither hints nor predictive models are present for the file, GreenHDFS operates in a reactive mode and relies on run-time information it receives from the cyber-monitor to guide the rest of the policies pertaining to this incoming file.

Next, the *Zoning Manager* decides the most conducive cluster zone for the incoming file based on the information it receives from the *Insight Manager*. At this point, based on the cluster zone decision for the file, the per-zone policies and managers do the rest of the incoming file's processing. Each zone is governed by a different set of policies based on the class of files residing on the zone. The incoming file is subjected to the policies of the zone it is going to be placed upon. $Active$ zone has strict SLA requirements and hence, performance is of the greatest importance. GreenHDFS trades-off power savings in interest of higher performance and no energy management is done in the servers in $Active$ zone. On the other hand, the $Inactive$ zone consists of dormant files. Since, SLA requirements for $Inactive$ zone are not strict, and the performance in this zone is not critical, GreenHDFS employs aggressive power management policies in the $Inactive$ zone and trades-off performance in interest of energy savings.

First, the *Chunk Manager* decides if the incoming file should be split into chunks for fine-grained load-balancing across the cluster, and the size and the count of the chunks if the file indeed needs to split. Then, the *Reliability Manager* decides the reliability mechanism for the file. Each zone has a different reliability mechanism based on the class of files stored on it. Next, the *Compression Manager* decides the compression mechanism for the incoming file. The decisions of the *Chunk Manager*, *Reliability Manager*, and the *Compression Manager* are sent to the *File Placement Manager* which decides the servers on which the file chunks and replicas should be placed. Finally, the *Data Mover* component in the actuator does the actual placement of the chunks and replicas on the target servers chosen by the controller for the file.

During the run-time, various policies run in the controller which are guided by the run-time information received from the monitor, insights gleaned, and any predictions or hints from the *Insight Manager*. The *File Migration Policy* runs in the $Active$ zone during periods of low load, determines the *dormancy* of the files and moves dormant, i.e., cold files to the $Inactive$ zone. The advantages of the *File Migration Policy* are two-fold: 1) leads to higher space-efficiency as precious storage space is freed up on the $Active$ zone for files which have higher SLA requirements by moving rarely accessed files out of the servers, and 2) allows significant energy-conservation by consolidating dormant files on $Inactive$ zone servers. Data-locality is an important consideration in the Map Reduce framework and computations are co-located with data. Thus, computations naturally happen on the data residing in the $Active$ zone. This results in significant idleness in all the components of the servers in the $Inactive$ zone (i.e., CPU, DRAM and Disks), allowing effective scale-down of these servers.

Scale-down of a server is initiated by the *Server Power Conservation Policy* running in the $Inactive$ zone which keeps track of the activity of the $Inactive$ zone servers. A server is scaled-down if it has been inactive for a period of time. The *File Reversal Policy* monitors the accesses to the files on the scaled-down servers, and if a dormant file starts getting accessess again, the *File Reversal Policy* moves that file from the $Inactive$ zone to the $Active$ zone so that the file accesses can avail high performance of the $Active$ zone. Next, we detail the various managers and policies in the $Active$ zone.

Table 4.3: **Table of Notations Used By Predictive Modelling**

Variable	Description	Units
f_j	File	
λ_j	Arrival rate of computational jobs targeted to a file f_j over its active lifespan	
$\mathcal{L}_{active,j}$	File lifespan between the file creation and last read access	Days
T	Training data set	
V	Validation data set	
C	Aggregated data set	
X	Independent variables	
Y	Response variables	
k	Regularization parameter	
β_i	Ridge coefficients	
S_i	Subdirectories in the file's absolute path	

4.6.1 Insights Manager

At file creation time, the *Insight Manager* takes as input incoming file's information such as its absolute hierarchical path, any associated user-level hints, file's user and owner information, and file's size. If the *Insight Manager* has any predictive models stored for the incoming file, then these models are used to make predictions about the file such as file's jobs arrival rate, size, and evolution life spans. If the predictive models don't exist, *Insight Manager* checks to see if any hints were given by the user and uses these hints to guide the rest of policies. If neither hints nor predictive models are present for the file, GreenHDFS operates in a reactive mode and relies on run-time information to guide the rest of the policies pertaining to this incoming file. We detail the predictive capability in GreenHDFS next.

To design our predictive scheme, we analyzed one-month of Hadoop Distributed File System (HDFS) audit logs and metadata checkpoints (fsimages) in a production Hadoop cluster at Yahoo!. The cluster had 2600 servers, hosted 34 million files in the namespace and the data set size was 5 petabytes. There were 425 million entries in the HDFS logs and each metadata checkpoint contained 30-40 million files.

We focused our analysis on the clickstream dataset as that dataset had a tremendous monetization value and amounted to 60% of the overall cluster data. A clickstream shows when and where a person came in to a web site, all the pages viewed, and the time spent on each page. Clickstream processing is extremely valuable in today's ecommerce corporate world. Clickstream analysis is used to predict whether a customer is likely to purchase from an e-commerce website [41], to improve customer satisfaction with the website and in effective content delivery (e.g., right bandwidth, resolution, etc.) and to assess the effectiveness of advertising on a web page or site. Internet services companies such as Facebook, Google, Twitter, LinkedIn, and Yahoo! [34] rely on clickstream processing [34, 52, 133] to calculate the web-based advertising revenues (source of majority of their income), and to derive user interest models and predictions. Huge logs of clickstreams are continuously collected and stored in flat files on the web servers. These logs are daily copied over to Map Reduce clusters and Map Reduce is used to compute various statistics and derive business insights from the data given the sheer amount of data involved [4]. Every day 60-90TB uncompressed raw log data is loaded into Hadoop at Facebook [135], 100s of GB log data is loaded at Orbitz [123], and at Europe's largest ad targeting company [72].

At Yahoo!, daily, several terabytes worth of incoming clickstream log data is stored in HDFS in a time-differentiated (one subdirectory in the path contains the daily timestamp), well-defined hierarchical file system directory structure. In order to make sense of the data, it is summarized by hours, days and weeks. HDFS directory structure is organized hierarchically by date to reflect this requirement. Daily, a fixed pipeline of production applications (e.g., advertising revenue calculation, machine learning to predict user interests, or data mining to learn trends and user behavior) is run to process, and to derive aggregate analysis from the logs. The aggregated information is again stored in a time-differentiated, well-defined hierarchical directory structure. Each file system directory is created for a specific data feed (for certain consumer applications) and all the files in the directory are used for the same use cases, and thus have the same file attributes. Also the data directory structure is relatively static, and data feeds (files) are regularly loaded/removed without changing the directory structure. The individual directories only differ in the timestamp information embedded in the directory path and rest of the path remains same. For

example, let there by a predefined directory to store the clicks that happen to an advertisement on date 08-20-2012 be denoted by /data/sds/08202012/clicks/. The directory path to store the clicks on date 08-21-2012 will be denoted by /data/sds/08212012/clicks/ in this example.

Files in such production Big Data analytics cloud are mainly used as containers for the input data that needs to be computed upon, aggregated data analytics output, and configuration information of the computational jobs. The files are only accessed by the computational jobs that are invoked on the files in a predefined manner and not by ad hoc users. We observe that the file attributes are very strongly, and statistically associated with the hierarchical directory structure in which the files are organized. This strong correlation between the directory hierarchy and file attributes is to be expected in a well-laid out and partitioned name space in log processing production environments [37,72]. Prior research has also indicated presence of statistical correlation between the filenames and file properties [55]. The presence of significant correlation between the hierarchical directory structure and file attributes leads us to design a predictor that takes the file path as an input and predicts file attributes based on the subdirectories in the file path.

The predictor component of GreenHDFS uses supervised machine learning on training data to learn the association between the hierarchical directory structure that the files are organized in and the file attributes of relevance to thermal-aware file placement, migration, and replication; namely, file's active lifespan, file's job arrival rate and file's size. The predictive models are used at the time of the file creation to predict file attributes and hence guide GreenHDFS's predictive policies. Next, we go over the supervised learning technique chosen by us to come up with the predictive models in GreenHDFS.

4.6.1.1 Predictive Analysis

To figure out the statistical correlation between the directory hierarchies and file attributes, GreenHDFS uses Ridge Regression [119], a variant of Multiple Linear Regression. Multiple Regression is a form of a supervised learning with input X (i.e., independent variables) and a response Y (i.e., dependent variable). The goal is to learn the correlation (regression) between X and Y as shown by Equations 4.29 and 4.30.

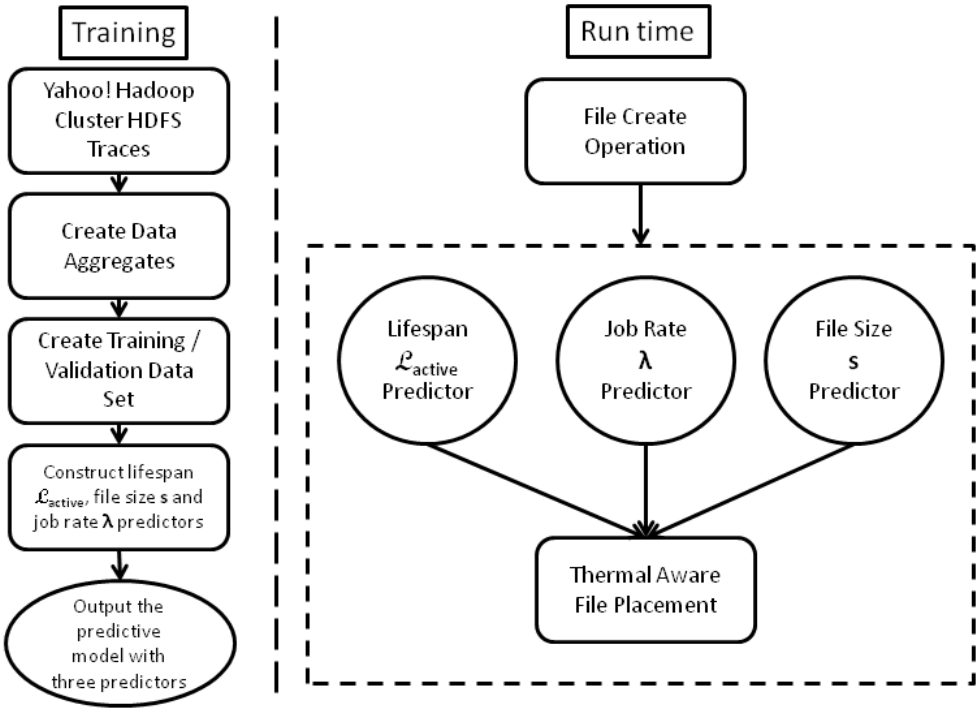

Figure 4.14: **Predictive module in GreenHDFS.**

$$E(Y|X(X_1, X_2, ..., X_p)) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon \tag{4.29}$$

Usually, β is given by:

$$\beta = [(X^T * X)]^{-1} * X^T * Y \tag{4.30}$$

However, if input values X_1,.. X_p are highly correlated, we get large errors/variance in coefficient β estimation. Ridge Regression is a variant of Multiple Linear Regression whose goal is to circumvent the problem of predictor's collinearity. An extra parameter is introduced in the model called the ridge regularization parameter k. If k is too small, Ridge Regression cannot fight

collinearity efficiently. If k is too large, the bias of β becomes too large. Therefore, there is an optimal value for k, which cannot be calculated accurately from the data only and has to be estimated by a series of trials and errors, usually resorting to cross-validation. Mathematically, the estimate of the β in Ridge Regression is given by:

$$\beta^{ridge} = [(X^T * X + k * I)]^{-1} * X^T * Y \tag{4.31}$$

Where X is the design matrix, X^T is its transpose, k is a non-negative constant called the ridge regularization parameter, I is the identity matrix, and Y is the response vector.

Based on our observation that there is a strong correlation between the directory hierarchy in which a file is organized and the file's attributes, we treat subdirectories in the training data set, denoted as T, as independent input variables X. Since, we aim to predict three file attributes: file's active lifespan $\mathcal{L}_{active,j}$, file's size and file's job arrival rate λ_j, we treat these three attributes as the response variable Y individually. First, we create an Array S of all the subdirectories present in training data set T by decomposing each absolute file path in T into its subdirectory components as shown in Algorithm 1. We ignore all the subdirectories which contain a timestamp in any form and replace the timestamp with series of asterisks. We make abstractions based on numerals, so any two directories differentiated only by timestamps are not differentiated in the feature representation. We also ignore the actual filename from the file path as majority of the files are simply named as part00, part01, $cdots$, part0N in these clusters and hence, don't add any value. Each subdirectory is then referred to by its index in Array S. We then create input Matrix X of dimensions (nxm), where n=number of file entries in T and m=number of subdirectories in Array S. Every file in the training data is represented as a binary feature vector where the index of any subdirectory present in the file path is marked as 1, and its absence is marked as 0. The binary vectors representing all the files in the training set are then inserted into the Matrix X. For example, if T contains two files, $/S_1/S_2/S_3/foo1.txt$, $/S_1/S_2/S_4/foo2.txt$, then $S = \{/S_1, /S_1/S_2, /S_1/S_2/S_3, /S_1/S_2/S_4\}$, foo1.txt will be represented as {1, 1, 1, 0} and foo2.txt as {1, 1, 0, 1}.

Ridge regression then learns the correlation between the independent variables X (i.e., subdirectories) and each of the response variables Y (file's active lifespan $\mathcal{L}_{active,j}$, file's size and file's job arrival rate λ_j) individually using Equation (4.31) during the training run and creates three separate predictive models for each of the three response variables. At a new file f_j's creation

time, the response variable (file's active lifespan $\mathcal{L}_{active,j}$, file's size and file's job arrival rate λ_j) for a new file is calculated as the sum of the coefficients corresponding to the independent variables (i.e., subdirectories) that are present in the file's absolute path as shown in Algorithm 2.

Algorithm 1 Predictor Training

{numerate all subdirectories in Training Data Set T}
{For each file in T}
for $i = 1$ to n **do**
  Decompose file path into subdirectories components $S_1, ..., S_p$
  Exclude all the Subdirectories containing time stamp information
  $S \cup \{S_1, ..., S_p\}$
end for
{For each file in T}
for $i = 1$ to n **do**
  Decompose file path into subdirectories components $S_1, ..., S_p$
  Represent file as vector F of size = size of S
  for $j = 1$ to p **do**
    if $S_j \in S$ **then**
      F.indexOf(S, S_j) = 1
    end if
    Insert F into Matrix X
    Insert value of response variable (file's active lifespan $\mathcal{L}_{active,j}$, file's size s_j, and file's job arrival rate λ_j) from T into Matrix Y
  end for
end for
{Figure out ridge regularization parameter k's value.}
while $mean\_squared\_error > expectation$ **do**
  Run validation round with the validation data set V
  Solve Equation 4.31 for Regression coefficients β^{ridge}
  Calculate predicted value of response variable using Equation 4.29
  {Choose ridge regularization parameter k's value by minimizing the mean squared error between the actual value and predicted value of the response variable}
end while
Save β^{ridge} for future predictions

Algorithm 2 Prediction

{At every new file creation time}
1: Decompose file path into subdirectories components $S_1, ..., S_p$
2: For each response variable, file's active lifespan $\mathcal{L}_{active,j}$, file's size s_j, and file's job arrival rate λ_j
3: **for** $j = 1$ to 3 **do**
4: Lookup regression coefficient from β^{ridge} for each subdirectory component
5: Calculate predicted value of response variable using Equation 4.29
6: **end for**

4.6.1.2 Training Data Preparation

We prepare the training data using one-month of HDFS audit logs and three-months of HDFS metadata checkpoints called *fsimages* of the clickstream dataset in the production Hadoop cluster at Yahoo!.

To accurately account for the file lifespan, we handle the following cases: (a) Filename reuse. We append a timestamp to each "file create" to accurately track the audit log entries following the file create entry in the audit log, (b) File renames. We use an unique id per file to accurately track its lifetime across create, rename and delete, (c) Renames and deletes at higher level in the path hierarchy are translated to leaf-level renames and deletes for our analysis, (d) HDFS logs do not have file size information and hence, we do a join of the dataset found in the HDFS logs and namespace checkpoint to get the file size information, (e) We use the creation time information from the metadata checkpoints for the files that were created earlier than the period of observation and (f) We use the last access time information from the metadata checkpoints for the files that were created close to the end of the observation period. We find that one-month look-back and look-ahead of the *fsimages* from the dates of observation were sufficient in accurately determining the file lifespans.

We wrote a pipeline of Pig [8] scripts to process the logs and the *fsimages* given the sheer size of the logs. The pipeline yields in an aggregated form the filename, file's active lifespan $\mathcal{L}_{active,j}$, file's size and file's job arrival rate λ_j for the files that were seen in the audit logs during the one-month long observation period. We then use the aggregated data set C for training and validation.

4.6.1.3 Training and Validation

The generalization ability of a predictor is best judged by using data outside of the training set. This data is called the validation set and the hypothesis that is most accurate on the validation set is the best one [22]. We split the aggregated data set C discussed above in Section 4.6.1.2, into two distinct subsets called the training data set T and validation data set V, in which $T \cap V = 0$. The training data set T had 20% of the records in C and the validation data set V had 80% of the records in C. We then calculate the mean squared error between the actual value of the file attribute (as present in an aggregated form on a per-file basis in the data sets T and V) and the predicted value in the cross-validation phase and choose the value of the regularization parameter that minimizes the mean squared error. The aggregated values of file's active lifespan $\mathcal{L}_{active,j}$, file's size f_j and file's job arrival rate λ_j in T help populate Y and allow for the calculation of β^{ridge} as shown in Equation 4.31.

4.6.1.4 Retraining

The training run needs to be repeated any time there is a substantial change in either the applications running on the data set or in the data set itself. New training data needs to be generated from the latest audit logs and Algorithm 1 needs to be rerun on the new training data set. The dataset that we considered is slow to change as each application change requires significant quality control before getting deployed on the production system given the associated monetary considerations. Hence, we envisage repeating the training only every 3-4 months. Another indicator of the need for retraining is an increase in the number of accesses to the $Inactive$ zone. Such an increase signifies growing inaccuracy in the lifespan predictor and hence, calls out for retraining.

4.6.1.5 Discussion

Predictive models are very workload- and dataset-specific. Same feature sets won't work across datasets and the feature sets that are relevant to a particular dataset need to be identified. GreenHDFS is not married to the predictive models generated for the clickstream dataset and

the predictive models in the *Insight Manager* can be replaced by the models that make sense for the dataset in question.

Predictive capability only works when the dataset and workload has some degree of predictability. If the dataset and workload are purely used in an ad hoc manner, the predictability is very low. For example, research Big Data analytics clusters are used in a purely ad hoc manner by the researchers and files are computed upon in a totally random fashion. Such a dataset can not be predicted with good accuracy atall.

4.6.2 Zoning Manager

The *Zoning Manager* decides the most conducive zone for the incoming new file. *Zoning Manager* is aware that the $Active$ and $Inactive$ zones' policies make different performance and energy savings trades-offs. $Active$ zone trades-off energy-savings in interest of high performance and hence, is the right match for files with strict service level agreements and deadlines which require performance guarantees. $Inactive$ zone trades-off performance in interest of energy savings and is a better match for files that do no have strict server level agreements or else, don't require performance guarantees. *File Migration Policy* running in $Active$ zone does keep track of the changing performance needs and evolution life spans of the files and moves files from $Active$ zone to the $Inactive$ zone everyday if they no longer need high data access performance or else have become dormant. Now, performance needs, popularity, or jobs arrival rate of the files may either reduce over time or may be low to begin with. If the later category files can be identified upfront and placed pro actively on the $Inactive$ zone, such a placement reduces the amount of data that needs to be moved by the *File Migration Policy*, and provides proactive cooling and compute energy savings.

Zoning Manager can operate in three modes: reactive, hint-driven and predictive. If no predictive models or hints exist for the incoming file, *Zoning Manager* places the file on the $Active$ zone by default. If the user has provided a hint about the file suggesting very low job arrival rate (e.g., file is of the backup or archival nature), or that the file is not subject to any strict service level agreements or deadlines, the file is placed in the $Inactive$ zone. Since, computations on these files can tolerate low performance, they can be placed on the $Inactive$ zone without any worries about the adverse performance impact. If the hint suggests heavy jobs arrival rate or strict service

level agreements, the file is placed in the $Active$ zone.

If predictive models do exist for the new incoming file, *Insight Manager* uses the predictive data models to predict file's jobs arrival rate, size, and evolution life spans. The *Zoning Manager* decides the zone assignment of the file based on the predicted values of file's jobs arrival rate and size. Files with very low predicted jobs arrival rate are marked as candidates for the aggressively power-managed $Inactive$ zone. Rest of the files are marked as candidates for one of the $Active$ zones. The prediction of the file size assists GreenHDFS in further fine-tuning the zone selection between the $Active$ zone and the $ActiveSSD$ zone as illustrated below. Such predictive zone assignment proactively yields high performance, more cooling energy savings, and reduces need for file migrations.

The specialized distributed file systems for Big Data analytics such as GFS and HDFS are designed to support large files (100MB or larger) [61, 84]. However, we observed significant heterogeneity in file sizes and popularity (heat) in the production Big Data analytics Hadoop cluster at Yahoo! as shown in Figure 4.15. Out of the 34 million files hosted in the cluster, 39% of files are smaller than 1K. Presence of small files has been noted in several other Big Data analytics clusters such as those at Google and Facebook. The small files are unavoidable as the aggregated results of the Big Data analytics jobs, which are again stored back on the cluster, may not always be big in size. In addition, files such as configuration files do tend to be small in size. This observation raises performance and file placement concerns as intermixing small files with large files is known to impact the service times of the small files and their access performance. A small file access may have to wait for a large file access which is queued ahead (head-of-line-blocking effect). Under heavy load, whereby, queuing delays dominate response time, such an intermix can have significant performance repercussion [90].

GreenHDFS places only small-sized and read-only files in $ActiveSSD$ zone as it is aware that SSDs have limited write cycles and storage capacity. This separation of small files from large files further helps improve small files' performance on the cluster. Such predictive zone placement implicitly yields (proactive) *performance guarantees* for the small files.

4.6.3 Chunk Manager

4.6.3.1 Active Zone

$Active$ zone contains files that are actively being computed upon and the Big Data analytics jobs running on these files typically have strict service level agreements and deadlines, and hence, require strong performance guarantees. Given, the data-intensive nature of these jobs, high performance of parallel data access is crucial for overall job completion times. Hence, the active files are chunked into 64 - 128 MB sized $\hat{n}_j$ number of chunks and distributed across the servers in the $Active$ zone for fine-grained load-balancing, and high parallel data access.

4.6.3.2 Inactive Zone

$Inactive$ zone trades-off performance in interest of high energy savings. Now, the longer the servers are scaled-down, the higher the energy savings. Long periods of scale-down can be achieved by reducing the need to wake-up the scale-down servers. Reducing server wake-ups is also crucial for server hardware reliability as some components such as the disks have limited start/stop cycles. Keeping this rationale in mind and recognizing the low performance needs and infrequency of data accesses to the $Inactive$ zone; GreenHDFS does not chunk the files in the $Inactive$ zone and a file is placed in entirety on a server. Such a placement ensures that only one server is sufficient to absorb the file and hence, only one server needs to be in an active power state. Had the file been chunked, $\hat{n}_j$ number of servers would need to be in an active power state to absorb the incoming file. The non-chunked placement also helps at the time when a file in the $Inactive$ zone is accessed again for some reason. Only one server hosting the file needs to be woken up in this case. Instead, had the file been chunked and if the number of chunks $\hat{n}_j$ was almost equivalent to the number of servers N in the $Inactive$ zone–a highly probable scenario given the huge sizes of some of the files in the cluster–the entire $Inactive$ zone would need to be woken up to process the file access. Thus, not chunking the files in the $Inactive$ zone helps elongate the idleness period, allowing longer period of scale-down and increasing energy savings. Now, if a file residing in the $Inactive$ zone does need to get accessed in the future for things like historical trend analysis, the file access will incur degraded performance courtesy of not load-balancing the file across servers. GreenHDFS uses *File Reversal Policy* covered in Section

4.6.9 to handle this scenario.

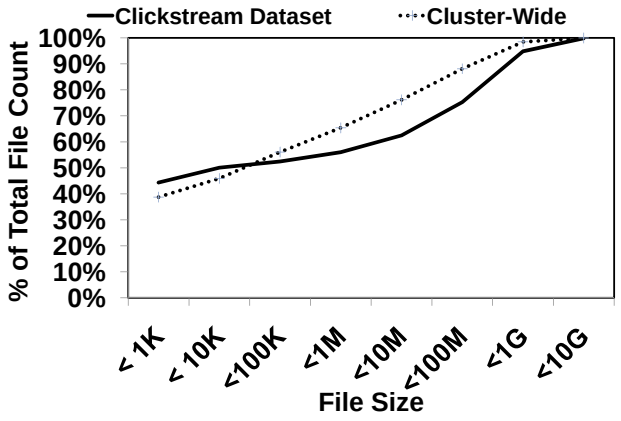

Figure 4.15: File Size Distribution.

4.6.4 Reliability Manager

4.6.4.1 Active Zone

Big Data analytics clusters are build on top of commodity hardware where failure is a norm. Hence, fault-tolerance and resilience are of utmost importance. For file resiliency and fault-tolerance, three-way replication is the norm in the production Big Data analytics cluster and $Active$ zone does the same to ensure resiliency. Replication serves another purpose also as it allows load-balancing of the performance hot spots in the cluster; thereby, results in better performance.

Now, performance-driven replication is done in two ways in GreenHDFS: predictive and reactive. In the reactive case, the GreenHDFS keeps track of the jobs targeted to the files in the cluster. The files that are getting high share of jobs are deemed hot by the reactive system and are then, replicated across the cluster to alleviate performance hot spots created by such hot files. In case predictive models are available for the incoming file, instead of relying on the administrator

or the user to define the replication factor, or relying on reactive replication techniques [136], GreenHDFS decides the replication factor for a file proactively at the time of file creation based on the file's predicted heat. GreenHDFS is based on the belief that the true definition of a file's heat from the perspective of replication is given by the ratio of total number of jobs to a file and the file's hot lifespan $\mathcal{L}_{active,j}$. The ratio helps distinguish between files that receive a large number of jobs in a short lifespan, from long-living files that accumulate a large number of job count over their lifetime; however, receive only few jobs on a daily basis. A file with a high heat

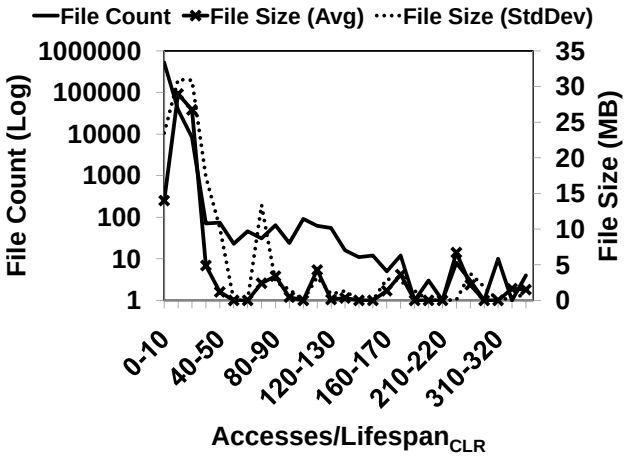

Figure 4.16: **File heat (jobs/$\mathcal{L}_{active,j}$) distribution in a production Big Data analytics cluster at Yahoo!, where $\mathcal{L}_{active,j}$ is in days.**

is proactively replicated with a higher replication factor proportionate to the file's heat. Such proactive replication results in upfront performance guarantees by proactively making sure that the performance hot spots don't arise in the first-place. Now, such performance-driven replication may turn into a real storage capacity hog if the hot files are very large in size. Fortunately, as shown in Figure 4.16, the good thing is that the really hot files are small-sized; hence, the additional replicas will not adversely affect the used capacity in the cluster. However, for a performance-driven replication scheme to not result in using too much storage capacity, it is equally important to reclaim the replicas once the files are no longer receiving as many jobs and are no longer as hot. In the cluster under analysis and evaluation, the active life span of the files, $\mathcal{L}_{active,j}$ is of short duration and hence, it is more important to garbage collect the replicas at the end of $\mathcal{L}_{active,j}$. GreenHDFS automatically garbage collects the replicas of the files at the end of their predicted $\mathcal{L}_{active,j}$ by keeping track of the predicted value of $\mathcal{L}_{active,j}$. This results in a

proactive and *self-adaptive* replication and load-balancing.

4.6.4.2 Inactive Zone

$Inactive$ zone can either use three-way replication which is the norm for providing fault-tolerance and resilience or RAID-6. RAID-6 has a better fault-tolerance than RAID-5 and can tolerate simultaneous failure of two disks. Performance of RAID-6 is slightly lower than RAID-5 overall; however, performance is not a concern in the $Inactive$ zone in the first place. The storage overhead of RAID-6 is a factor of 1.2 as opposed to a factor of 3 with three-way replication. Reducing the storage requirement of the $Inactive$ zone servers aids in reducing the server footprint of the $Inactive$, which is important in allowing a higher number of servers to get allocated to the $Active$ zone for high data access performance.

GreenHDFS uses software-based RAID-6 for multiple reasons: a) *Cost*, hardware-based RAID controllers are expensive and add to the total cost of ownership of a data center. On the other hand, there is zero-capital cost involved in software-based RAID, b) *Higher performance* as the RAID parity computations will happen in the host CPU and given the multi-core CPU trend in servers, speed of computation is bound to be better than that of a RAID controller. c) *Easily available* RAID-6 is already supported in Linux and hence, there is no need to write software from scratch.

4.6.5 Compression Manager

Files are typically compressed and then placed on the Big Data analytics clusters because of the sheer size of the majority of the files. Since, the files in the $Inactive$ zone are computed upon very infrequently, it doesn't need a very high decompression throughput. In addition, the compression throughput also doesn't need to be high as the files are placed on the $Inactive$ zone during periods of low load by the *File Migration Policy* and hence, are outside of the critical data access path. Hence, the most important characteristic for a compression algorithm in the $Inactive$ zone is the compression ratio of a compression algorithm. Higher compression ratio translates into higher space-efficiency in the $Inactive$ zone and further, helps reduce its server footprint. The lower the server footprint of the $Inactive$ zone, higher can be the server footprint

of the $Active$ zone–something, very important for ensuring high throughput and performance. Following compression algorithms were used in the analysis with varying levels of compression [14]:

- LZO (compression levels 1, 3, 6, 7, 8, 9)
- GZIP (compression levels 1, 3, 4, 5, 6, 9)
- BZIP (compression levels 1, 3, 5, 7, 9)
- LZMA (compression levels 1, 2, 3, 5, 7, 9)

The results of experiments with compression algorithms on representative data samples from production hadoop cluster at Yahoo! are shown in Table 4.4. The compression phase consisted of reading the raw data in the dataset, and writing the compressed data to the disk. The decompression phase involved reading the compressed data from the disk and writing the decompressed data to /dev/null. As shown in Table 4.4 the compression and decompression throughput of lzma

Table 4.4: **Comparison of Compression Algorithms**

algorithm	**ratio**	**comp(mb/s)**	**decomp(mb/s)**
lzo.1	3.37	29.18	168.15
lzo.3	3.38	28.95	166.45
lzo.6	3.38	29	166.03
lzo.7	4.31	6.81	157.01
lzo.8	4.33	4.6	198.38
gzip.3	4.3	24.46	87.14
gzip.4	4.7	20.04	87.67
gzip.5	4.9	18.16	91.58
gzip.6	4.99	18.34	94.18
gzip.9	5.05	14.47	94.72
bzip.1	4.81	4.73	20.77
bzip.3	5.77	4.35	22.49
bzip.5	6.2	3.94	22.61
bzip.7	6.48	3.6	21.04
bzip.9	6.68	3.34	19.96
lzma.1	5.89	7.17	32.65
lzma.2	6.65	3.42	36.22
lzma.3	7.47	1.1	38.86
lzma.5	8.16	0.57	38.16
lzma.7	8.76	0.44	38.37
lzma.9	9.49	0.27	38.61

algorithm at compression level 9 (lzma.9) is the lowest; however, the compression ratio of lzma.9 is the highest. Hence, GreenHDFS opts for lzma.9 in compressing the data in the $Inactive$ zone to enhance the storage-efficiency of the $Inactive$ zone and further, reduce the server footprint of the $Inactive$ zone.

4.6.6 File Placement Manager

The *File Placement Manager* tackles the problem of placing the files on the servers in the cluster. The *File Placement Manager* are guided by very different policies in the $Active$ and the $Inactive$ zones as illustrated below:

4.6.6.1 Active Zone

GreenHDFS divides the incoming file into multiple file chunks and then places these chunks into many servers distributed across the $Active$ zone for fine-grained load-balancing and high parallel data access performance. The *File Placement Manager* in the $Active$ zone maps the file chunks and replicas to the servers using a mapping function π which is described next.

Let f_j^k = set of replicas of chunk k of file j, $1 \leq k \leq \hat{n}_j$, $1 \leq j \leq |Z|$ where $|Z|$ is the number of files in the cluster and $\hat{n}_j$ is the number of chunks into which file j is divided. $f_j^k = \{f_{j,1}^k$, $f_{j,2}^k$, $f_{j,3}^k\}$, where $f_{j,n}^k$ are the three replicas of the file chunk. Denote $\tilde{f}_j = \{f_j^1, f_j^2, \cdots, f_j^{\hat{n}_j}\}$ to be the set of all chunks of file j and $Z = \{\tilde{f}_1, \tilde{f}_2, \cdots, \tilde{f_{|Z|}}\}$ to be the set of all the file chunks in the cluster.

Let $c_{j,i}^k$ be the replica of chunk k of file j assigned to server i. Let $C_{j,i}$ be the set of replicas of file j's chunks assigned to server i, i.e., $C_{j,i} = c_{j,i}^1 \cup c_{j,i}^2 \cup \cdots \cup c_{j,i}^{n_{j,i}}$, where $n_{j,i}$ is the number of file $\tilde{f}_j$'s chunks assigned to server i. Let $\tilde{C}_i$ be the set of chunk replicas of different files assigned to server i. $\tilde{C}_i$ is equal to a set of the set of file chunks assigned to server i, i.e., $\tilde{C}_i = C_{1,i} \cup C_{2,i} \cup \cdots \cup C_{N_i,i}$, where N_i is the number of the files whose chunks are assigned to server i. Let $\ddot{C} = \{\tilde{C}_1, \tilde{C}_2, \cdots, \tilde{C}_{\ddot{N}}\}$, where $\ddot{N}$ = number of servers in the cluster with chunks assigned to them. $\ddot{N}$ is usually equal to the total number of the servers S in the cluster.

The mapping function π is defined as $\pi : Z \rightarrow \ddot{C} \times S$ which takes the set of all file chunks Z as its input and produces an ordered set $\ddot{C}$ as output. S is the set of servers in the cluster, $|S|$ is the number of servers in the cluster, $S = \{S_1, S_2, \cdots, S_{|S|}\}$ or $S = \{1, 2, \cdots, |S|\}$.

The $Active$ zone's *File Placement Manager* uses a thermal-aware variant of the mapping function π. The thermal-aware, proactive file placement of GreenHDFS allows more uniform thermal-profile, lower overall cluster temperature, and thereby, lower cooling costs as is discussed in detail in Chapter 5.

4.6.6.2 Inactive Zone

The $Inactive$ zone's *File Placement Manager* is driven by a goal to maximize the energy savings. It seeks to place a file $\tilde{f_j}$ in its entirety, onto a server i in the $Inactive$ zone. By default, the servers in $Inactive$ zone are in an inactive power state. A server is woken up when either new file needs to be placed on it or when a file already residing on the server is accessed. The server then stays on till the *Server Power Conserver* policy kicks in and transitions the server to inactive power state. SAM tries to avoid powering-on a server and maximizes the use of the existing powered-on servers in its server allocation decisions in interest of maximizing the energy savings. GreenHDFS uses an in-order placement policy. A data structure maintains a sorted list of all the server IDs and the first server in the data structure is chosen as a target for the dormant files being moved by the File Migration Policy. This server is filled completely to its capacity and is then scaled-down. At this point, just prior to the *File Migration Policy* run, next server in the sorted list is woken-up and is used as the target for the dormant data. In the cluster-level zone partitioning scheme covered in Section 4.5.2.1, the in-order placement and sorted list are maintained on a cluster-wide basis, and in the rack-level zone partitioning scheme covered in Section 4.5.2.2, the in-order placement and sorted list are maintained on a per-rack basis. There is a temporal locality in the accesses and jobs that happen on the files, and the files that enter the dormant evolution phase at the same time, typically receive future accesses at the same time. Putting the files that get dormant on the same day on the same server, limits any future accesses to these files to one server and hence, significantly cuts down on the server wake-ups needed. This results in higher energy costs savings as it allows servers to enjoy longer periods of scale-down.

4.6.7 File Migration Policy

The *File Migration Policy* runs in the $Active$ zone during periods of low load and migrates dormant files from the $Active$ zone to the $Inactive$ zone. The advantages of this policy are two-fold: 1) leads to higher space-efficiency as space is freed up on the $Active$ zone for files which have higher SLA requirements by moving rarely accessed files out of the servers in these zones, and 2) allows significant energy-conservation. Data-locality is an important consideration in the Map Reduce framework and computations are co-located with data. Thus, computations naturally happen on the data residing in the $Active$ zone. This results in significant idleness in all the components of the servers in the $Inactive$ zone (i.e., CPU, DRAM and Disks), allowing effective scale-down of these servers. The policy comes in two flavors: 1) predictive - if the workload running on GreenHDFS is predictable, and GreenHDFS has predictive models generated for the workload, predictive version of policy is used which is guided by the predictions generated by the *Insight Manager* in conjunction with some run-time information, and 2) reactive - if the workload is not predictable at all, GreenHDFS relies on insights gleaned at run-time to guide the policy.

4.6.7.1 Reactive File Migration Policy

Files in the cluster can be in different evolution phases: some files may be in their active phase; whereby, the files are actively being computed upon and accessed, and other files may be in their dormant phase; whereby, the files are past their active phase and are now lying dormant in the system without receiving any computations or accesses. The *File Migration Policy* runs in the $Active$ zone, monitors the *dormancy* of the files as shown in Algorithm 3 and moves dormant, i.e., cold files to the $Inactive$ zone. The cyber-controller keeps track of the last access time of all the files in the cluster and if the last access time of a file is greater than the threshold value t_{FMP}, the file is deemed to have transitioned into a dormant evolution phase, is classified as a dormant file, and is moved to the $Inactive$ zone. The metadata information of the file is removed from the $Active$ zone and is added to the $Inactive$ zone. The value of threshold t_{FMP} is workload dependent and requires an analysis of the evolution life spans of the files in the workload.

A good *File Migration Policy* should result in maximal energy savings, minimal data oscillations between GreenHDFS zones, maximal and fast active space reclamation, and minimal performance

Algorithm 3 File Migration Policy

{For every file j in $Active$ zone}
for $j = 1$ to n **do**
  $t_{dormancy,j} \Leftarrow t_{current} - t_{lastaccess,j}$
  if $t_{dormancy,j} \geq \mathrm{t}_{FMP}$ **then**
    {File j is added to the $Inactive$ zone}
    {$Inactive$ zone} $\Leftarrow$ {$Inactive$ zone} $\cup$ {j}
    {File j is removed from the $Active$ zone}
    {$Active$ zone} $\Leftarrow$ {$Active$ zone} / {j}
  end if
end for

degradation. Minimization of the accesses to the $Inactive$ zone files results in maximal energy savings and minimal performance impact. The more accurate the policy is in determining truly dormant files, less are the number of jobs/accesses to the files residing in the $Inactive$ zone. Reduction in the accesses to the $Inactive$ zone increases the periods of idleness in the $Inactive$ zone servers, allowing them to be scaled-down for a longer period of time, resulting in higher energy savings. The accuracy of the policy directly depends on the choice of the policy threshold t_{FMP} which should be chosen in a way that *minimizes* the number of accesses to the files residing in the $Inactive$ zone while *maximizing* the movement of the dormant files to the $Inactive$ zone. Maximizing the movement of the dormant files to the $Inactive$ zone is important for the success of active file allocation on the outer disk cylinder zones of disks with zoned bit recording in $Active$ zone as covered in the Section 4.5.1. The faster the files are moved out of the $Active$ zone upon entering their dormant evolution phase, the higher is the space left on the outer disk cylinder zones for hosting new incoming files and active files. Low (i.e., aggressive) value of t_{FMP} results in an ultra-greedy selection of files as potential candidates for migration to the $Inactive$ zone. While there are several advantages of an aggressive t_{FMP} such as higher space-savings in the $Active$ zone, there are disadvantages as well. If files have intermittent periods of dormancy, the files may incorrectly get labeled as dormant, and get moved to the $Inactive$ zone as illustrated in the example in Figure 4.17. At time t=5, the difference between the last access time and the current time is greater than the threshold t_{FMP}'s value for the file shown in the example in Figure 4.17. The Algorithm 3 incorrectly labels this file as a dormant file at time t=5 as it has no way to foretell the future, or to realize that the file is still in its active life span, is just enjoying a brief period of dormancy, and is soon going to get accessed again. Based on the dormant verdict of Algorithm 3, *File Migration Policy* moves the file to the $Inactive$ zone at t=5. Shortly after its move to the $Inactive$ zone, this file starts receiving file accesses again. Now, an access to a

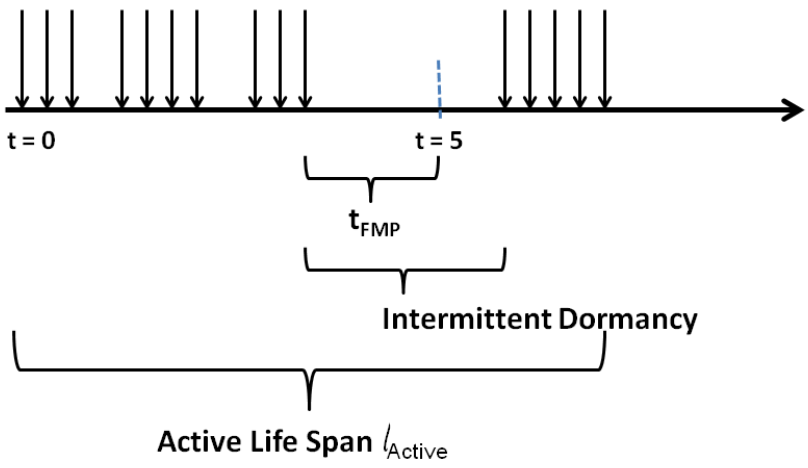

Figure 4.17: **Illustration of the importance of threshold t_{FMP}'s selection.**

file in the $Inactive$ zone may result in server wake-up of the server hosting the file, which comes with an energy expenditure and a performance penalty as some components such as the disks take as high as ten seconds to transition from inactive-to-active power state. The file access will suffer an additional performance degradation courtesy of not chunking a file in the $Inactive$ zone; placing a file on its entirety on a server doesn't allow parallel inter-file access to the file, resulting in low data access performance. Also, the access to the file in the $Inactive$ zone lowers the server energy costs savings as it results in a server wake-up.

A higher value of t_{FMP} results in a higher accuracy in determining the truly dormant files. Hence, the number of file reversals, server wake ups, and associated performance degradation decreases as the threshold is increased in duration. On the other hand, higher value of t_{FMP} means that files will be chosen as candidates for migration only after they have been lying dormant in the system for a long period of time. This would be an overkill for files with very short $\mathcal{L}_{active,j}$ (hotness lifespan) as such files will unnecessarily lie dormant in the system, occupying precious $Active$ zone capacity for a longer period of time. If less number of dormant files are moved out of the $Active$ zone, lesser space is left on the outer disk cylinder zones for hosting new incoming files. Less available space in the outermost cylinder zones complicates $Active$ zone's active and new file layout covered in Section 4.5.1. If the dormant files don't get moved to the $Inactive$ zone automatically by the *File Migration Policy*, file reorganization will need to take place in the

outer disk cylinder zone to create space for the new incoming files. GreenHDFS makes every attempt to prevent file reorganization of the active files.

4.6.7.2 Predictive File Migration Policy

A good energy-aware file migration policy in GreenHDFS should result in maximal energy savings, maximal movement of dormant data to the $Inactive$ zone, and minimal performance degradation. Minimization of the accesses to the files on the $Inactive$ zone servers translates into longer periods of idleness and few server wake ups resulting in high energy savings, and reduces the performance impact of the energy management. Thus, policy decisions need to be made in such a way that they *minimize* the number of accesses to the files residing in the $Inactive$ zone.

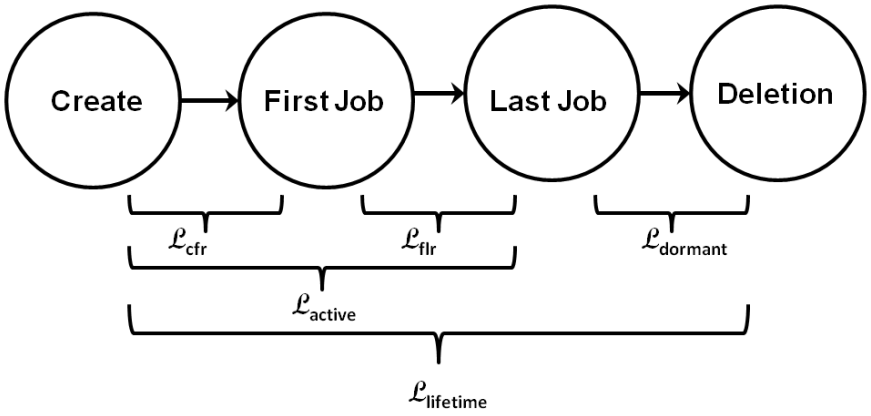

Figure 4.18: **Stages in a File's Evolution.**

A file goes through several stages in its lifetime as shown in Figure 4.18: 1) file creation is the time when the file is created and is the only time (other than file appends) that the file has data written to it, 2) *active* period during which the file is frequently computed upon and thereby, frequently accessed, 3) *dormant* period during which the file is not computed upon and thereby, is rarely accessed, and 4) *deletion* at which point the file's life time gets over and the file is deleted from the system. $\mathcal{L}_{active,j}$ metric is defined as the lifespan between the file's creation and it's last computational job. This metric is used to determine the *active* profile of each file. $\mathcal{L}_{dormant,j}$

metric is defined as the lifespan between the file's last job and it's deletion. This metric helps in determining the *dormancy* profile of the files as this is the period for which files are dormant in the system. The longer the dormancy period, higher the suitability of the file for migration to the $Inactive$ zone.

File migration can be done in a very fine-grained, and per-file basis if the lifespans of the various stages in the evolution of a file can be predicted at the time of the file creation. A per-file, finely-tuned file migration policy that can predict the $\mathcal{L}_{active,j}$ of file and proactively migrate the file to the $Inactive$ zone at the end of the $\mathcal{L}_{active,j}$ will result in a much higher accuracy in determining the truly cold (i.e., dormant) files and thus reduces the accesses to the files that have been migrated to the $Inactive$ zone. Furthermore, by moving a file to the $Inactive$ zone on the same day the file enters its dormancy evolution phase, allows fast reclamation of the storage space in the outer cylinder zones, leaving more space available for the new incoming and active files.

There are typically two classes of dormant files on the production clusters: 1) files that need to be stored for some retention period because of compliance or regulatory reasons, and 2) files that are not subject to compliance regulations and are deleted soon after their hotness period is over. There is little value in moving the second class of dormant files that get deleted soon after their $\mathcal{L}_{active,j}$ is over to the $Inactive$ zone. $\mathcal{L}_{dormant,j}$ is useful to consider here as it gives an idea of the dormancy lifespan of a file. A file with a very short $\mathcal{L}_{dormant,j}$ (e.g., 1 day) is not a viable candidate for migration and should be ignored.

First of all, predictive models are generated using Algorithm 1 as shown in Section 4.6.1.1. Algorithm 2 is then used to predict the $\mathcal{L}_{active,j}$ and $\mathcal{L}_{dormant,j}$ of a new file at the file creation time. The $Migration_{time}$ (in days) for a file is computed using the predicted value of $\mathcal{L}_{active,j}$ and a pointer to the file is inserted in a data structure called *MigrateInfo* as shown in Algorithm 4. *MigrateInfo* maintains a list of files indexed by their $Migration_{time}$. Every day, the *File Migration Policy* runs at periods of low load and enumerates all the files that have a $Migration_{time}$ value corresponding to the current day. These files are migrated to the $Inactive$ zone as shown in Algorithm 5. A file with $\mathcal{L}_{dormant,j}$ of less than two days is not considered a viable candidate for migration and is ignored in the migration process.

Summarizing, there are several advantages to the predictive, per-file granularity file migration policy in comparison to the reactive file migration policy: 1) high accuracy in identifying truly dormant files and considerably *reducing accesses* to the $Inactive$ zone. This in turn results in:

Algorithm 4 Migration Time Calculation

{At every new file j's creation, the file's $\mathcal{L}_{active,j}$ and $\mathcal{L}_{dormant,j}$ are predicted using Algorithm 2 and anticipated $t_{migrate}$ of the file is calculated based on the predicted value of $\mathcal{L}_{active,j}$ and inserted into cluster-wide *MigrateInfo* data structure.}
if $\mathcal{L}_{dormant,j} > 2$ **then**
  $t_{migrate,j} = t_{current} + \mathcal{L}_{active,j} + 1$
  INSERT $(t_{migrate}, j)$ INTO *MigrateInfo*
end if

Algorithm 5 Predictive File Migration Policy

{At policy run-time, lookup *MigrateInfo* which is populated by Algorithm 4 at every new file creation.}
{*Migration List*} $= Lookup($*MigrateInfo*$, t_{current})$
{For every file j in Migration List which has $t_{migrate} == t_{current}$}
for $j = 1$ to n **do**
  {File is migrated to the $Inactive$ zone}
  {$Inactive$ *zone*} $\Leftarrow$ {$Inactive$ *zone*} $\cup$ {j}
  {File is removed from the $Active$ zone}
  {*Active zone*} $\Leftarrow$ {*Active zone*} / {j}
end for

a) higher amount of idleness in the $Inactive$ zone leading to higher energy savings, b) none or significantly reduced performance impact (caused by accesses to the $Inactive$ zone), and c) higher reclamation of the hot space given the fine-grained migration of dormant files which is very important for the feasibility of the high performance active data placement covered in Section 4.5.1; 2) *higher scalability* because instead of enumerating the entire file name space for identifying dormant files, GreenHDFS just needs to look at the *MigrateInfo* data structure for the list of migration file candidates on the day of the policy run. The candidate migration files on any day are a very small fraction of the total name space, and 3) *low runtime overhead* as there is no need to record the last access time of the files.

4.6.8 Server Power Conserver Policy

The *Server Power Conserver Policy* runs in the $Inactive$ zone and determines the servers which can be transitioned into a power saving standby/sleep mode in the $Inactive$ zone. As discussed in Section 2.1.5, power management of any one component is no longer sufficient in commodity

servers and a server-wide energy management approach needs to be used. GreenHDFS leverages energy cost savings at the entire server granularity (CPU, Disks, and DRAM) by scaling-down the servers in the $Inactive$ zone. The cyber-monitor keeps track of the jobs running and file accesses to all the servers in the $Inactive$ zone. If a server hasn't been received any jobs or accesses in the last threshold t_{SPC} number of days, the server is scaled-down by the *Server Power Conserver Policy* as shown in Algorithm 6.

A high value of t_{SPC} increases the number of the days the servers in the $Inactive$ zone remain in active power state and hence, lowers the energy savings. On the other hand, a high value of t_{SPC} helps amortize power state transition latency in case of Reactive GreenHDFS. If the Reactive GreenHDFS is inaccurate in its dormancy determination of a file as illustrated in the example in Figure 4.17, the file gets accesses soon after it is moved to the $Inactive$ zone. With a high value of t_{SPC}, it is possible that the server hosting the file has still not been transitioned to an inactive state, obviating the need to do a power state transition which results in improved performance of the accesses to the $Inactive$ zone. Thus, a trade-off needs to be made between energy-conservation and data access performance in the selection of the value for t_{SPC} in Reactive GreenHDFS. No such trade-off needs to be made in case of Predictive GreenHDFS given its high accuracy in dormancy determination and t_{SPC} can be as low as one day.

A scaled-down server is transitioned back automatically to an active power state upon the receipt of the triggering events: 1) a file placed on the server needs to be accessed, 2) bit rot integrity checker needs to run on the disks of the server ensure there is no data corruption, 3) file needs to be deleted from the server, and 4) compute capacity of the server needs to be tapped if the load on the system is higher than the compute capacity of the $Active$ zone servers (e.g., peak load has arrived).

The scale-down mechanism for each server component is discussed next:

4.6.8.1 Disks

Disk drives consume significant amount of power simply to keep the platters spinning, possibly as much as 70% of their total power for high RPM drives [65]. GreenHDFS uses either the standby or the sleep power states of the disks while scaling a server down; both these states put the spindle at rest and thereby, consume almost negligible power compared to the idle power state

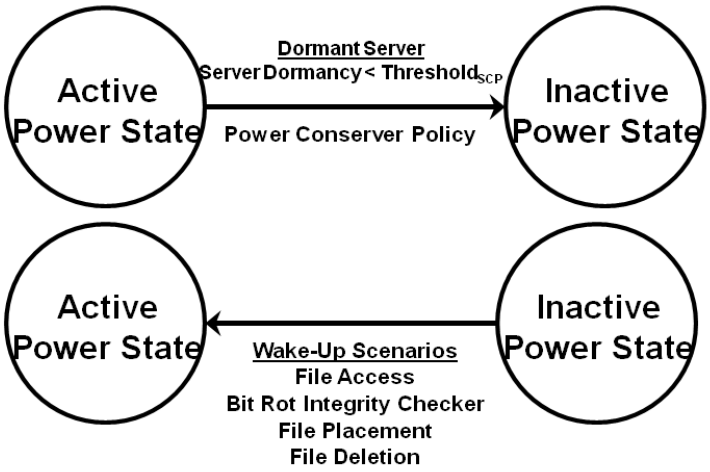

Figure 4.19: **Triggering events leading to Power State Transitions in the** $Inactive$ **zone**

whereby the platters are still kept spinning.

The disk drive enters standby power state when the host sends a standby command. If the host has set a standby timer, the drive can also enter standby mode automatically after the drive has been inactive for a specifiable length of time. In standby mode, the drive buffer is enabled, the heads are parked and the spindle is at rest. The drive accepts all commands and returns to active mode any time disk access is necessary.

The drive enters sleep power state after receiving a sleep command from the host. In sleep power state, the drive buffer is disabled, the heads are parked and the spindle is at rest. The drive leaves sleep mode after it receives a hard reset or soft reset from the host. After receiving a reset, the drive exits sleep mode and enters standby mode with all current translation parameters intact.

4.6.8.2 Processor

Advanced Configuration and Power Interface (ACPI) is an open industry standard for doing OS-level power management. GreenHDFS uses "Sleep" state such as S3 state defined by the ACPI standard to transition a processor to a sleep power state. In case S3 state is unavailable for a

processor, clock gating can provide substantial energy savings. For example, Intel's Xeon 5400-series processor's power requirements drop from 80W to 16W upon executing a halt instruction. From this state, resuming execution requires only a nano-second delay. The Intel Atom Z5xx series processor has sleep state C4 whereby the processor maintains its context, phase-locked loop (PLL), and all the internal clocks are stopped. In addition, Intel Atom also supports deep and deeper sleep states which are capable of saving even more power such as the C6 state. The inactive-to-active transition latency is ten microseconds for C4 state and 100 microseconds for the C6 state.

4.6.8.3 DRAM

DRAM is very power-hungry when active. Several recent DRAM specifications feature an operating mode, called self-refresh, where the DRAM is isolated from the memory controller and autonomously refreshes DRAM content. In this mode, the memory bus clock and PLLs are disabled, as is most of the DRAM interface circuitry. Self-refresh saves more than an order of magnitude of power. For example, a DRAM DIMM consumes only 0.2 watts of power during self-refresh as opposed to consuming 3.5-5 watts in active power state. Transitions into and out of self-refresh can be completed in less than a microsecond.

4.6.8.4 Network

GreenHDFS requires that it should be able to automatically wake-up servers when triggering events shown in Figure 4.19 do happen. To that effect, GreenHDFS requires that the network interface card (NIC) should be able to wake the system upon arrival of a special network packet. Existing NICs already provide support for Wake-on-LAN to perform this function and consume only 400mW while in this mode. GreenHDFS uses Wake-on-LAN in the NICs to send a magic packet to transition a server back to an active power state if a file residing on the $Inactive$ zone needs to be accessed. GreenHDFS also wakes up a server occasionally if bit rot checker needs to run, or files need to be deleted on the servers in the $Inactive$ zone.

Algorithm 6 Server Power Conserver Policy

{For every server i in $Inactive$ zone}
for $i = 1$ to n **do**
  $t_{dormancy,i} \Leftarrow t_{current}$ - $t_{lastaccess,i}$
  if $t_{dormancy,i} \geq t_{SPC}$ **then**
    $i \Leftarrow$ INACTIVE_POWER_STATE
  end if
end for

4.6.9 File Reversal Policy

The File Reversal Policy runs in the $Inactive$ zone and ensures that the QoS, bandwidth and response time of files that becomes popular again after a period of dormancy is not impacted. As discussed earlier, the *Chunk Manager* for the $Inactive$ zone doesn't split a file into chunks and a file is placed in its entirety on a server. Since, the Big Data analytics jobs require superior inter-file data access parallelism, the performance of a job targeted to a file residing on the $Inactive$ zone significantly goes down in absence of file chunking and striping across servers. The File Reversal Policy monitors the accesses to the files residing on the $Inactive$ zone and if the number of accesses to a file that is residing in the $Inactive$ zone becomes higher than the threshold t_{FRP}, the file is moved back to the $Active$ zone as shown in Algorithm 7 in interest of performance. The file is then chunked and placed across the servers in the $Active$ zone for fine-grained load-balancing and high inter-file parallelism in congruence with the high-performance policies in the $Active$ zone.

Algorithm 7 File Reversal Policy

{For every file j in $Inactive$ zone}
for $j = 1$ to n **do**
  if *num_accesses*$_j \geq t_{FRP}$ **then**
    {File j is added to the $Active$ Zone}
    {$Active$ zone} $\Leftarrow$ {$Active$ zone} $\cup$ {j}
    {File j is removed from the *InActive* Zone}
    {$Inactive$ zone} $\Leftarrow$ {$Inactive$ zone} / {j}
  end if
end for

A higher value of t_{FRP} ensures that files are accurately classified as *hot-again* files before they are moved back to the $Active$ zone from the $Inactive$ zone. This reduces data oscillations in the system and unnecessary file reversals. However, a higher value of t_{FRP} increases the performance

impact on the response time which is inherent till the file resides in the $Inactive$ zone.

4.7 Physical-Side

The physical-side in the cyber-physical GreenHDFS consists of a monitor, controller, and an actuator. The physical-monitor sends the following information to the physical-controller:

- Overall cluster temperature T_c^{out}
- Thermal-map which comprises of the server outlet temperatures $\{T_1^{out}, T_2^{out}, \cdots, T_{|S|}^{out}\}$ of all the servers in the cluster.
- Thermal-map which comprises of the server inlet temperatures $\{T_1^{in}, T_2^{in}, \cdots, T_{|S|}^{in}\}$ of all the servers in the cluster.

The efficiency of the cooling unit, i.e., the computer room air conditioner (CRAC) is characterized by its coefficient of performance (COP). COP is defined as the ratio of the amount of heat *Q* removed by the cooling device to the energy *W* consumed by the cooling device. Thus, work required to remove heat is inversely proportional to the COP.

The cooling power consumption is calculated as shown below [114]:

$$P_{c,t} = \frac{c_{air} \cdot f_{ac} \cdot \left(T_{c,t}^{out} - T_{ac} \right)}{COP}, \tag{4.32}$$

where, c_{air} is the heat capacity constant, and f_{ac} is the flow rate of the cooling unit. T_c^{out} is the temperature of the hot exhausted air that needs to be cooled by the CRAC, and T_{ac} is the set point temperature of the cold air supplied by the CRAC. A typical COP model obtained from a Liebert CRAC unit [29]:

$$COP\left(T_{out}\right) = 0.0068 \cdot (T_{ac})^2 + 0.0008 \cdot T_{ac} + 0.458; \tag{4.33}$$

The physical-actuator controls the cooling unit's air supply temperature T_{ac} according to the overall temperature T_c^{out} observed in the physical-system. If the overall temperature T_c^{out} in the

physical-system has cooled down below a threshold temperature, the physical-actuator increases T_{ac} while ensuring that the inlet temperatures of the servers remain below the redline temperature (as that would pose a risk to the reliability). Increasing T_{ac} increases the efficiency of the cooling unit and results in even lower cooling energy costs. On the other hand, the physical-actuator reduces T_{ac} if the overall temperature of the cluster becomes high again for any reason.

4.8 Yahoo! Production Hadoop Cluster Analysis

We analyzed one month of Hadoop Distributed File System (HDFS) audit logs and namespace checkpoints in Big Data analytics Hadoop cluster at Yahoo! to understand the characteristics (e.g., evolution life spans, sizes, jobs arrival rate, access patterns, and popularity) of the Big Data stored in the cluster. The cluster had 2600 servers, hosted 34 million files in the namespace, and had a data set size of 5 Petabytes. HDFS maintains the metadata i.e., the inode data (file size, last access time, last modified time, file name, user, and owner information) and the list of chunks belonging to each file in the master server's (NameNode) memory in a structure called fsimage. The persistent record of the fsimage is called a checkpoint and is typically taken once daily for the cluster.

In addition, HDFS has the ability to log all file system requests, which is required for auditing purposes in enterprises. The audit logging is implemented using log4j and once enabled, logs every HDFS event in the NameNode's log [144]. The file system operations logged by HDFS in the audit logs are "Open", "Create", "Rename", "Delete", "SetReplication", "SetOwner", "SetPermission", and "MkDir". The current implementation of HDFS doesn't support appends and the only writes to a file happen during file creation as part of the "Create" call. The files follow the write-once ready-many (WORM) pattern. The reads to a file happen as part of the "Open" call.

We use the above-mentioned checkpoint and HDFS logs for our analysis. There are 425 million entries in the HDFS logs and each namespace checkpoint contains 30-40 million files. The cluster namespace is divided into top-level directories, whereby each top-level directory handles different set of workloads, applications, and datasets. We consider four main directories and refer to them as: d, p, u, and m in our analysis instead of referring them by their real names. The total number of unique files that are seen in the HDFS logs in the one-month duration are 70 million (d-1.8

million, p-30 million, u-23 million, and m-2 million).

The directory d comprises entirely of the clickstream dataset which is the most important dataset in the cluster. Yahoo! [34] relies on clickstream processing [34, 52, 133], an example of Big Data analytics, to calculate the web-based advertising revenues (source of majority of their income), and to derive user interest models and predictions just like other Internet services companies such as Google, Facebook, Twitter, LinkedIn, and Facebook. Huge logs of clickstreams are continuously collected at the web servers and are copied over to the Hadoop production clusters daily. Map Reduce based applications are then used to compute various statistics and derive business insights from the data. The amount of the data that needs to be processed, makes clickstream processing a classic example of Big Data analytics.

The logs and the metadata checkpoints are huge in size and we used a large-scale research Hadoop cluster at Yahoo! extensively for our analysis. We wrote the analysis scripts in PIG [8]. One of the main goals of the study was to identify the evolution of the files in the cluster. To accurately understand the life spans and lifetime of the files in the cluster, we consider several file evolution cases in our analysis as shown below:

- Files created before the analysis period and which were not read or deleted subsequently at all. We classify these files as *long-living dormant* files.
- Files created before the analysis period which were read during the analysis period.
- Files created before the analysis period which were both read and deleted during the analysis period.
- Files created during the analysis period which were not read or deleted during the analysis period.
- Files created during the analysis period which were not read during the analysis period, but were deleted.
- Files created during the analysis period which were read and deleted during the analysis period.

To further accurately account for the files life spans, lifetime and to figure out complete set of information about the files, we handled the following cases: (a) filename reuse - we appended a

timestamp to each file create to accurately track the audit log entries following the file create entry in the audit log; (b) file renames - we used an unique id per file to accurately track its lifetime across create, rename and delete; (c) renames and deletes can happen at a much higher directory level than at an individual file-level. Thus, renames and deletes at higher level in the path hierarchy had to be translated to leaf-level renames and deletes for accurate determination of the lifetime of the files; (d) HDFS logs do not have file size information and, hence, we did a join of the dataset found in the HDFS audit logs and the namespace checkpoint to get the file size information.

4.8.1 File Lifespan Analysis of the Yahoo! Hadoop Cluster

A file goes to several stages in its lifetime in the Big Data analytics cluster: 1) file creation during which the file's chunks and replicas are written across the servers in the cluster, 2) *active* period during which the file is frequently computed upon and thereby, accessed, 3) *dormant* period during which file is no longer being computed upon or accessed and is awaiting its deletion. The file can retained in the cluster for long retention periods if it is subject to compliance, regulatory, or disaster recovery constraints, and 4) *deletion* when the file is deleted, and its storage space and metadata is reclaimed. We introduced and considered various life span metrics in our analysis to characterize a file's evolution. A study of the various life span distributions helps in deciding the energy-management policy thresholds that need to be in place in GreenHDFS.

- $\mathcal{L}_{cfr,j}$ metric is defined as the file life span between the file creation and first computational job directed to the file. This metric is used to find the clustering of the computational jobs around the file creation.

- $\mathcal{L}_{active,j}$ metric is defined as the file life span between creation and last computational job received by the file. This metric is used to determine the *active* profile of the files.

- $\mathcal{L}_{dormant,j}$ metric is defined as the file life span between last computational job directed to the file and file deletion. This metric helps in determine the *dormancy* profile of the files as this is the period for which files are dormant in the system.

- $\mathcal{L}_{flr,j}$ metric is defined as the file life span between first computational job and last computational job directed to the file. This metric helps in determining another dimension of

the *active* profile of the files.

- $\mathcal{L}_{lifetime,j}$. This metric helps in determining the lifetime of the file between its creation and its deletion.

4.8.1.1 $\mathcal{L}_{cfr,j}$

The $\mathcal{L}_{cfr,j}$ distribution sheds light on the clustering of the computational jobs with the file creation. As shown in Figure 4.20, 99% of the files have a $\mathcal{L}_{cfr,j}$ of less than two days, i.e., majority of the files get their first computational job within two days of their creation. The Big Data analytics jobs are data-intensive in nature and require high data access performance. Since files receive computational jobs soon after their creation, it is important to place the files upfront on the cluster in a manner that allows high write and read data access performance.

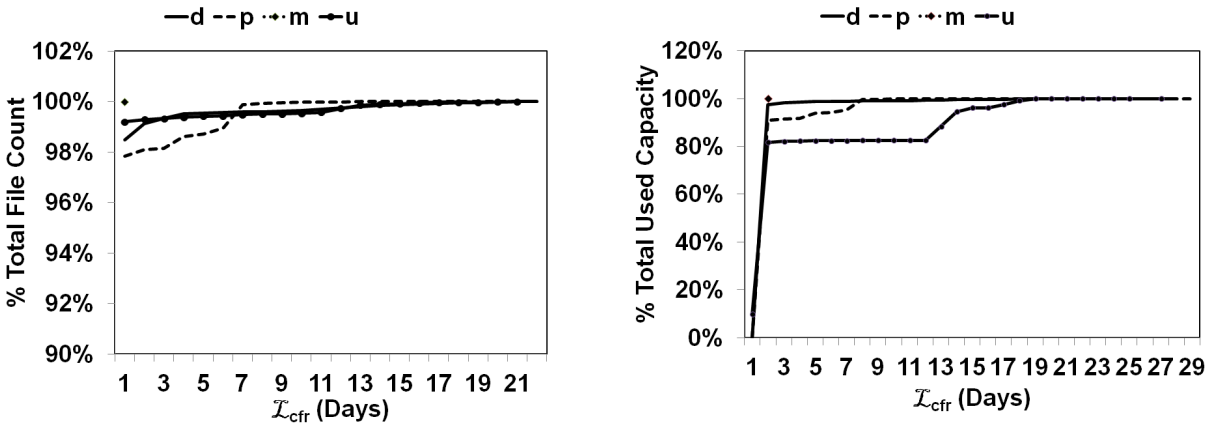

Figure 4.20: $\mathcal{L}_{cfr,j}$ **CDF as a percentage of total file count and total used capacity. 99% of files in directory d and 98% of files in directory p are accessed for the first time less than two days of their creation.**

4.8.1.2 $\mathcal{L}_{active,j}$

Figure 4.21 show the distribution of the active life span, $\mathcal{L}_{active,j}$ in the cluster, which sheds light on the duration of the active phase of the files. In directory d, 80% of files are actively computed

upon for less than eight days and 90% of the files amounting to 94.62% storage, are active for less than twenty four days. The active life span $\mathcal{L}_{active,j}$ of 95% of the files amounting to 96.51% storage in the directory p is less than three days and the $\mathcal{L}_{active,j}$ of the 100% of files in directory m is as short as two days. In directory u, 98% of files amounting to 98% of storage have $\mathcal{L}_{active,j}$ of less than one day. Thus, majority of the files in the cluster have a short active life span, i.e, they are actively computed upon only for short period of time. The directory d has files with longer active life span than the other directories.

An analysis of the distribution of the active life span, $\mathcal{L}_{active,j}$ in the cluster is important to guide the *File Migration Policy*'s threshold t_{FMP}. t_{FMP} needs to be chosen in a way that it allows in maximal energy savings, minimal data oscillations between GreenHDFS zones, maximal and fast active space reclamation, and minimal performance degradation. The value of the threshold t_{FMP} should also be chosen in way that is representative of a majority of the files in the dataset. For example, in directory p a small value of threshold t_{FMP} of two days may also work well. Since, 95% of the files have an active life span of less than three days, the intermittent dormancy that any of these files can experience is less than two days. Thus an aggressively low threshold value of two won't result in false dormancy determination of the files; thereby, won't result in . The advantage of low threshold value is that it allows fast reclamation of the precious active storage space; thereby, allowing placement of active files on the outer disk cynlinder zones for high transfer bandwidth (as discussed in Section 4.5.1). In directories m and u, the threshold t_{FMP} can be even lower in value. However, $\mathcal{L}_{active,j}$ can not be used in a stand-alone manner to decide the threshold values and it is important to consider the dormant life span $\mathcal{L}_{dormant,j}$ in conjunction as illustrated in the next section.

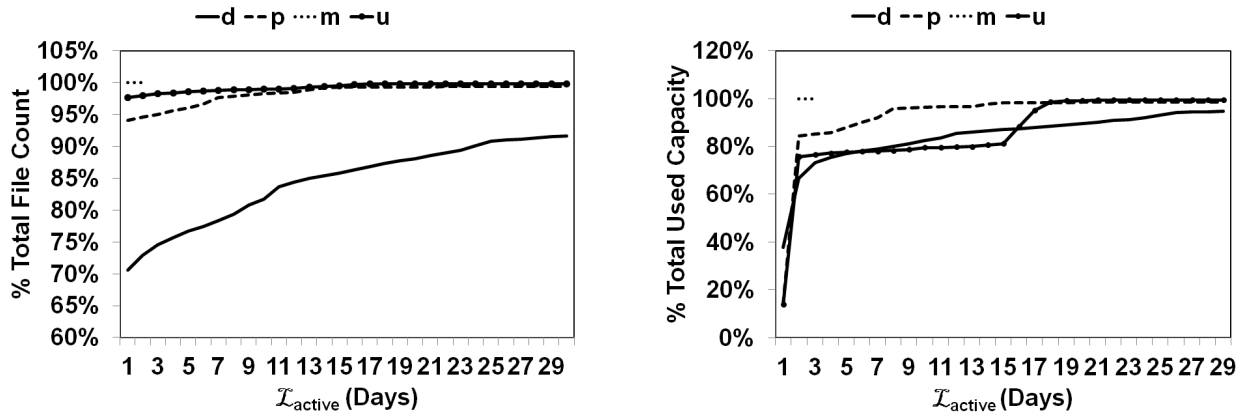

Figure 4.21: **Active life span, $\mathcal{L}_{active,j}$ CDF distribution as percentage of total file count and total used storage capacity in the four main top-level directories in the Yahoo! poduction cluster. $\mathcal{L}_{active,j}$ characterizes the life span for which files are actively being computed upon.**

4.8.1.3 $\mathcal{L}_{dormant,j}$

Files in the cluster enter a dormant phase once their active phase gets over. $\mathcal{L}_{dormant,j}$ indicates the time for which a file stays in a dormant state in the system, i.e., without getting accessed or computed upon, prior to getting deleted. Files are retained in dormant state for various reasons such as compliance, government regulations around file retentions, and disaster recovery requirements. The longer the dormancy period, higher is the suitability of the file for migration to the $Inactive$ zone. Figure 4.22 shows the distribution of $\mathcal{L}_{dormant,j}$ in the cluster. In directory d, 90% of files are dormant beyond one day and 80% of files, amounting to 80.1% of storage exist in dormant state past twenty days. In directory p, only 25% files are dormant beyond one day and only 20% of the files remain dormant in the system beyond ten days. In directory m, only 0.02% files are dormant for more than one day and in directory u, 20% of files amounting to 28.6% of storage are dormant beyond ten days.

The dormant life span $\mathcal{L}_{dormant,j}$ needs to be considered in conjuction with the active life span $\mathcal{L}_{active,j}$ in determination of the *File Migration Policy*'s threshold t_{FMP}. For directories *p, and u*, only 25% of the files are dormant beyond one day. A low threshold t_{FMP}'s value will result in unnecessary movement of files to the $Inactive$ zone as these files will get due for deletion

right after getting migrated to the $Inactive$ zone. On the other hand, given the short active life span $\mathcal{L}_{active,j}$ in these directories, high value of t_{FMP} won't do justice to space-efficiency in the $Inactive$ zone as discussed in Section 4.6.7.1. High value of t_{FMP} will leave files longer in the $Active$ zone even though the files are already in their dormant life span. A fine-tuned migration of the files right at the end of their active life span is the most conducive for using high-performance active file layouts such as using outer cylinder zones for higher transfer bandwidth. Now, such a scheme is only viable when dormant files are migrated as soon as possible, leaving available space for the newly active and new incoming files on the outer cylinder zones.

The dormant life span $\mathcal{L}_{dormant,j}$ needs to be considered to find true migration suitability of a file. For example, given the extremely short dormancy period of the files in the directory m, there is no point in exercising the *File Migration Policy* on directory m as these files are going to get deleted shortly anyways. File migration makes sense only for 75% of the files in case of directory p as the rest of the files have a very short dormant life span. Directory d stands to gain the most from the file migrations as it has a large number of files with long dormant life span.

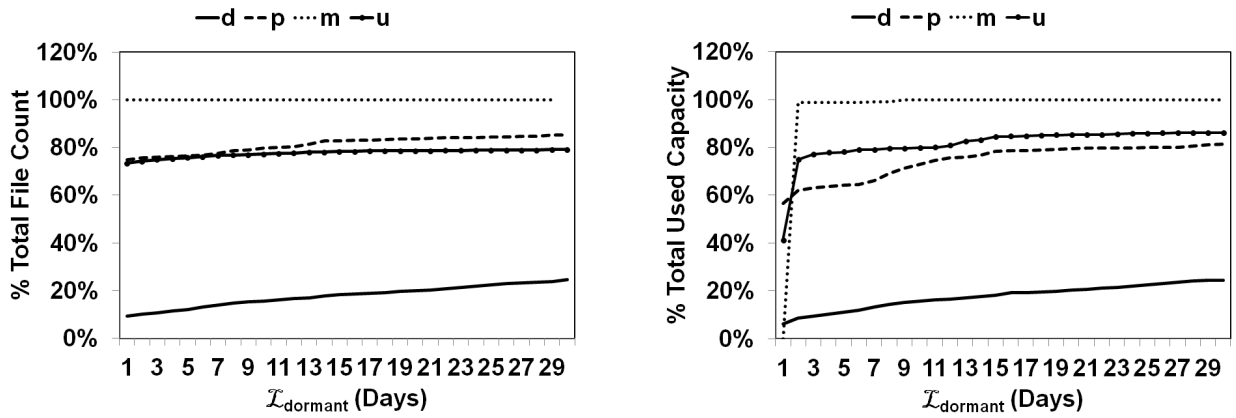

Figure 4.22: $\mathcal{L}_{dormant,j}$ **CDF distribution as percentage of total file count and total used storage capacity in the top-level directories in the Yahoo! production cluster.** $\mathcal{L}_{dormant,j}$ **characterizes the dormancy life span of the files and is indicative of the time a file stays in a dormant state in the cluster.**

4.8.1.4 File Lifetime Analysis

Knowledge of the FileLifetime further assists in the migration file candidate selection and needs to be accounted for in addition to the $\mathcal{L}_{dormant,j}$ and $\mathcal{L}_{active,j}$ metrices covered earlier. As shown in Figure 4.23, directory p only has 23% files that live beyond twenty days and 67% of the files are deleted within one day of their creation. On the other hand, 80% of files in directory d live for more than thirty days and 80% of the files have an active life span of less than eight days. Thus, directory d is a very good candidate for invoking the *File Migration Policy*.

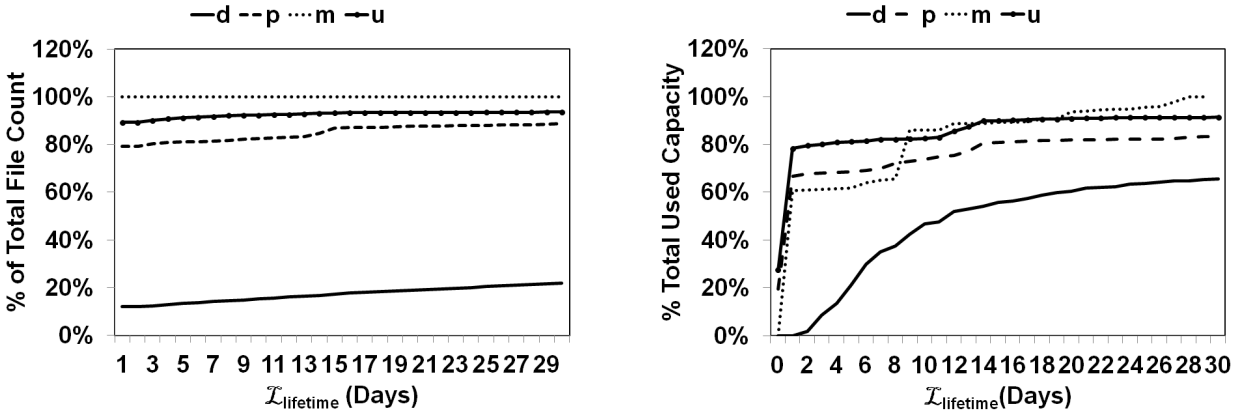

Figure 4.23: **File's lifetime CDF distribution as percentage of total file count and total used storage capacity in the top-level directories in the Yahoo! production cluster. File's lifetime is the span between file creation and deletion.**

4.8.2 Dormancy Characterization of the Files

The HDFS audit log analysis gives information about the files that are accessed in the one-month long observation duration. The files that are not accessed during this duration are not even present in the audit logs. Now, there are many more files in the cluster than the ones that show up in the log. To do a more comprehensive analysis, we analyzed the namespace checkpoints in addition to the audit logs for file dormancy characterization. In this Section, we show the file count and the storage capacity used by the long-living dormant files. The long-living dormant files are defined as the files that are created prior to the start of the observation period and are

not accessed during the one-month period of observation at all. Only directories d, p, and u are considered in the analysis as m doesn't have any long-living dormant files. As shown in Figure 4.33, in case of directory d directory, 13% of the total file count in the cluster which amounts to 33% of total used capacity is dormant. In case of directory p, 37% of the total file count in the cluster which amounts to 16% of total used capacity is dormant. Overall, 63.16% of total file count and 56.23% of total used capacity is in its dormant evolution phase in the system. Such long-living dormant files present significant opportunity to conserve energy in GreenHDFS. In addition to the long-living dormant files, there were other dormant files as well which were created towards the beginning of the observation period and were not accessed subsequently at all.

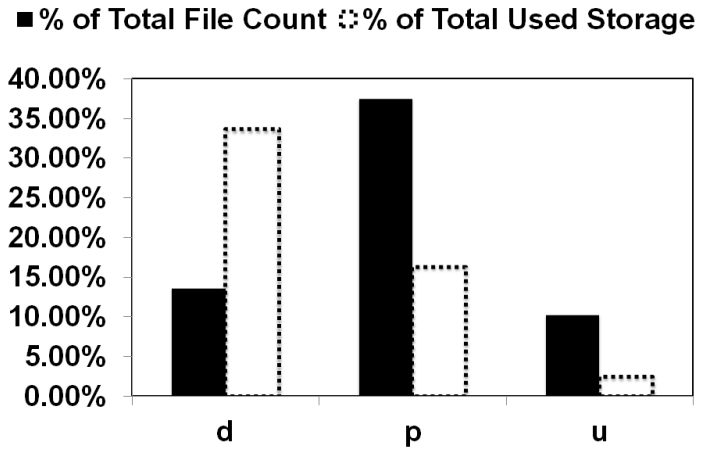

Figure 4.24: **File size and file count percentage of long-living dormant files. The long-living dormant files are defined as the files that are created prior to the start of the observation period of one-month and are not accessed during the period of observation at all.**

4.9 GreenHDFS Reactive Zoning Evaluation

In this section, we first present our experimental platform and methodology, followed by a description of the workloads used and our experimental results. Our goal is to answer *five* high-level sets of questions:

- How much energy is GreenHDFS able to conserve compared to a baseline HDFS with no

energy management?

- What is the penalty of energy management on the average response time?
- What is the sensitivity of the various policy thresholds used in GreenHDFS on the energy savings results?
- How many power state transitions does a server go through in average in the $Inactive$ zone?
- What is the number of accesses that happen to the files in the $Inactive$ zone, the days servers are in an active power state, and the number of migrations and reversals observed in the system?

The following evaluation sections answer these questions, beginning with a description of our methodology, and the trace workloads we use as inputs to the experiments.

4.9.1 Evaluation Methodology

We evaluated GreenHDFS using a trace-driven simulator. The simulator was driven by real-world HDFS traces generated by a production Hadoop cluster at Yahoo!. The cluster had 2600 servers, hosted 34 million files in the namespace and the data set size was 6PB.

We focus our evaluation on the largest (amounts to 60% of the total used capacity) and most important log-processing (clickstream) dataset in the Yahoo! Hadoop production cluster (4PB out of the 6PB total used capacity). Since, we were evaluating only with 60% of the used capacity of the original cluster, we assumed a cluster with just 60% of the 2600 servers assigned to it. The total number of unique files that were seen in the HDFS traces dataset in the one-month duration were 0.9 million. In our experiments, we compared GreenHDFS to the baseline case (HDFS without energy management). The baseline results gave us the upper bound for energy consumption and the lower bound for average response time.

Simulation Platform: We used models for the power levels, power state transitions times, and access times of the disk, processor, and the DRAM in the simulator. The GreenHDFS simulator was implemented in Java and MySQL distribution 5.1.41 and executed using Java 2 SDK, version

1.6.0-17. The GreenHDFS simulator uses models for the power levels, power state transitions times and access times of the Seagate Barracuda ES.2 1TB disk [9], Intel X25-E 64GB SSD [129], NIC [10], Intel Xeon X5400 CPU [75], power supply unit [117] and the DRAM [104]. Table 4.6 lists the various power, latency, and transition times used in the Simulator.

Table 4.5: **Power and power-on penalties used in Simulator**

Component	Active Power (W)	Idle Power (W)	Sleep Power (W)	Power-up time
CPU (Quad core, Intel Xeon X5400 [75])	80-150	12.0-20.0	3.4	30 us
DRAM DIMM [104]	3.5-5	1.8-2.5	0.2	1 us
NIC [10]	0.7	0.3	0.3	NA
SATA HDD (Seagate Barracuda ES.2 1TB [9])	11.16	9.29	0.99	10 sec
PSU [117]	50-60	25-35	0.5	300 us
Active zone server (2 CPU, 8 DRAM DIMM, 4 1TB HDD)	445.34	132.46	13.16	
Inactive zone server (2 CPU, 8 DRAM DIMM, 12 1TB HDD)	534.62	206.78	21.08	

4.9.2 Energy-Conservation

In this section, we show the energy savings made possible by GreenHDFS, compared to baseline, in one-month by doing power management of the clickstream dataset. The cost of electricity is assumed to be \$0.063/KWh to be consistent with the electricity rates in California. Figure 4.25 shows a 24% reduction in energy costs of a 1560 server cluster with 80% capacity utilization. At Yahoo!, an upwards of 38000 servers are deployed across the Big Data analytics Hadoop clusters. Extrapolating, \$2.1 million can be saved in the energy costs if GreenHDFS technique is applied

to all the Hadoop clusters at Yahoo!. Energy costs savings from scaled-down servers is further compounded by the cooling energy costs savings. For every watt of power consumed by the compute infrastructure, a modern data center expends another one-half to one watt to power the cooling infrastructure [113].

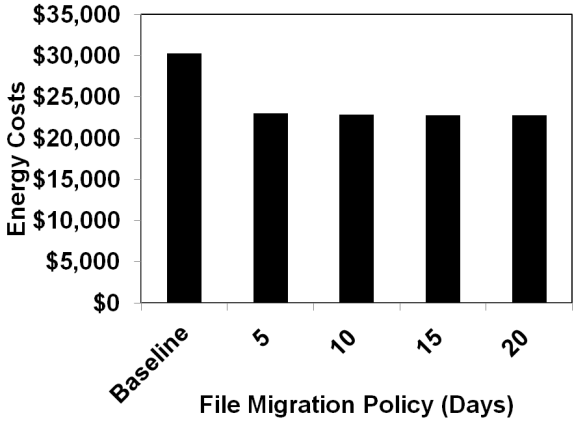

Figure 4.25: **Energy costs savings with GreenHDFS in one-month by doing data-centric power management for the clickstream data set in the production Yahoo! cluster.**

4.9.3 Storage-Efficiency

The *File Migration Policy* in GreenHDFS moves dormant files from the $Active$ zone and consolidates them on the $Inactive$ zone. While, the primary goal of the policy is to enable data-centric scale-down, it also aids in freeing up valuable storage capacity in the $Active$ zone. The additional free storage space can be used for a variety of purposes such as high-performance layout of the active files as covered in 4.5.1 that depends on file migration to free up space on the outer disk cylinder zones of zoned bit recording disks for the placement on the new files, and aggressive replication of the high popularity active files for better load-balancing and alleviation of performance hot spots. Figure 4.26 shows the storage space used in the $Active$ zone with different thresholds t_{FMP} of the *File Migration Policy*. In the baseline case, the average storage capacity utilization of the 1560 servers deployed in the cluster is higher than GreenHDFS which just has 1170 servers out of the 1560 servers deployed in the $Active$ zone. More aggressive, i.e., lower the policy threshold t_{FMP}, more space is available in the $Active$ zone as more files get migrated out to the $Inactive$ zone daily.

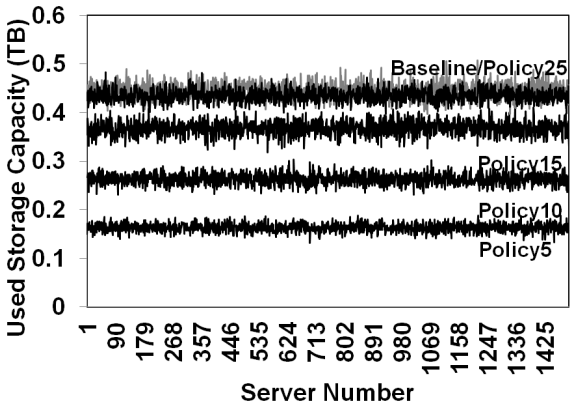

Figure 4.26: **Used storage capacity in the $Active$ zone with different File Migration Policy's threshold t_{FMP} values. GreenHDFS substantially increases the free space in the $Active$ zone by migrating dormant data to the $Inactive$ zone.**

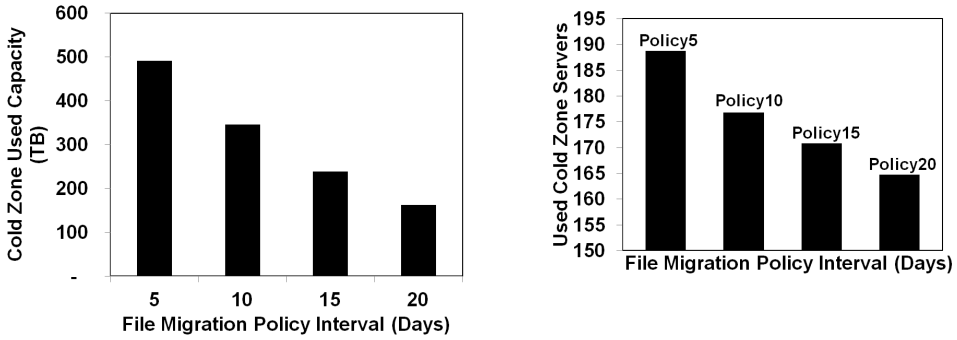

Figure 4.27: **Storage capacity and number of servers used in the $Inactive$ zone to store dormant files with different File Migration Policy's threshold t_{FMP} values.**

4.9.4 File Migrations

It is important to evaluate the amount of data that would need to get migrated everyday by the *File Migration Policy* to understand the advantages and repurcussions of the same. The Figure 4.28 shows the total number and storage size of the files which are migrated to the $Inactive$ zone daily with a *File Migration Policy*'s threshold t_{FMP} value of ten days. Every day, on average 6.38TB worth of data and 28.9 thousand files are migrated to the $Inactive$ zone. Since, we

have assumed storage-heavy servers in the $Inactive$ zone where each server has 12, 1TB disks, assuming 80MB/sec of disk bandwidth, 6.38TB data can be absorbed in less than 2 hours by one server. The *File Migration Policy* is run during off-peak hours to minimize any performance impact.

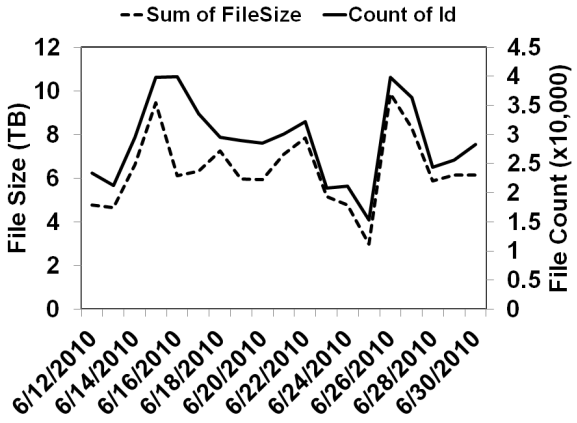

Figure 4.28: **The number and total size of the files migrated daily to the $Inactive$ zone with t_{FMP} value of ten days.**

4.9.5 Impact of Power Management on Response Time

It is very important to understand the performance impact of energy management. In GreenHDFS, an access to a file residing on the $Active$ zone server gets the same performance as the baseline cluster. However, an access to a file residing on the $Inactive$ zone may suffer performance degradation in two ways: 1) if the file resides on a server that has been scaled-down, the server would need to be transitioned back into an active power state. Transitioning components such as disks can take an upwards of ten seconds and thereby, result in significant performance degradation, and 2) the files are not chunked in interest of energy savings in the $Inactive$ zone and hence, can not enjoy high inter-file data access bandwidth which is possible when file is chunked across the servers in the cluster. Figure 4.29 shows the impact on the average response time. 97.8% of the total read requests are not impacted by the power management. Impact is seen only by 2.1% of the reads. One main reason for the quite low performance impact is that the clickstream workload has a news-server-like access pattern. Files are predominately accessed and computed upon right after their creation. As the file ages, the accesses go down. With a less

aggressive, i.e., higher value of *File Migration Policy*'s threshold t_{FMP} (e.g., 15, 20 days), impact on the response time reduces much further. A higher value of threshold t_{FMP} results in higher accuracy in determining the truly dormant files and is less susceptible to intermittent periods of dormancy in the file's active life span. As a result, once the files are deemed dormant and are moved to the $Inactive$ zone, they receive much less (if any) accesses. Since, all the performance degradation stems from an access to the $Inactive$ zone, cutting down the accesses cuts down the performance impact.

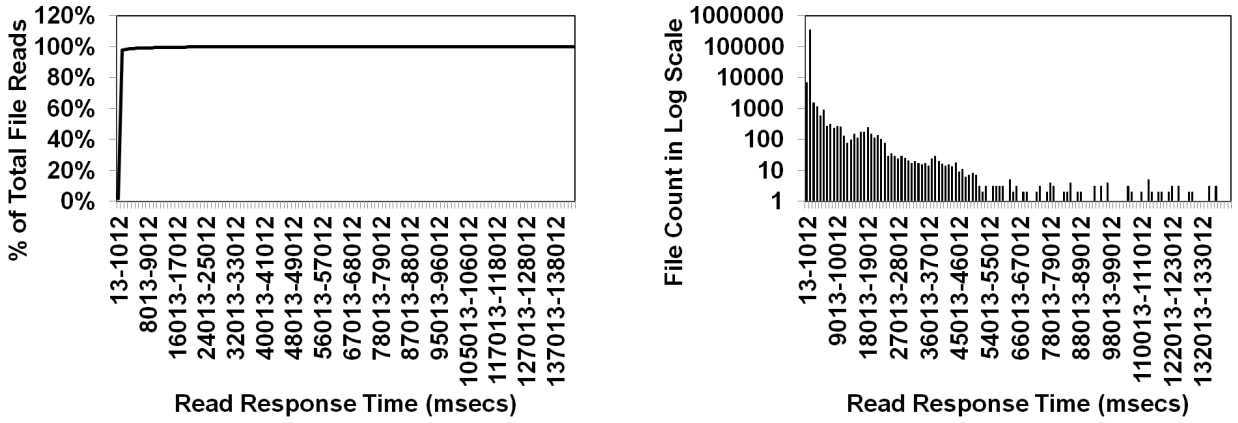

Figure 4.29: **Performance Analysis: Impact on average response time because of power management with a t_{FMP} value of 10 Days.**

4.9.6 Sensitivity Analysis

We tried different values of the thresholds for the *File Migration* policy and the *Server Power Conserver* policy to understand the sensitivity of these thresholds on storage-efficiency, energy-conservation and number of power state transitions.

t_{FMP}: We found that the *energy costs* are *minimally sensitive* to the t_{FMP} threshold value. As shown in Figure 4.25, the overall energy cost savings varied minimally when the t_{FMP} was changed to 5, 10, 15 and 20 days.

The performance impact is minimally sensitive to the t_{FMP} value as well as shown in the Figure 4.30. This behavior can be explained by the observation that majority of the data in the production

Hadoop cluster at Yahoo! has a news-server-like access pattern. This implies that once data is deemed cold, there is low probability of data getting accessed again.

The Figure 4.30 shows the total number of migrations of the files which were deemed *dormant* by the file migration policy and the reversals of the moved files in case they were later accessed by a client in the one-month simulation run. There were more instances (40,170, i.e., 4% of overall file count) of file reversals with the most aggressive t_{FMP} of 5 days. With less aggressive t_{FMP} of 15 days, the number of reversals in the system went down to 6,548 (i.e., 0.7% of file count). The experiments were done with a t_{FRP} value of 1. The number of file reversals are substantially reduced by increasing the t_{FRP} value. With a t_{FRP} value of 10, zero reversals happen in the system.

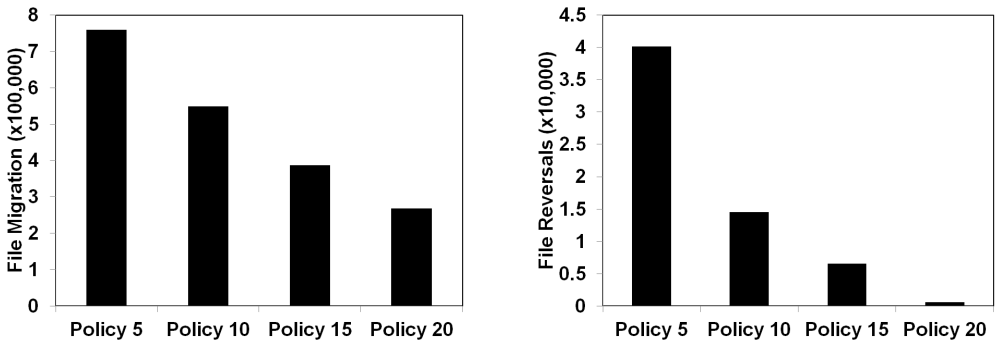

Figure 4.30: **Number of migrations/reversals in GreenHDFS with different values of the t_{FMP} threshold.**

The storage-efficiency is sensitive to the value of the t_{FMP} threshold as shown in Figure 4.31. An increase in the t_{FMP} value results in less efficient capacity utilization of the $Active$ zones. Higher value of t_{FMP} threshold signifies that files will be chosen as candidates for migration only after they have been dormant in the system for a longer period of time. This would be an overkill for files with very short $\mathcal{L}_{active,j}$ as they will unnecessarily lie dormant in the system, occupying precious $Active$ zone capacity for a longer period of time.

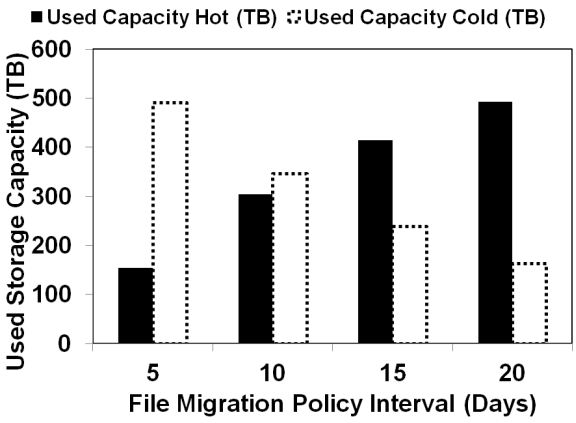

Figure 4.31: **Sensitivity Analysis: Sensitivity of used capacity per zone to the File Migration Policy's threshold t_{FMP}.**

t_{SCP}**:** As Figure 4.32 illustrates, increasing the t_{SCP} value, minimally increases the number of the days the servers in the $Inactive$ zone remain ON and hence, minimally lowers the energy savings. On the other hand, increasing the t_{SCP} value results in a reduction in the power state transitions which improves the performance of the accesses to the $Inactive$ zone. Thus, a trade-off needs to be made between energy-conservation and data access performance.

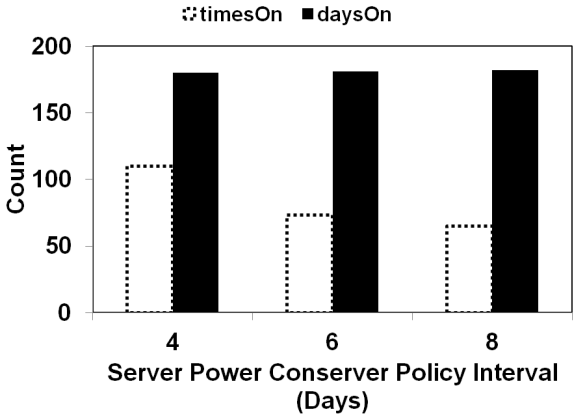

Figure 4.32: **Sensitivity Analysis: Sensitivity of the number of power state transitions to the Server Power Conserver Policy's threshold t_{SCP}.**

Summary on Sensitivity Analysis: From the above evaluation, it is clear that a trade-off needs

to be made in choosing the right thresholds in GreenHDFS based on an enterprise's needs. If $Active$ zone space is at a premium, more aggressive t_{FMP} needs to be used. This can be done without impacting the energy-conservation that can be derived in GreenHDFS.

4.9.7 Number of Server Power Transitions

It is important to limit the power state transitions incurred by servers as some server components such as disks have limited start/stop cycles. Too many power state transitions can significantly impact the hardware reliability. The Figure 4.33 shows the number of power state transitions incurred by the servers in the $Inactive$ zone. We only show those servers in the $Inactive$ zone that either received newly cold data or had data accesses targeted to them in the one-month simulation run. The maximum number of power state transitions incurred by a server in a one-month simulation run is just 11 times and only 1 server out of the 390 servers provisioned in the $Inactive$ zone exhibited this behavior. Most of the disks are designed for a maximum service life time of five years and can tolerate up to 50,000 start/stop cycles. Given the very small number of transitions incurred by a server in the $Inactive$ zone in a year, GreenHDFS has no risk of exceeding the start/stop cycles during the service life time of the disks.

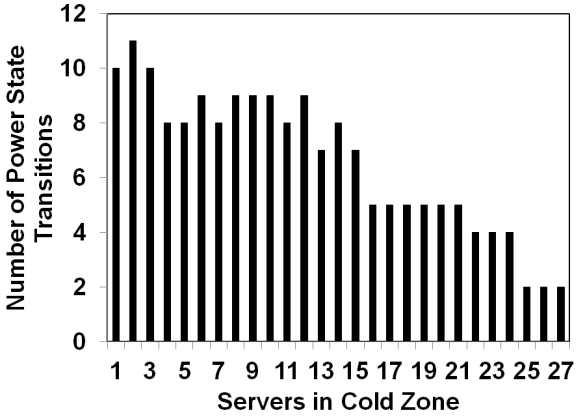

Figure 4.33: $Inactive$ **zone Behavior: Number of power state transitions of servers in the** $Inactive$ **zone with** t_{FMP} **of ten days.**

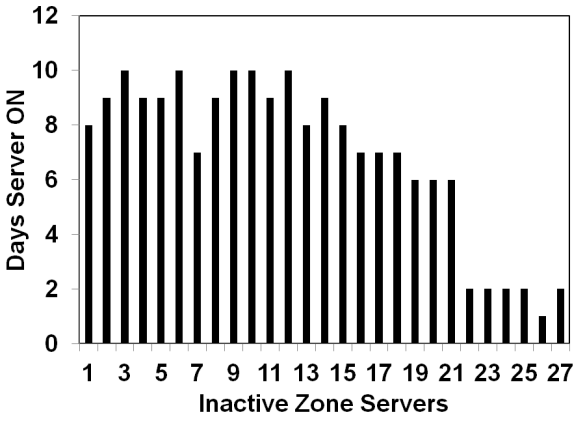

Figure 4.34: **Days servers in $Inactive$ zone are in active power state during the one-month simulation run. The servers in the $Inactive$ zone are scaled-down by the server power conservation policy and are transitioned back to active power state with triggering events specified in Section 4.6.8.**

4.10 GreenHDFS Predictive Zoning Evaluation

In this section, we evaluate the predictive capability of GreenHDFS and the predictive variants of the File Migration, Zone Placement, and Replication policies. We first present our experimental platform and methodology, and then we give our experimental results. Our goal is to answer high-level sets of questions:

- How does Predictive GreenHDFS compare with Reactive GreenHDFS in energy costs savings?
- How many data accesses happen to the files residing in the $Inactive$ zone?
- How does Predictive GreenHDFS compare with Reactive GreenHDFS in performance impact of energy management?
- How accurate are the predictions?
- What is the storage utilization in Predictive GreenHDFS compared to the Reactive GreenHDFS?

- How many file migrations and reversals happen in Predictive GreenHDFS compared to Reactive GreenHDFS?

4.10.1 Methodology

To measure the accuracy of the prediction model, it is important to evaluate the prediction model on a test dataset which is *different* from the training and the validation dataset discussed in Section 4.6.1.3. Our test dataset comprises of the HDFS traces from the month following the month used for creating aggregates for the training and the validation datasets. To be consistent with the data analysis and predictor training, we focus our evaluation on the largest (amounts to 60% of the total used capacity) and most important log-processing (clickstream) dataset in the Yahoo! Hadoop production cluster.

Table 4.6: **Power and power-on penalties used in Simulator**

Component	Active Power (W)	Idle Power (W)	Sleep Power (W)	Power-up time
CPU (Quad core, Intel Xeon X5400 [75])	80-150	12.0-20.0	3.4	30 us
DRAM DIMM [104]	3.5-5	1.8-2.5	0.2	1 us
NIC [10]	0.7	0.3	0.3	NA
SATA HDD (Seagate Barracuda ES.2 1TB [9])	11.16	9.29	0.99	10 sec
PSU [117]	50-60	25-35	0.5	300 us
$Active$ zone (SSD) [129] Server (2 CPU, 8 DRAM DIMM, 256GB SSD)	411.1W	132.46	13.16	
$Active$ zone Server (2 CPU, 8 DRAM DIMM, 4 1TB HDD)	445.34	132.46	13.16	
$Inactive$ zone Server (2 CPU, 8 DRAM DIMM, 12 1TB HDD)	534.62	206.78	21.08	

The GreenHDFS simulator uses models for the power levels, power state transitions times and access times of the Seagate Barracuda ES.2 1TB disk [9], Intel X25-E 64GB SSD [129], NIC [10], Intel Xeon X5400 CPU [75], power supply unit [117] and the DRAM [104] in the simulator and is implemented in Java and MySQL distribution 5.1.41 and executed using Java 2 SDK, version 1.6.0-17. The predictor module uses the JAMA-1.0.2 matrix library.

We use 60% (i.e.,1560) of the total 2600 cluster nodes in our analysis to be consistent with the test dataset. The simulator assumes 50 servers in the SSD-based *ActiveSSD* Zone, 1120 servers in the SATA-based $Active$ zone and 390 servers in the $Inactive$ zone. We assumed

SSD bandwidth of 250MB/sec and latency of 75 microseconds, and SATA HDD bandwidth of 105MB/s and latency of 12.66 milliseconds. The Predictive GreenHDFS simulator incorporates the predictor component with its predictive models and predictive data management policies. The simulator uses the same file chunking policy and three-way replication of the file chunks as is done in the Hadoop Distributed File System (HDFS) in the $Active$ zone. The $Inactive$ zone also does three-way replication of the file chunks just like HDFS.

In our experiments, we compare Predictive GreenHDFS to the Reactive GreenHDFS [79]. We use three values of policy thresholds ranging from low (5 days) to high (15 days) in the Reactive GreenHDFS as each threshold makes a different energy-performance-storage efficiency trade-off. To ensure best performance (i.e., least performance degradation associated with the energy management) in the Reactive GreenHDFS, we move a file back to the $Active$ zone if the file is accessed even once while it is on the $Inactive$ zone. We cover the evaluation results in the following sections.

4.10.2 Energy-Conservation

In this section, we discuss the energy savings made possible by Predictive GreenHDFS, in comparison to the Reactive GreenHDFS. The cost of electricity is assumed to be \$0.063/KWh. We run the Predictive and Reactive simulators with the one-month long traces and determine the energy costs incurred by the servers in the $Inactive$ zone. As shown in Figure 4.35, we find that predictive GreenHDFS is able to cut down on the energy costs in the $Inactive$ zone compared to all threshold values in Reactive GreenHDFS.

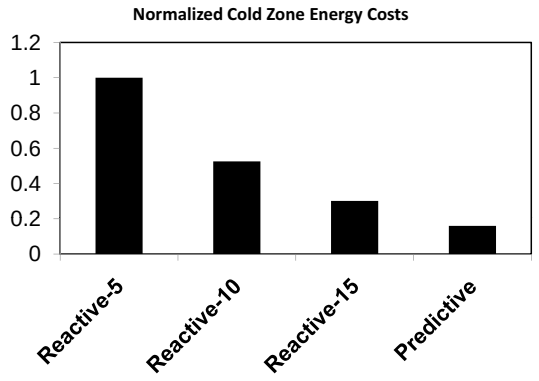

Figure 4.35: **Comparison of normalized energy costs of the $Inactive$ zone servers in Predictive GreenHDFS vs. Reactive GreenHDFS with different threshold values (in days)**

GreenHDFS keeps a set of three servers in an active power state in the $Inactive$ zone at any point in time to consume the files deemed as cold by the predictive File Migration policy. Once, these servers have been filled to their usable storage capacity, they become candidates for scale-down and a new set of three servers are chosen for the incoming cold data. Server Power Conserver policy in GreenHDFS transitions the servers in the $Inactive$ zone to inactive power state if the cold files residing on these servers haven't been accessed in a day. In the future, any access to a file residing on a sleeping server in the $Inactive$ zone results in a server wakeup via Wake-on-LAN capability. Thus, frequent accesses to the files residing on the $Inactive$ zone run the risk of reducing energy savings.

Predictive GreenHDFS cuts down on the accesses to the files in the $Inactive$ zone, as shown in Figure 4.37(b) courtesy of its predictive file migration policy. On the other hand, the Reactive GreenHDFS is not able to foretell the future accurately and hence, results in file accesses to the $Inactive$ zone, thereby resulting in lower energy savings.

4.10.3 Overall Performance

An access to a file residing on a server in the $Inactive$ zone may suffer performance degradation in two ways: 1) if the file resides on a server that has been scaled-down, the server will need to be woken up, thereby incurring a significant server wakeup time penalty courtesy of components such as the hard disk drives (In the sleep mode, the drive heads are parked away from the drive platters (unloaded), and the platters are completely spun down, resulting in negligible power consumption. However, bringing drive back can take as long as 10 seconds.), 2) in interest of aggressive energy savings, GreenHDFS doesn't chunk files in the $Inactive$ zone and places a file in its entirety on a server. This ensures that a future access to the file only results in one server wakeup as opposed to waking up all the servers containing the file's chunks. However, such a non-parallel data policy does result in a degradation in the average response time of the file accesses to the $Inactive$ zone. We logged the response time of each file access during the simulator run for this experiment. As shown in the Figure 4.37, predictive GreenHDFS results in significantly, 40% lower average response time than the best performing Reactive GreenHDFS as it is effective in cutting down all the accesses to the $Inactive$ zone.

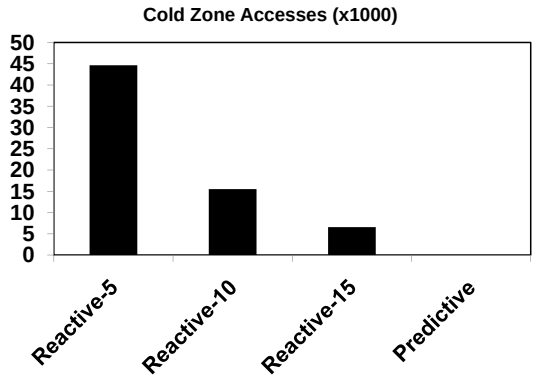

Figure 4.36: **Number of accesses to the $Inactive$ zone in Predictive vs. Reactive GreenHDFS.**

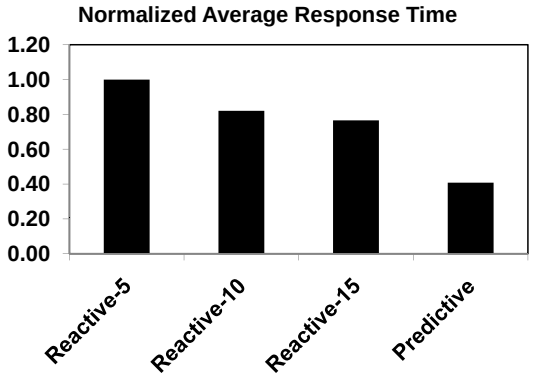

Figure 4.37: **Comparison of normalized average response time of Predictive GreenHDFS vs. Reactive GreenHDFS with different threshold values.**

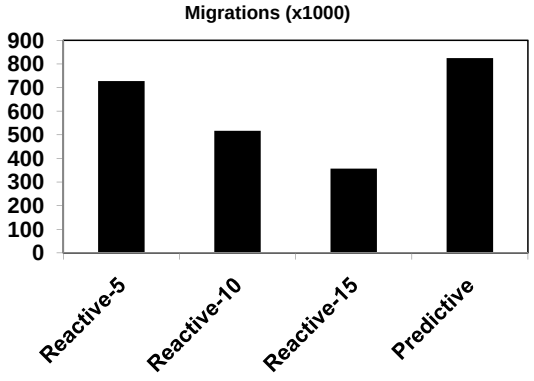

Figure 4.38: **File migrations in Predictive vs. Reactive GreenHDFS with different threshold values.**

4.10.4 $Active$ zone Free Space

In this section, we show the amount of the hot storage space made available in the $Active$ zone by moving cold files to the $Inactive$ zone. As shown in Figure 4.40, predictive GreenHDFS results in significant increase in the hot space available in the $Active$ zone by moving cold files in

a proactive and fine-tuned manner at the end of the files' hotness lifespan to the $Inactive$ zone as shown in Figure 4.40. The extra capacity available in the $Active$ zone can be used for storing more hot data and also allows more space for more performance-driven file replication.

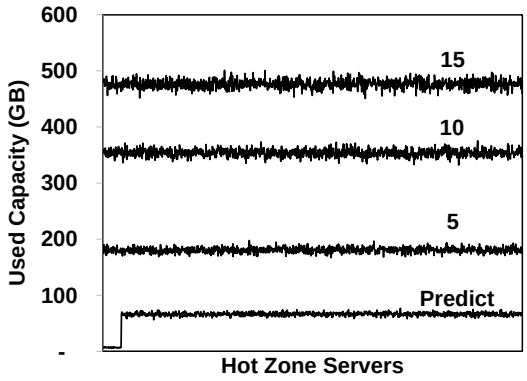

Figure 4.39: **Space used in $Active$ zone in Predictive vs. Reactive GreenHDFS with different threshold values.**

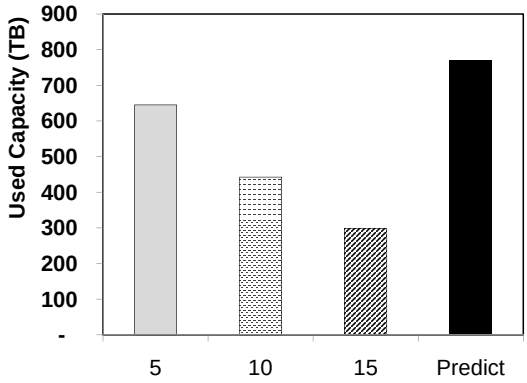

Figure 4.40: **Space used in the $Inactive$ zone in Predictive vs. Reactive GreenHDFS with different threshold values.**

4.10.5 Accuracy of Prediction

We computed the root mean squared error (RMSE) between the predicted file attribute values during the evaluation run and the aggregated, actual value of the file attributes to evaluate the accuracy of the predictors. We got an RMSE of 1.870 in the prediction of the file size which range from few bytes to several terabytes in the dataset. The RMSE for the lifespan predictor was 0.4 where the dataset values ranged from 0 to 29+ days. The RMSE for the number of accesses to the files was 0.204 where the number of accesses ranged from 0 to few thousands.

Instead of going after 100% prediction accuracy, we chose to ensure that there was no cost associated with the mispredictions. For example, prediction of file size is used in an advisory fashion only in our zone placement algorithm covered in Section 4.6.2. Hence, the algorithm is not sensitive to the RMSE of 1.870 and there is no need to make the prediction any more accurate.

The real measure of a predictive model's effectiveness is its performance on unseen samples. Our results indicate the presence of significant predictability and correlation between the filenames in dataset and the file attributes of the resident files.

4.10.6 Summary

We compared Predictive GreenHDFS with three threshold values (5 (aggressive), 10 and 15 (least aggressive) days) of the file migration policy in Reactive GreenHDFS. In Reactive GreenHDFS, policy threshold of 5 days aggressively moves data to the $Inactive$ zone, thereby resulting in highest free storage space savings in the $Active$ zone compared to other policy thresholds. However, the aggressive policy results in the highest number of accesses to the $Inactive$ zone thereby decreasing energy savings and increasing average response time performance degradation of the file accesses compared to other thresholds. The least aggressive reactive policy threshold (15 days) results in the least number of accesses to the $Inactive$ zone and hence, increases the energy savings while reducing performance degradation. However, it results in leaving cold data in the $Active$ zone for much longer time and hence, results in least amount of free storage space savings. Predictive GreenHDFS results in higher energy savings, less performance degradation and higher free storage space savings in the $Active$ zone than all flavors of Reactive GreenHDFS

courtesy of its predictive, finely-tuned, per-file file migration policy.

4.10.7 Thermal-Aware Zoning

In this section, we evaluate the cooling energy costs savings possible with thermal-aware zone partitioning feature of GreenHDFS. GreenHDFS does zoning in a thermal-aware manner and assigns the most inherently cooling-inefficient servers in the cluster to the $Inactive$ zone as $Inactive$ zone hosts dormant files whose low or negligible computational load-profile is a great fit for the cooling-inefficient servers given their impaired heat dissipation capability. Such a placement ensures that the cooling-inefficient servers receive negligible computations; as a result, the cooling-inefficient servers don't generate much heat, and their exhaust temperature remains bounded. The servers in the $Inactive$ zone experience sufficient idleness because of the dormant class of files hosted by the zone, and can be effectively scaled-down. In this section, we evaluate and isolate the contribution of various features of GreenHDFS towards saving cooling energy costs. The high-level questions that we attempt to answer are:

- How does thermal-aware zone partitioning compare with non-thermal-aware zone partitioning in saving cooling energy costs? In case of non-thermal-aware zone partitioning, servers are randomly allocated to the $Inactive$ and $Active$ zone. On the other hand, servers are allocated in a cooling-efficiency differentiated manner in case of thermal-aware zone partitioning.

- How much does scale-down of $Inactive$ zone servers contribute to cooling energy costs savings?

- Which zone partitioning scheme is more effective in saving cooling energy costs: rack-level zone partitioning or cluster-level zone partitioning?

- Can thermal-aware file placement save cooling energy costs on its own without thermal-aware zoning? How much additional savings can be realized upon combining both the techniques?

Table 4.7: **Evaluation Configurations**

Configuration	Options	Explanation
HDFS		Baseline. Cluster without GreenHDFS deployment, i.e., with no energy- or thermal-management
RR		GreenHDFS uses thermal-aware rack-aware zone partitioning covered in Section 4.5.2.2 to create thermal-aware $Active$ and $Inactive$ zones. GreenHDFS does file migration, and server scale-down. There is no thermal-aware fine-grained file placement done in $Active$ zone in this scenario.
	RR_10	10% cluster servers allocated to $Inactive$ zone and 90% to $Active$ zone.
	RR_20	20% cluster servers allocated to $Inactive$ zone and 80% to $Active$ zone.
	RR_30	30% cluster servers allocated to $Inactive$ zone and 70% to $Active$ zone.
CR		GreenHDFS uses cluster-level zone partitioning as covered in Section 4.5.2.1 to create thermal-aware $Active$ and $Inactive$ zones . GreenHDFS does file migration, and server scale-down. There is no thermal-aware fine-grained file placement done in $Active$ zone in this scenario.
	CR_10	10% cluster servers allocated to $Inactive$ zone and 90% to $Active$ zone.
	CR_20	20% cluster servers allocated to $Inactive$ zone and 80% to $Active$ zone.
	CR_30	30% cluster servers allocated to $Inactive$ zone and 70% to $Active$ zone.
NSD		Same as GreenHDFS_CR_30, but doesn't do any server scale-down in the $Inactive$ zone
TPDOnly_CR_30		Same as GreenHDFS_CR_30. In addition, thermal-aware fine-grained file placement is done in the $Active$ zone as covered in Section 5.3.2.2.
TPDOnly		GreenHDFS doesn't divide the cluster into zones, and doesn't do any file migration or server scale-down. GreenHDFS only does thermal-aware, predictive, fine-grained data placement cluster-wide.

To simulate a Big Data analytics cluster we use Mentor Graphic's floVENT, a computational fluid dynamics (CFD) simulator [6]. floVENT has been extensively used and validated in several research papers in the past [29, 107, 126]. The cluster under evaluation has 4 rows of 14 industry standard 47U racks arranged in cold- and hot-aisle layout [132]. Each rack contains 46, 1U servers for a total of 2576 servers. The cold air is supplied by CRACs with supply temperature T_{ac} fixed at 15^{o}C.

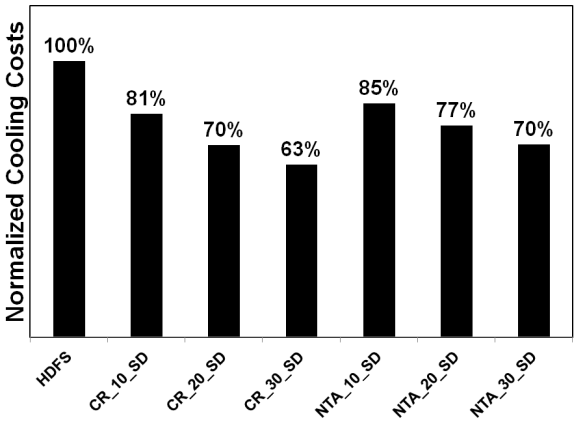

Figure 4.41: **The cooling energy costs with non-thermal-aware and thermal-aware variant of GreenHDFS normalized to baseline HDFS.**

4.10.7.1 Thermal-Aware Vs. Non-Thermal-Aware GreenHDFS Zoning

In this section, we compare a non-thermal-aware variant of GreenHDFS in which $Inactive$ zone servers are chosen randomly from the cluster without any thermal-awareness. In the thermal-aware variant of GreenHDFS, servers in the cluster are ranked by their cooling-efficiencies (i.e., inlet temperatures) and the most cooling-efficient servers are allocated to the $Inactive$ zone and the rest of the cooling-efficient servers are assigned to the $Active$ zone. As shown in the Figure 4.41, higher cooling energy costs savings are realized by doing thermal-aware zoning in GreenHDFS than by doing non-thermal-aware zoning. The thermal-profile of the servers becomes more uniform and lower, and thermal hot-spots reduce further courtesy of a reduction in hot air recirculation with thermal-aware zoning compared to non-thermal-aware zoning; thereby, resulting in higher cooling energy costs savings.

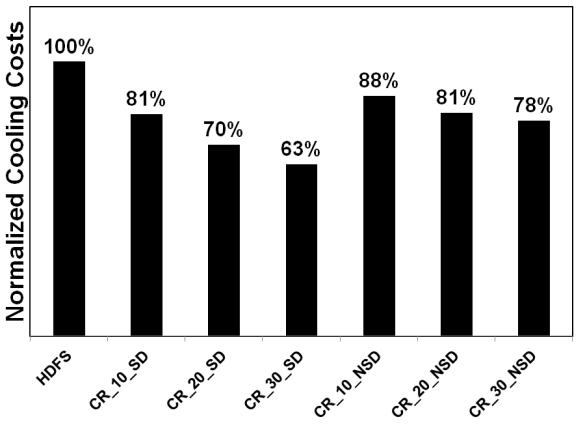

Figure 4.42: **The cooling energy costs with thermal-aware GreenHDFS with and without scaling-down the $Inactive$ zone normalized to baseline. GreenHDFS has cluster-level zone partitioning activated and a varying percentage of servers are scaled-down.**

4.10.7.2 Scale-Down vs. Not Scaled-Down

In this section, we evaluate the effect of scale-down of the $Inactive$ zone servers on cooling energy costs savings. We use cluster-level zone partitioning and employ different scale-down factors ranging from 10%-30%. As shown in the Figure 4.42, more cooling energy costs savings can be realized by scaling-down the $Inactive$ zone servers and scale-down allows an additional 7%-15% in cooling energy costs. GreenHDFS is capable of saving both server and cooling energy costs even if as low as 10% servers in the cluster are assigned to the $Inactive$ zone. As shown in Figure 4.42, even 10% scale-down factor is capable of saving 9%-12% of cooling energy costs.

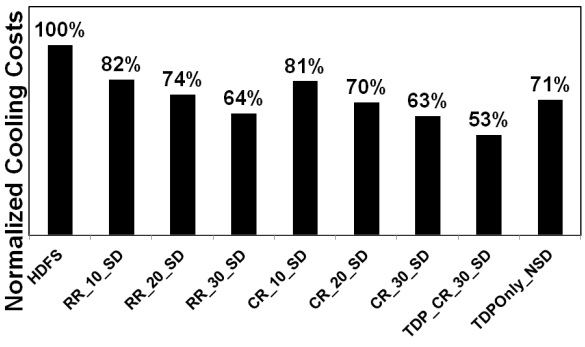

Figure 4.43: **Normalized cooling costs with respect to baseline HDFS of the rack-level and cluster-level zone partitioning schemes in GreenHDFS.**

4.10.7.3 Thermal-Aware Zone Partitioning Schemes

Next, we compare the cooling energy costs reduction possible in GreenHDFS with different zone partitioning schemes covered in Sections 4.5.2.1 and 4.5.2.2. As shown in Figure 4.43, the cooling costs reduction is very similar in rack-level zone partitioning and cluster-level zone partitioning. Since, rack-Level zone partitioning has performance advantages over cluster-Level zone Partitioning with respect to file migration and file writes; it can be used without any cooling costs trade-offs. However, the compute (i.e. server) operating energy costs are higher in case of rack-level zone partitioning compared to cluster-level zone partitioning; hence, a trade-off between performance, and compute energy costs savings need to be made in deciding which scheme to use.

4.10.7.4 Thermal-Aware Zoning Combined with Thermal-Aware File Placement

In this section, GreenHDFS does thermal-aware file placement in addition to thermal-aware zoning. As shown in the Figure 4.43, GreenHDFS realizes much higher cooling energy costs savings by doing thermal-aware file placement in the $Active$ zone in addition to thermal-aware zoning. We also compare a variant of GreenHDFS *TPDOnly* which doesn't do any file migration, zon-

ing, or scale-down and only does predictive, thermal-aware file placement covered in the Section 5.3.2.2 in the entire cluster. *TPDOnly* is capable of cutting down cooling energy costs by 29% which is commendable and very encouraging as *TPDOnly* doesn't do any scale-down. Sometimes, enterprises are vary of scaling-down their servers and for such enterprises, *TPDOnly* will work well. Thermal-aware file placement used in conjunction with thermal-aware zoning as represented by TPDOnly_CR_30 realizes even high cooling energy costs savings of 47%. If scale-down is permissible, a combination of thermal-aware file placement and thermal-aware zoning should be used to achieve high cooling energy costs savings.

4.10.7.5 Additional Cooling Energy Costs Savings

Both Thermal-aware file placement and thermal-aware zoning/scale-down result in a reduction in the temperature of the exhausted air entering the inlet of the computer room air conditioner (CRAC); hence, CRACs can be operated at a higher set point temperature T_{ac}. Increase in T_{ac} increases the efficiency (COP) of the CRAC allowing it to remove more heat with less work and thus, reduces cooling energy costs. For example, operating the CRACs at 5^{o}C higher supply temperature of 20^{o}C increases COP from 5 to 6, resulting in additional cooling costs savings. Thus, the cooling energy costs savings shown in the earlier sections will actually be higher once increase in the CRAC's efficiency is incorporated.

Table 4.8: **Table of Notations Used Total Cost of Ownership Analysis**

Variable	Description	Units
$\mathfrak{D}_a$	Data center amortization per month	\$/Watts
$\mathfrak{D}_c$	Data center capex	\$
$\mathfrak{D}_p$	Data center amortization period	Years
$\mathfrak{D}_i$	Data center interest per month	\$/Watts
I_{Annual}	Annual interest rate	
y	Loan years	Years
$\mathfrak{S}_a$	Server amortization per month	\$/Watts
$\mathfrak{S}_c$	Server capex	\$
$\mathfrak{S}_p$	Server amortization period	Years
$\mathfrak{S}_i$	Server interest per month	\$/Watts
$\mathfrak{S}_{lifetime}$	Server life time	Years
P_i	Power consumption of server i	Watts
$P_{peak,i}$	Peak power consumption of server i	Watts
$P_{month,i}$	Power consumption of server i per month	Watts
$\mathfrak{R}$	Average power to peak power ratio	
$PUE_{overhead}$	PUE overhead	\$/Watts
PUE	Power usage effectiveness	
TCO_{3-year}	Total cost of ownership	\$
E	Electricity cost per Kilowatt Hour	\$

4.11 Total Cost of Ownership Analysis

Total cost of ownership (TCO) of a data center has two main components [68]. Capex refers to the investments that are made upfront and then, are depreciated over a period of time. Expenses such as the cost of purchase of the data center land, building construction, and hardware acquistion fall under the capex category of expenses. Opex refers to the recurring costs of the data center such as operating energy costs, equipment and data center maintainence costs, and salaries of the data center personnel. Some of the costs such as operating energy costs depend on the geographic location of the data center as the underlying electricity rates, property values, and personnel costs differ by region.

TCO = data center depreciation + data center opex + server depreciation + server opex

4.11.1 Assumptions

Our calculations are based on the total cost of ownership analysis done by Barroso et. al. [68], and we make the following assumptions in our total cost of ownership analysis similar to the total cost of ownership analysis done at Google [11]:

- The interest rate is 12%, and servers are financed with a 3-year interest-only loan.
- The cost of data center construction is \$15/W amortized over 12 years.
- Datacenter opex is \$0.04/W per month.
- Server lifetime is 3 years, and server repair and maintenance is 5% of capex per year.
- The server's average power draw is 75% of peak power.
- PUE value is 2.

Power Usage Effectiveness (PUE) is used as an indicator of the energy-efficiency of data centers and is the ratio of the total building power to the IT power (i.e., power consumed by the actual computing equipment such as servers, network, and storage). For example, a data center with a PUE of 2.0 uses one watt of additional power for every watt used to power the IT equipment. Majority of the data centers still have a high PUE of 1.8 as per a 2011 survey of 500 data centers [2], and cooling system's power overhead is the major culprit behind the high PUE values. Hence, a rounded-off PUE value of 2 is used in the analysis. The total cost of analysis is based on the energy rates in California as maximum number of data centers in United States are based in California [12, 13].

The following equations help calculate the total cost of ownership of the data center. The notations used in the equations are given in Table 4.8.

$$\mathfrak{D}_a = (\mathfrak{D}_c/\mathfrak{D}_p)/12 \tag{4.34}$$

$$\mathfrak{D}_i = (\mathfrak{D}_a \cdot I_{Annual})/y \tag{4.35}$$

$$\mathfrak{S}_a = ((\mathfrak{S}_c/\mathfrak{S}_{lifetime}))/12)/P_i \tag{4.36}$$

$$\mathfrak{S}_i = (\mathfrak{S}_a \cdot I_{Annual})/y \tag{4.37}$$

$$P_{month,i} = (\mathfrak{R} \cdot ElectricityRate \cdot 24 \cdot 30)/1000 \tag{4.38}$$

$$PUE_{Overhead} = P_{month,i} \cdot (PUE - 1) \tag{4.39}$$

$$TCO_{3-year} = (\mathfrak{D}_a + \mathfrak{D}_i + \mathfrak{S}_a + \mathfrak{S}_i + P_{month,i} + PUE_{Overhead}) \cdot P_{peak,i} \cdot 3 \tag{4.40}$$

4.11.2 Baseline TCO Analysis

We first do a total cost of ownership analysis of a baseline Big Data analytics cluster (i.e., cluster with no GreenHDFS deployment). In the baseline case, we assume that each server in the cluster has two quad core, Intel Xeon X5400 CPU, eight DRAM DIMM, and four Seagate Barracuda ES.2 1TB Hard Drives. The acquisition cost of the baseline server is assumed to be $2000. Table 4.9 shows the per-component breakdown of the server power consumption of the baseline server. Table 4.10 gives a breakdown of the storage costs and Table 4.11 gives a breakdown of the server costs used in the analysis. No energy management is performed in the cluster and the servers are in an active power state all the time.

Table 4.9: **Server Power Breakdown**

Component	Active Power (W)	Idle Power (W)	Sleep Power (W)
Xeon CPU (Quad core, Intel Xeon X5400)	80-150	12.0-20.0	3.4
Atom CPU (Singe core, Intel Atom Z560)	2.5	0.01	
DRAM DIMM	3.5-5	1.8-2.5	0.2
NIC	0.7	0.3	0.3
SATA HDD (Seagate Barracuda ES.2 1TB)	11.16	9.29	0.99
SSD (Intel X25-E 64GB)	2.6	0.06	NA
PSU	50-60	25-35	0.5
Baseline server (2 Xeon CPU, 8 DRAM DIMM, 4 1TB HDD)	445.3	132.46	13.16
ActiveSSD server (2 Xeon CPU, 8 DRAM DIMM, 256GB SSD)	411.1	95.54	9.44
ActiveHDD-4 server (2 Xeon CPU, 8 DRAM DIMM, 4 1TB HDD)	445.3	132.46	13.16
ActiveHDD-8 server (2 Xeon CPU, 8 DRAM DIMM, 8 1TB HDD)	490.0	169.62	17.12
InactiveXeon-12 server (2 Xeon CPU, 8 DRAM DIMM, 12 1TB HDD)	534.6	206.78	21.08
InactiveAtom-12 server (1 Atom CPU, 2 DRAM DIMM, 12 1TB HDD)	207.12	151.79	13.09

4.11.3 GreenHDFS TCO Analysis

In this section, we do a total cost of ownership analysis of a Big Data analytics cluster managed by GreenHDFS. We assume a hybrid cluster whereby the servers differ in their storage quantity, storage type and hence, server price. We consider five server types in the total cost of ownership calculations:

- **ActiveSSD:** $ActiveSSD$ zone server with 256GB solid state drive (SSD), two quad core Intel Xeon X5400 CPU, and eight DRAM DIMM. To come up with the server price, we add the price differential of the storage to the baseline server price of $2000 with four 1TB

Table 4.10: **Storage Costs**

Characteristics	Unit	SATA	SSD
Storage		Seagate Barracuda ES.2 1TB	Intel X25-E 64GB
Capacity	TB	1	0.064
Price / GB	$ / GB	0.12	12

Table 4.11: **Server Costs**

Characteristic	ActiveSSD	ActiveHDD-4	ActiveHDD-8	InactiveXeon-12	InactiveAtom-12
Storage (TB)	0.256	4	8	12	12
Total Power (W)	411.1	445.3	490.0	534.6	207.12
Cost ($)	$ 4,592	$ 2,000	$ 2,480	$ 2,960	$ 1,539

disks. Assuming a price of SSD equal to $12/GB, 256GB of SSD increase the server price to $4592.

- **ActiveHDD-4:** $Active$ zone server with four Seagate Barracuda ES.2 1TB Hard Drives, two quad core Intel Xeon X5400 CPU, eight DRAM DIMM, and costs $2000 to acquire. The hardware configuration and acquisition cost of the server is the same as that of the baseline cluster server.

- **ActiveHDD-8:** $Active$ zone server with eight Seagate Barracuda ES.2 1TB Hard Drives, two quad core Intel Xeon X5400 CPU, eight DRAM DIMM, and costs $2,480 to acquire. To come up with the server's acquisition cost, we add the price differential of the additional storage (four SATA disks) to the ActiveHDD-4 server price of $2000. Assuming, a price of $0.12/GB, additional four 1TB disks increase server price to $2,480.

- **InactiveXeon-12:** $Inactive$ zone server with twelve Seagate Barracuda ES.2 1TB Hard Drives, two quad core Intel Xeon X5400 CPU, and eight DRAM DIMM. To come up with the $Inactive$ zone server price, we add the price differential of the additional storage (eight SATA disks) to the $Active$ zone server price of $2000. Assuming, a price of $0.12/GB, additional eight 1TB disks increase server price to $2,960.

- **InactiveAtom-12:** In this flavor of GreenHDFS, servers in $Inactive$ zone have one low-power, low-performance Atom Z560 CPU instead of high-power, high-performance two quad core Intel Xeon X5400 CPU and have two DRAMs instead of eight DRAMs. Using cheaper Atom processor instead of Xeon processor brings the server acquisition cost down

to $1539.

Table 4.12 gives a breakdown of the server's 3-yr total cost of ownership TCO_{3yr} value for various server options listed above. The equations and the assumptions used in the analysis are presented in Section 4.11.1.

Next, to understand the total cost of ownership savings resulting with the scale-down of the servers in GreenHDFS, we need to calculate the 3-yr total cost of ownership of a server when the server is in a scaled-down power state. We refer to the 3-yr TCO in the scaled-down state as 3-yr $TCO_{3yr,SD}$. In the scaled-down state, the only component that doesn't factor in the server's total cost of ownership is the server operating energy cost.

3-yr energy cost of a server, denoted by $Costs_{3-yr}$ is calculated as follows:

$$Costs_{3-yr} = 3 \cdot 8760 \cdot E \cdot P_i/1000 \cdot PUE \tag{4.41}$$

Next, we introduce a metric called $ETCORatio$. $ETCORatio$ is defined as the ratio of server' 3-year energy costs to server's 3-year total cost of ownership. $ETCORatio$ sheds light on the contribution of the server's operating energy costs on the total cost of ownership of the server. In California, $ETCORatio$ is 30% with PUE value of 2. This means that in California, $TCO_{3yr,SD}$ is 70% of the normal total cost of ownership. We refer to the ratio of the scaled-down total cost of ownership to the normal total cost of ownership as $SNSRatio$.

We considered various GreenHDFS configurations in this section as described below:

- 7030_n: 70% of servers in the cluster were allocated to the $Active$ zone and 30% to the $Inactive$ zone. The $Active$ zone servers are in the ActiveHDD-4 configuration and the $Inactive$ zone servers are in the InactiveXeon-12 configuration. n% of the servers in the $Inactive$ zone are in a scaled-down state at any point in time. *n* ranges from 10% to 100%.

- 8020_n: 80% of servers in the cluster were allocated to the $Active$ zone and 20% to the $Inactive$ zone. The $Active$ zone servers are in the ActiveHDD-4 configuration and the $Inactive$ zone servers are in the InactiveXeon-12 configuration. Where, n% of the servers in the $Inactive$ zone are in a scaled-down state at any point in time and *n* ranges from 10% to 100%.

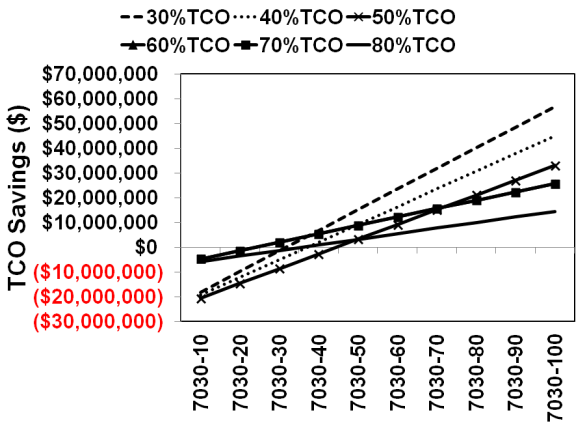

Figure 4.44: **TCO Savings with GreenHDFS configurations and for various percentages of scaled-down server TCO with respect to non-scaled-down server TCO ($SNSRatio$). In California, $SNSRatio$ is 70% for PUE values of 2. $SNSRatio$ of 70% with 7030_95 configuration yields 3-year TCO savings of $22.3 million compared to baseline cluster with no GreenHDFS deployment.**

Figures 4.44 and 4.45 show the TCO savings that can be observed by scaling down an increasing percentage of servers in the $Inactive$ zone. We have assumed the total number of servers across the Hadoop clusters to be 38000. This is the number of servers deployed across all the production Hadoop clusters at Yahoo!. There can be significant variation in the ratio of the total cost of ownership of a scaled-down server vs. a non-scaled-down total cost of ownership of a server ($SNSRatio$). Hence, we plotted total cost of ownership savings for various values of the $SNSRatio$ for a server. If the scaled-down total cost of ownership is 80% or above of the non-scaled-down total cost of ownership, zoning will not result in any total cost of ownership savings. Scaled-down total cost of ownership values lower than 80% of non-scaled-down total cost of ownership result in significant savings. For example, in California, scaled-down total cost of ownership is 70% of the non-scaled-down total cost of ownership. In previous work on GreenHDFS, we found that $> 90\%$ of the servers are in a scaled-down state at any point in time. As shown in the Figure 4.44, 7030_95 configuration's 70% $SNSRatio$ curve shows a saving of approximately $6 million in 3-yr total cost of ownership. Figure 4.45 shows the total cost of ownership savings in the 8020 configuration. 8020_95 configuration's 70% $SNSRatio$ curve shows a saving of approximately $4 million in 3-yr total cost of ownership.

Figure 4.46 shows the total cost of ownership savings that can be observed by scaling down an increasing percentage of servers in the $Inactive$ zone with servers in *InactiveAtom* configuration,

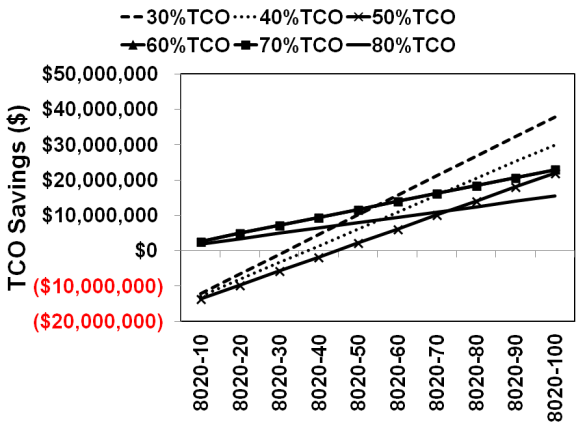

Figure 4.45: **TCO Savings for various GreenHDFS configurations and for various percentages of scaled-down server TCO with respect to non-scaled-down server TCO ($SNSRatio$). In California, $SNSRatio$ is 70% for PUE values of 2. $SNSRatio$ of 70% with 8020_95 configuration yields 3-year TCO savings of $20.7 million compared to baseline cluster with no GreenHDFS deployment.**

i.e., using low-cost, low-power Intel Atom processor instead of expensive, high-power Intel Xeon processors. We have assumed the total number of servers across the Hadoop clusters to be 38000.

Server costs are increasingly going down and energy costs are either equal to or more than the initial acquisition costs of the servers. In that scenario, the scaled-down TCO will be an even smaller percentage of the non-scaled-down TCO. GreenHDFS will result in even higher TCO savings in that scenario.

4.11.4 To Zone or Not to Zone

In this section we compare GreenHDFS with different baseline cluster configurations whereby only 80% of the servers with slightly higher storage are allocated in the cluster instead of 100% servers, dormant data is either stored in the cluster itself or on an external cloud, and finally, during peak loads, cloud-bursting to an external cloud is used to take care of the excess load. On the other hand, GreenHDFS uses the in-house $Inactive$ zone servers for consolidating and storing the dormant data. The $Inactive$ zone servers are scaled-down during average load conditions,

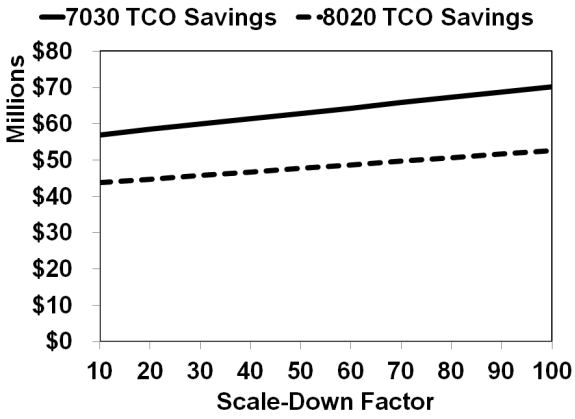

Figure 4.46: **TCO Savings using low-power, low-cost Atom processor in $Inactive$ zone. In California, $SNSRatio$ is 70% for PUE values of 2. $SNSRatio$ of 70% with 7030_95 configuration yields 3-year TCO savings of $68.7 million compared to baseline cluster with no GreenHDFS deployment. $SNSRatio$ of 70% with 8020_95 configuration yields 3-year TCO savings of $51.6 million compared to baseline cluster with no GreenHDFS deployment.**

and are woken up during peak load scenario. 20% of the cluster servers are assigned to the $Inactive$ zone.

- **80_SATA4**: In this option, only 80% servers of the baseline cluster are allocated to the cluster. All the servers in the cluster are assumed to be in the ActiveHDD-4 configuration.

- **80_SATA8_Burst_OnDemand**: In this option, servers in the cluster have a higher number of eight 1TB SATA drives each to allow higher amount of overall storage capacity in the cluster as in the ActiveHDD-8 configuration, and the dormant data is stored in the cluster itself. The peak loads are handled by bursting to an external cloud. The compute capacity on the external cloud is obtained on-demand. The cost of buying compute capacity on-demand is lower compared to buying reserved compute capacity; however, availability at all times is not guaranteed.

- **80_SATA8_Burst_Reserved**: In this option, servers in the cluster have a higher number of eight 1TB SATA drives each as in the ActiveHDD-8 configuration, the dormant data is stored in the cluster itself. The peak loads are handled by bursting to an external cloud. The compute capacity on the external cloud is kept reserved year-around to ensure that there is no dearth of computational power when the peak loads do occur. However, the cost

of buying reserved compute capacity is higher as shown in the Amazon's EC2 pricing [5].

- **GreenHDFS_Atom**: In this flavor of GreenHDFS, servers in $Inactive$ zone have one low-power, low-performance Atom Z560 CPU instead of high-power, high-performance two quad core Intel Xeon X5400 CPU and have two DRAMs instead of eight DRAMs as in the InactiveAtom-12 configuration. During average load, these servers are scaled-down and during peak load, the *Inactive* are woken up and their compute capacity is used to handle the additional load. Thereby, there is no need to do cloud-bursting to an external cloud during peak loads.

- **GreenHDFS_Xeon**: This flavor of GreenHDFS has the same hardware configuration and server total cost of ownership as covered in Section 4.11.3. During average load, these servers are scaled-down and during peak load, the *Inactive* are woken up and their compute capacity is used to handle the additional load. Thereby, there is no need to do cloud-bursting to an external cloud during peak loads.

As shown in Figure 4.47, the TCO costs with GreenHDFS are lower than even the clusters with only 80% servers. The TCO costs of GreenHDFS_Atom are lower than GreenHDFS_Xeon, making it the most cost-efficient solution. In addition, GreenHDFS also supports higher overall storage capacity than baseline cluster as shown in the the Figure 4.48.

Table 4.12: **TCO Calculation per Server**

	ActiveSSD	ActiveHDD-4	ActiveHDD-8	InactiveXeon-12	InactiveAtom-12
cost of electricity ($/kWh)	$0.106	$0.106	$0.106	$0.106	$0.106
interest rate	12%	12%	12%	12%	12%
DC capex ($/W)	15	15	15	15	15
DC amortization period (years)	12	12	12	12	12
DC opex ($/kW/mo)	$0.04	$0.04	$0.04	$0.04	$0.04
PUE	2	2	2	2	2
server capex	$4,592	$2,000	$2,480	$2,960	$2,960
server lifetime (years)	2	2	2	2	2
server W	411.1	445.34	490.0	534.62	207.12
server opex	5%	5%	5%	5%	
server avg power relative to peak	75%	75%	75%	75%	75%
$/W per month					
DC amortization	$0.104	$0.104	$0.104	$0.104	$0.104
DC interest	$0.093	$0.093	$0.093	$0.093	$0.093
DC opex	$0.040	$0.040	$0.040	$0.040	$0.040
server amortization	$0.465	$0.187	$0.141	$0.231	$0.206
server interest	$0.091	$0.037	$0.028	$0.045	$0.040
server opex	$0.023	$0.009	$0.007	$0.012	$0.010
server power	$0.057	$0.057	$0.057	$0.057	$0.057
PUE overhead	$0.057	$0.057	$0.057	$0.057	$0.057
total	$0.738	$0.507	$0.526	$0.543	$0.608
3-yr TCO	$10,917	$8,121	$9,284	$10,446	$4,536

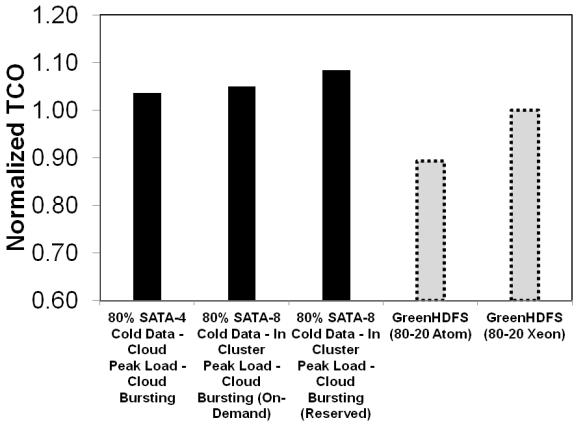

Figure 4.47: **3-yr TCO savings normalized to the GreenHDFS_Atom TCO. GreenHDFS_Atom TCO is lower than even cluster configurations with only 80% servers allocated.**

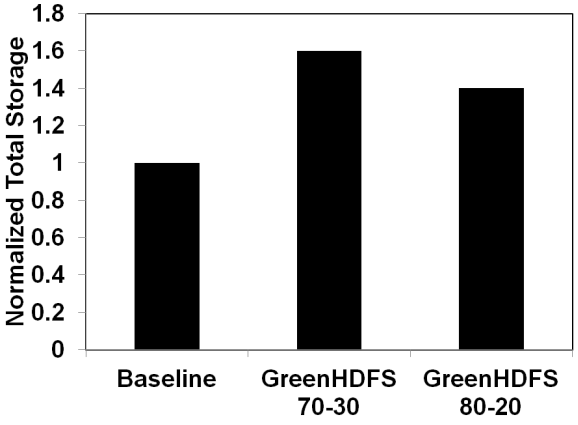

Figure 4.48: **Normalized total storage capacity in the cluster with baseline and GreenHDFS Big Data analytics cluster.**

CHAPTER 5

DATA-CENTRIC COOLING ENERGY MANAGEMENT

GreenHDFS takes a data-centric, cyber-physical energy management approach to reduce cooling energy costs in the Big Data analytics clouds. On the physical-side, GreenHDFS is cognizant of the uneven thermal-profile in data centers due to complex airflow patterns and varying ability of the cooling system to cool different parts of the data center. On the cyber-side, GreenHDFS is cognizant of the fact that files in the cluster differ in their computational job arrival rates, file sizes, and evolution life spans. GreenHDFS combines its data-semantics knowledge on the *cyber-side* with the thermal-profile knowledge of the cluster on the *physical-side* to do proactive cyber-side thermal-aware file placement.

In this chapter, in Section 5.1 we provide the motivation for thermal-aware file placement with the help of an illustrative example and analysis of a real-world Big Data analytics cluster. In Section 5.3.2.2, we present the various thermal-aware file placement heuristics in place in GreenHDFS. In Section 4.6.1, we present an in-depth look at the predictive modeling in GreenHDFS. In Section 5.4, we present the evaluation results of the various thermal-aware file placement heuristics.

5.1 Thermal-Aware File Placement Motivation

In this section, we present the motivation behind GreenHDFS's thermal-aware file placement. We start with the thermal and power model of the cluster to provide background to the motivation discussion.

5.1.1 Thermal and Power Model

A server i in the cluster can be modeled as a heat source (i.e., CPU, Disk, DRAM, Network), where $T_{i,t}^{in}$ is the inlet temperature, $T_{i,t}^{out}$ is the outlet temperature, and $T_{i,t}$ is the temperature of server i at time *t*. The power $P_{i,t}$ consumed by a server i with computational load $L_{i,t}$ at time *t* can be stated as a linear function of $L_{i,t}$ [67]:

$$P_{i,t} = w_{i_1} \cdot L_{i,t} + w_{i_2}, \tag{5.1}$$

where w_{i_1} and w_{i_2} are coefficients expressed in joules/sec units. Coefficient w_{i_1} is the difference between the peak power consumption and the static power consumption (i.e., when the server is at idle utilization with no computational load), and coefficient w_{i_2} gives the static power consumption. We assume a homogeneous cluster in this paper, whereby all servers have the exact same hardware. Hence, the values of w_{i_1} and w_{i_2} are the same for each server.

At steady state, the temperature of a server *i*, $T_{i,t}$ is a function of the inlet temperature $T_{i,t}^{in}$ and the power $P_{i,t}$ [93]:

$$T_{i,t} = T_{i,t}^{in} + c_i \cdot P_{i,t} \tag{5.2}$$

where, constant c_i expressed in kelvin.sec/joules units, is a factor of server's heat exchange rate, it's air flow, and heat capacity density of the air. In a data center with free-cooling or with hot- and cold-aisle air containment, the difference in the inlet temperatures of the servers is predominantly dependent on the distance of the server from the vents or the cooling unit. In traditional data centers without containment or economization, hot air recirculation and air bypass result in a higher difference in inlet temperatures. We assume $T_{i,t}^{in}$ to be known empirically for each server. Thus, temperature $T_{i,t}$ can be modeled as a linear function of $L_{i,t}$, i.e.

$$T_{i,t} = T_{i,t}^{in} + c_i \cdot (w_{i_1} \cdot L_{i,t} + w_{i_2}) \tag{5.3}$$

It is important to ensure that a server i's temperature $T_{i,t}$ remains less than T_{max} at all times *t*, where T_{max} is the reliability driven upper-bound on the temperature that can be tolerated by a

server as specified in its data sheet; exceeding T_{max} results in higher hardware failure rates.

At steady-state, the temperature of the hot air exhausted from the server i's outlet, $T_{i,t}^{out}$ is a function of the power $P_{i,t}$ being consumed at server i at load $L_{i,t}$, the temperature of the server $T_{i,t}$, and the heat exchange rate of the server i, θ_i expressed in joules/kelvin.secs units as shown below:

$$T_{i,t}^{out} = T_{i,t} - \frac{P_{i,t}}{\theta_i} \tag{5.4}$$

The hot air exhausted by the servers in the cluster enters the cooling subsystem which then cools the hot air to a temperature T_{ac} which is the air supply temperature set point of the cooling subsystem. The cooling power consumption is proportional to the heat removed by the cooling subsystem, which is proportional to the difference in the temperature $T_{c,t}^{out}$ of the hot exhausted air entering the cooling subsystem and the temperature T_{ac}. The cooling power consumption is given by:

$$P_{c,t} = \frac{c_{air} \cdot f_{ac} \cdot \left(T_{c,t}^{out} - T_{ac} \right)}{COP}, \tag{5.5}$$

where, c_{air} is the heat capacity constant, f_{ac} is the flow rate of the cooling unit, and $|S|$ is the number of servers in the cluster. The efficiency of cooling unit is characterized by its coefficient of performance *(COP)* defined as the ratio of the amount of heat removed by the cooling device and the work required to do so.

5.1.2 File Model

Big Data analytics frameworks, such as Map Reduce, comprise of a distributed file system underneath, which distributes the file chunks and their replicas across the cluster for fine-grained load-balancing and high parallel data access performance. Each file chunk is typically 64-128 MB in size [61, 84] and is replicated three-ways for resiliency and fault-tolerance.

Let f_j^k = set of three replicas of chunk k of file j, $1 \leq k \leq \hat{n}_j$, $1 \leq j \leq |Z|$, where $|Z|$ is the number of files in the cluster and $\hat{n}_j$ is the number of chunks into which file j is divided. Denote $\tilde{f}_j = \{f_j^1, f_j^2, \cdots, f_j^{\hat{n}_j}\}$ to be the set of all chunks of file j and $Z = \{\tilde{f}_1, \tilde{f}_2, \cdots, \tilde{f_{|Z|}}\}$ to be the

set of all the file chunks in the cluster.

Let $c_{j,i}^{k}$ be the replica of chunk k of file j assigned to server i. Let $C_{j,i}$ be the set of file j's chunk replicas assigned to server i, i.e., $C_{j,i} = c_{j,i}^{1} \cup c_{j,i}^{2} \cup \cdots \cup c_{j,i}^{n_{j,i}}$, where $n_{j,i}$ is the number of file $\tilde{f}_j$'s chunk replicas assigned to server i. Let $\tilde{C}_i$ be the set of chunk replicas of different files assigned to server i. $\tilde{C}_i$ is equal to a set of the set of file chunk replicas assigned to server i, i.e., $\tilde{C}_i = C_{1,i} \cup C_{2,i} \cup \cdots \cup C_{N_i,i}$, where N_i is the number of the files whose chunks are assigned to server i.

Let $\ddot{C} = \{\tilde{C}_1, \tilde{C}_2, \cdots, \tilde{C}_{|S|}\}$. S is the set of servers in the cluster, $|S|$ is the number of servers in the cluster, and $S = \{1, 2, \cdots, |S|\}$. The file placement function π is defined as $\pi : Z \rightarrow \ddot{C} \times S$ which takes the set of all file chunks Z as its input and produces an ordered set $\ddot{C}$ as output.

5.1.3 Importance of Thermal-Aware File Placement with Big Data

A typical Big Data analytics job targeted to a file $\tilde{f}_j$ is split into multiple sub-jobs equivalent to the number of chunks $\hat{n}_j$ of the file. Each sub-job is sent to the server hosting its target file chunk in interest of data-locality. Let $\tilde{J}_{k,j}(t)$ be the computational job load to a chunk $c_{j,i}^{k}$ at a time interval t. The cumulative computational load $L_{i,t}$ on a server i is a result of the load arising from the sub-jobs targeted to the various file chunks stored on the server i, which in turn depends directly on the placement function $\pi : Z \rightarrow \ddot{C}, \pi(f_j^k) = c_{j,i}^{k} \in \tilde{C}_i, \tilde{C}_i \subset \ddot{C}, i \in S$. Hence we have:

$$L_{i,t}(\pi(Z), t) = L_{i,t}(\tilde{C}_i, t) = \sum_{j=1}^{N_i} \sum_{k=1}^{n_{j,i}} \tilde{J}_{k,j}(t) \tag{5.6}$$

The compute energy costs in Equation (5.1) and cooling energy costs in Equation (5.5) depend directly on the computational load $L_{i,t}$, which in turn depends directly on the file placement π. This brings to light the importance of thermal-aware file placement π. To ensure no adverse impact to the performance, cooling energy management techniques for Big Data analytics clusters need to treat data as a *first-class* object in computing, Files need to be placed first in the cluster in a thermal- and energy-aware manner so that the computational jobs can then automatically enjoy cooling energy costs savings and high data-local performance by following files placed in a thermal-efficient manner.

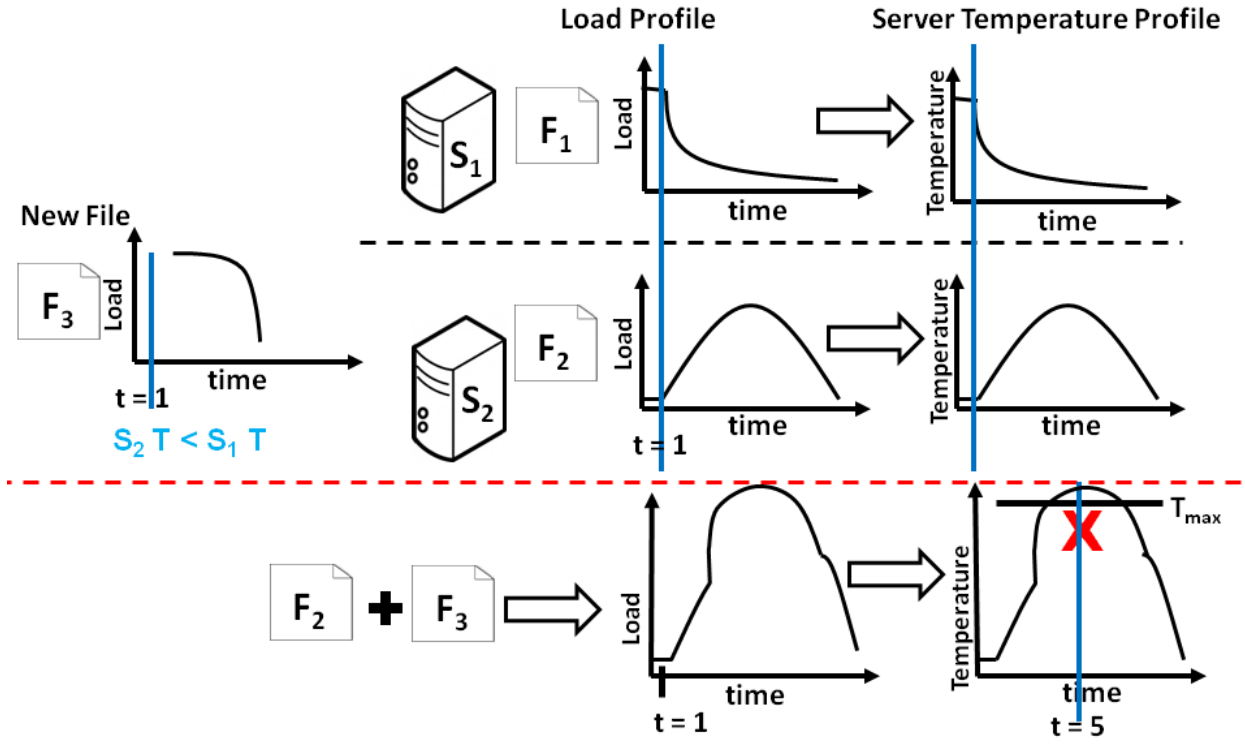

Figure 5.1: **Thermal-aware file placement challenge.**

5.1.4 Challenges

A naive way to do the thermal-aware file placement is to place an incoming file $\tilde{f_j}$ on the server i with the lowest temperature at that time instant. However, this naive file placement may result in a much higher thermal-profile than anticipated and may even result in a server exceeding T_{max}. In the example illustrated in Figure 5.1, let there be two servers S_1 and S_2 in the cluster. At file f_3's arrival at $t = 1$, server S_2's temperature $T_{2,t=1}$ is lower than server S_1's temperature $T_{1,t=1}$ and the naive file placement chooses server S_2 as the destination for file f_3. However, S_2 is a bad choice for f_3. The cumulative load of files f_3 and f_2 (f_2 is already resident on the server S_2), results in a very high computational load on S_2 at t=5. As a result, S_2's temperature exceeds T_{max}, impacting thermal-reliability of S_2 and contributing to higher cooling energy costs.

On the contrary, f_3 is a much better fit for S_1 as f_3's load-profile is complementary to f_1's load-profile. The cumulative load-profile of f_3 and f_1 is lower and doesn't result in S_2 exceeding T_{max} at any point in time. Thus, there is a need for more sophisticated and predictive thermal-aware

file placement algorithms and heuristics that can somehow glean information about the "future" computational load profile of an incoming file and the "future" thermal-profile of the servers in the cluster and then place the file "now" on the most thermally-conducive server in the "future".

Thermal-aware file placement is very challenging and raises several *hard* questions:

- Is it even possible to have an optimal solution for thermal-aware file placement or else, is there a need to rely on approximate algorithms?
- How and what aspects of the computation job-profile of the file should be predicted to gain futuristic knowledge to guide the file placement?
- How should the files be placed on servers in a way that ensures that the cumulative computational load of the files results in a thermal- and cooling-efficient temperature-profile of the server?
- Will the technique ensure thermal reliability and lower thermal-profile in new data centers with air- or water-side economization and aisle containment?
- Will the technique reduce cooling energy costs in traditional data centers without air containment or economizer modes?
- How sensitive is the technique to cluster parameters such as file size skew, delta in inlet thermal-profile, utilization, file creation rate and sampling rate?

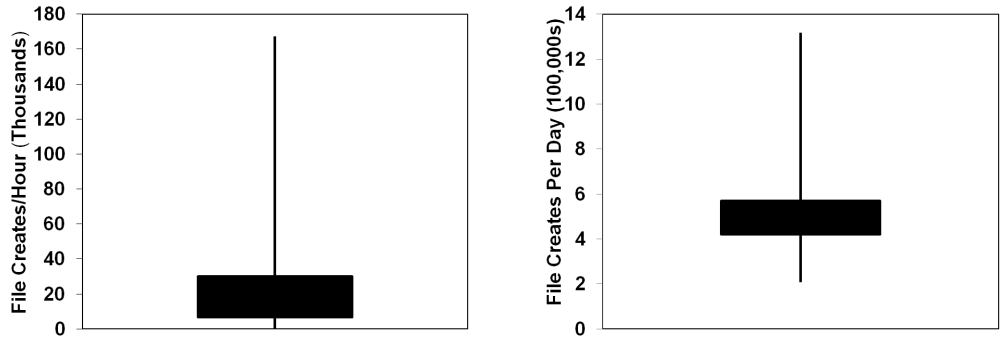

Figure 5.2: **File creates at hour, and daily granularity in production real-world Big Data analytics cluster at Yahoo!.**

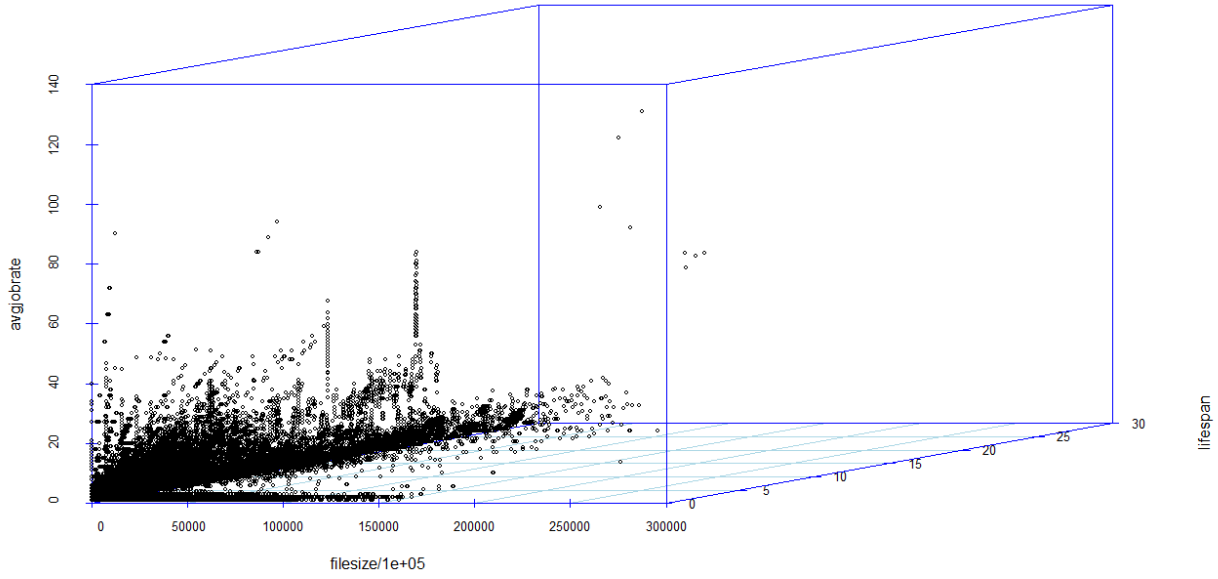

Figure 5.3: **3-D scatter plot of file average job rate/min, file life span, and file size in the production Big Data analytics cluster at Yahoo!. Majority of the files have small life spans and smaller sizes. The job rates are higher at the beginning of a file's life span.**

5.1.5 Importance in Production Systems

There are two possibilities for doing thermal-aware file placement: 1) at the creation time of the file itself, and 2) in the middle of file's lifetime in response to a triggering thermal-event. Figure 5.2 shows the box plot of file creates happening in a real-world Big Data analytics cluster at Yahoo!; significant number of files are created even at an hour granularity. Figures 5.3 and 5.4 show that files in the same cluster have a higher job rate towards the beginning of their lifetime and the jobs start arriving to majority of the files soon after the file's creation. Hence, to ensure lower cooling costs and to not overheat servers beyond their reliability limit from the very onset of a file's creation, it is important to do thermal- and energy-aware file placement $\pi : Z \rightarrow \ddot{C} \times S$ of files in Z, where Z is the unordered incoming file (chunk) set, $\ddot{C}$ is ordered file (chunk) set and S is the server set, in the Big Data environment at the time of file creation itself.

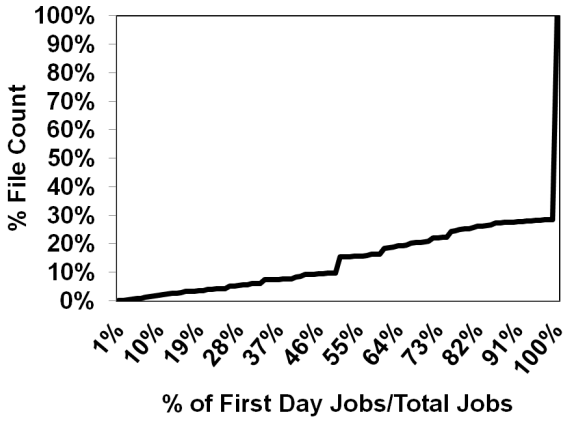

Figure 5.4: **CDF of the percentage of total file count and ratio of jobs received on the first day of a file's life time vs. total number of jobs received by the file in the production Big Data analytics cluster at Yahoo!. 70% of the files receive more than 90% of the total jobs on the first day itself.**

5.1.6 Asymmetric Load Tolerance of Servers

It is important to ensure that a server i's temperature $T_{i,t}$ remains less than T_{max} at all times *t*, where T_{max} is the reliability driven upper-bound on the temperature that can be tolerated by a server as specified in its data sheet. Every server i has an upper-bound value $L_{max,i}$ which is the maximum load that can be subjected to a server i without exceeding the red-line temperature $T_{\max}$. $L_{max,i}$ of a server i can be determined by setting $T_{i,t}$ to T_{max} in Equation 5.3 as shown below:

$$L_{max,i} = \frac{\frac{T_{max} - T_{in}^{i}}{c_i} - w_{i_2}}{w_{i_1}} \tag{5.7}$$

In Equation 5.7, assuming a homogenous cluster, values of w_{i_1}, w_{i_2} and c_i will be same for all the servers in the cluster. However, since the inlet temperature $T_{i,t}^{in}$ is different for all the servers, $L_{max,i}$ is different for each server i. Differences in $T_{i,t}^{in}$ are ubiquitous across data centers. Even in the new data centers which have hot- and cold-aisle containment, inlet temperature differences do arise because of air leakage from the containment, and distance of the servers from the cooling system. In traditionally cooled data centers, the inlet temperatures have a much higher difference because of hot air recirculation and air bypass. Thus, the servers in the cluster can not be loaded equally in interest of thermal-reliability, and there is a need for asymmetric thermal-aware file placement as illustrated in the Figure 5.5.

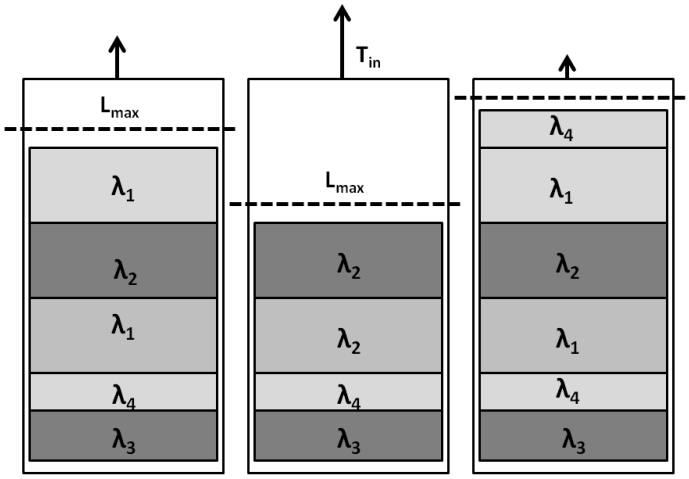

Figure 5.5: **The Servers in the cluster vary in inlet temperatures T_i^{in} (i.e., cooling-efficiencies) resulting in a variation in their load tolerance $L_{max,i}$. Exceeding $L_{max,i}$ may result in exceeding the reliability upper-bound T_{max} of the temperature that can be tolerated by the servers. The dotted line in the figure shows the variations in the $L_{max,i}$ Values of the servers.**

5.2 Problem Statement

We seek a file placement function π^* that minimizes the cooling power consumption $P_{c,t}$ of the data center and total compute power consumption $P_{i,t}$ of the S servers in the data center over time. We assume there exists at least one placement π^* that satisfies Equation (5.8).

$$\arg\left(\min_{\pi}\left(\sum_{t=1}^{T}\left(P_{c,t}+\sum_{i=1}^{|S|}w_{i_1}L_{i,t}(\pi)+w_{i_2}\right)\right)\right) \tag{5.8}$$

s.t.

$$L_{i,t}(\pi)\leq L_{max,i}\quad \forall i,t, \tag{5.9}$$

where $|S|$ is the number of servers in the cluster, $TP=[1,2,\cdots,T]$ is the overall life time of the cluster for which the optimal value of placement π^* is to be determined, and $L_{max,i}$ is the maximum load that can be subjected to a server i without exceeding the red-line temperature $T_{\max}$ as discussed in Section 5.1.6.

The problem of optimally placing a collection of files in a cluster can be decomposed into two subproblems. Let $\tilde{J}_j(t)$ be the function giving the computational load on a given file at a time

interval t. The first subproblem relates to the prediction of each file's future computational load profile $\tilde{J}_j(t)$. The second subproblem relates to the actual optimal placement of files in the cluster subject to energy costs, temperature and capacity constraints of the cluster servers, *given* knowledge of each file's future computation job profile $\tilde{J}_j(t)$. The first problem can be formulated as a machine learning problem and we will see several proposed solutions later in this section for estimating the $\tilde{J}_j(t)$ function. Given such a $\tilde{J}_j(t)$, let's now focus on the second file placement subproblem, i.e., finding optimal π function. Unfortunately, file placement problem is known to be NP-Hard [53]. The computational complexity of an idealized thermal-aware file placement problem suggests the need for an approximation algorithm to solve the general problem.

In Big Data analytics clusters, the number of servers $|S|$ can be as high as 4000-10,000 and number of files $|Z|$ can be 30-60 millions. Big Data analytics clusters are also very dynamic in nature and a huge number of files are created even at a minute granularity. The pseudo-polynomial dynamic-programming solutions would be computationally expensive in this environment as some of them require $O(|Z|L^T_{\max,i})$ time for each server. One can use approximation strategies like relaxing the file placement problem into a linear programming problem by allowing files to first be 'fractionally' assigned to different servers, followed by some rounding rule to recover a proper assignment. As seen above, one can come up with fairly complex approximation schemes. But in the context of the design and implementation of a robust file system, algorithmic simplicity is usually preferred. In that spirit, we use simple heuristics for thermal-aware file placement in GreenHDFS.

5.3 GreenHDFS Thermal-Aware File Placement Heuristics

GreenHDFS represents each file $\tilde{f}_j$ in the Big Data analytics cloud as a tuple $\tilde{f}_j = \{\mathcal{L}_{active,j}$, $\lambda_j(t)$, $J_{T,j}$, $J_{f,j}$, $s_j\}$, where, $\mathcal{L}_{active,j}$ is the active lifespan in the evolution of a file in days, $\lambda_j(t)$ represents the arrival rate of the computational jobs targeted to file $\tilde{f}_j$ at time interval t, $J_{T,j}$ is the total number of computational jobs received by file $\tilde{f}_j$ during its lifetime, $J_{f,j}$ is the number of computational jobs received by a file on the first day of its creation and s_j is the file size. The servers in the cluster are assumed to be placed in racks and each server i is represented as tuple $= \{\mathfrak{R}_i, \mathfrak{N}_i\}$, where $\mathfrak{R}_i$ is the rack number of the server i and $\mathfrak{N}_i$ is the server number of the server i in the rack $\mathfrak{R}_i$. GreenHDFS's monitoring service collects thermal-map at regular time intervals which comprises of the server temperatures $\{T_{1,t}, T_{2,t}, \cdots, T_{N,t}\}$ of all the servers in the

cluster.

5.3.1 Prediction of Computational Load

To solve the first subproblem in Section 5.2, i.e., prediction of each file's future computational load profile $\tilde{J}_j(t)$, we analyzed clickstream processing workload (most important Big Data analytics workload for Internet services companies) at Yahoo!, and observed that there is a strong correlation between the directory hierarchy in which a file is organized and the file's attributes such as file's size, jobs arrival rate, and active life span. Based on this observation, GreenHDFS generates predictive data models by using supervised learning on historical information present in the file system traces and meta data images as discussed in Section 4.6.1. Section 4.6.1 also details all aspects of the predictor such as training, testing, accuracy of predictions, and evaluation with real-world traces from production cluster [81]. To figure out the statistical correlation between the directory hierarchies and file attributes, predictor uses ridge regression which is a form of a supervised learning with input X (i.e., independent variables) and a response Y (i.e., dependent/response variable). The goal is to learn the correlation (regression) between X and Y. Each file in the training set R is represented as a binary vector $\{0, 1, \cdots, 1\}$, where each subdirectory in the training set is represented as an index in the vector and presence/absence of the subdirectory in the file path is denoted by 0/1. The binary vectors are the independent variables and file's $\mathcal{L}_{active,j}, \lambda_j(t), J_{T,j}, J_{f,j}$ are the dependent/response variables. The predicted values of the response variables are determined at file creation time using the stored ridge coefficients and are used by thermal-aware file placement heuristics.

Now, predicting the exact function $\tilde{J}_j(t)$ for each and every file in the cluster is simply not feasible both in terms of predictor complexity and accuracy. GreenHDFS aims to find predictive, simple, and yet accurate approximation for the function $\tilde{J}_j(t)$. In the Big Data analytics environment, each job translates into one sub-job per file chunk of the file $\tilde{f}_j$. The computational load $\tilde{J}_j(t)$ at time interval t contributed by a file chunk $c^k_{j,i}$ of file $\tilde{f}_j$ residing on server i can be given by $\lambda_j(t) \cdot l$, where l is the average computational job duration. The job duration includes the cpu computation time, and disk data access time. Since these jobs are data-intensive in nature, the overall job time is dominated by the disk access time, which in-turn is dominated by the transfer time. GreenHDFS uses the following two ways to approximate a file's job arrival rate $\lambda_j(t)$.

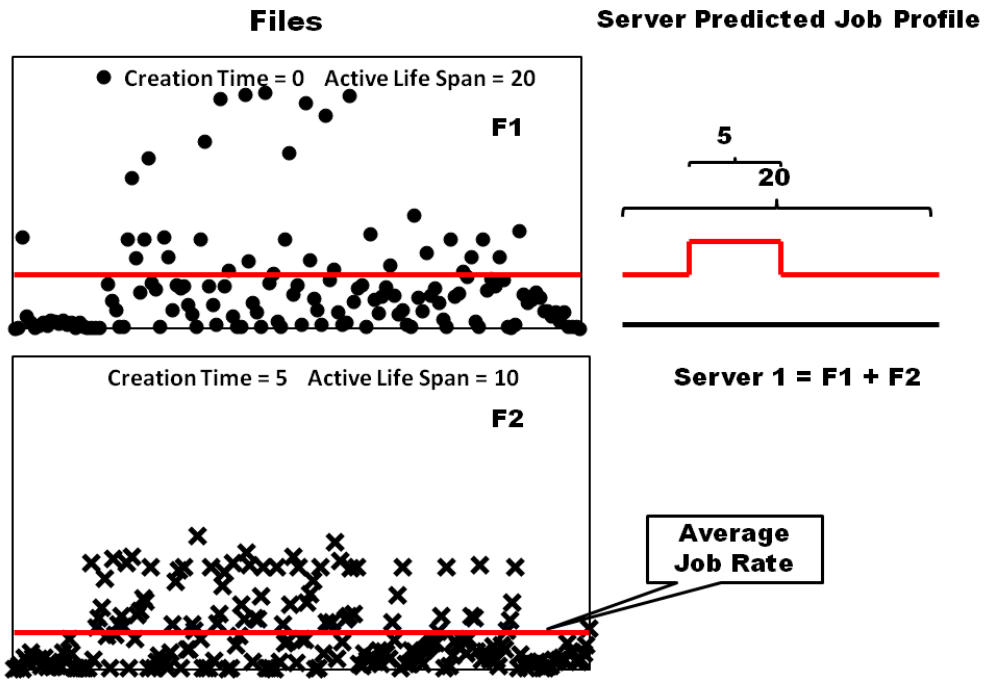

Figure 5.6: **Cumulative predictive job profile of a server with files $F1$ and $F2$ stored in it.**

5.3.1.1 Constant Rate

In this set of heuristics, $\lambda_j(t)$ is approximated by a constant rate across its entire active life span $\mathcal{L}_{active,j}$ and by 0 at the end of the active life span $\mathcal{L}_{active,j}$ as shown in Figure 5.6. The constant rate is either average $\lambda_{avg,j}$, median $\lambda_{med,j}$, or maximum $\lambda_{max,j}$ jobs arrival rate at different time granularities minute/hour/day. The statistical job rates are calculated only for those intervals in the file's active life span that do receive jobs. The intervals that don't receive any jobs are ignored in the calculation; accounting for intervals that don't receive any jobs leads to an almost negligible average and median job arrival rate value which doesn't approximate the computation job profile well and is not conducive to the thermal management of the servers.

- **AvgJRMin**: In this heuristic, a file's computation job-profile $\lambda_j(t)$ is approximated by a fixed value given by the file's average $\lambda_{avg,j}$ per-minute jobs arrival rate during the file's entire active life span $\mathcal{L}_{active,j}$.

- **AvgJRHr**: In this heuristic, a file's computation job-profile $\lambda_j(t)$ is approximated by the

file's average $\lambda_{avg,j}$ per-hour jobs arrival rate during the file's entire active life span $\mathcal{L}_{active,j}$.

- **AvgJRDay**: In this heuristic, a file's computation job-profile $\lambda_j(t)$ is approximated by the file's average $\lambda_{avg,j}$ per-day jobs arrival rate.

- **MedJRMin**: In this heuristic, a file's computation job-profile $\lambda_j(t)$ is approximated by the file's median $\lambda_{med,j}$ per-minute job arrival rate.

- **MedJRHr**: In this heuristic, a file's computation job-profile $\lambda_j(t)$ is approximated by the file's median $\lambda_{med,j}$ per-hour job arrival rate.

- **MedJRDay**: In this heuristic, a file's computation job-profile $\lambda_j(t)$ is approximated by the file's median $\lambda_{med,j}$ per-day jobs arrival rate.

- **MaxJRMin**: In this heuristic, a file's computation job-profile is approximated by the file's maximum $\lambda_{max,j}$ per-minute jobs arrival rate.

- **MaxJRHr**: In this heuristic, a file's computation job-profile is approximated by the file's maximum $\lambda_{max,j}$ per-hour jobs arrival rate.

- **MaxJRDay**: In this heuristic, a file's computation job-profile is approximated by the file's maximum $\lambda_{max,j}$ per-day jobs arrival rate.

- **JobsLS**: In this case, GreenHDFS calculates a file's approximate jobs arrival rate by dividing the predicted value of the total number of jobs $J_{tot,j}$ observed by a file in its lifetime by its predicted active life span $\mathcal{L}_{active,j}$ value. The calculated jobs arrival rate is then used in the algorithm 8 to guide GreenHDFS's thermal-aware file placement. This heuristic works well if the files have a uniform jobs arrival rate during their lifetime. However, if the files have a skew in the jobs arrival rates on certain day (e.g., more jobs soon after file creation), then this heuristic doesn't perform as well.

Choice of the statistical value of jobs arrival rate involves thermal-reliability vs. efficient server file allocation trade-offs and requires a study of the standard deviation in a file's jobs arrival rate. Approximating each file $\tilde{f_j}$'s jobs arrival rate $\lambda_j(t)$ with $\lambda_{max,j}$ ensures that even when the computational load on the files is at its highest, the temperature $T_{i,t}$ of a server i won't exceed beyond its reliability upper-bound T_{max}. Thus, choosing $\lambda_{max,j}$ to approximate files' jobs arrival rate is the most reliable approximation method for ensuring thermal-reliability. However, $\lambda_{max,j}$

over-approximates a file's jobs arrival rate, and can be constraining in file placement. If each server i were to be represented as a bin of capacity $L_{max,i}$, the server bin may get full sooner if each file was represented by the high value of $\lambda_{max,j}$.

If the standard deviation in the files' job arrival rate is low, using the average jobs arrival rate, $\lambda_{avg,j}$ gives a better thermal-reliability vs. efficient server file allocation trade-off. For example, in the evaluation data set, majority of the files have very low standard deviation in their observed jobs arrival rate at per-minute granularity. Hence, $\lambda_{avg,j}$ at a per-minute granularity turns out to be a very good approximation for the per-minute jobs arrival rate for the dataset. If the standard deviation of files' job arrival rates is high, then using $\lambda_{avg,j}$ as the approximation may again lead to thermal-reliability vs. efficient server file allocation trade-offs as $\lambda_{avg,j}$ is affected by the high value of the outliers. In such a scenario, using $\lambda_{med,j}$ works better for efficient file allocation to the server bins as it is less sensitive to the high outliers. However, the thermal-reliability may get compromised when the outliers do happen during the run-time. The efficient server file allocation consideration is less of an issue if the active life span $\mathcal{L}_{active,j}$ of the files are small and more of an issue if the files have long active life spans.

5.3.1.2 Variable Rate

In this case, GreenHDFS roughly fits $\lambda_j(t)$ to a linear curve $\lambda_j(t) = m \cdot t + v$, where *v* is the number of jobs received by the file on the first day of its creation, denoted as $J_{f,j}$ and $m = J_{f,j}/\mathcal{L}_{active,j}$ in units of jobs/days. The job arrival rate of a file on a particular day is calculated as the value of $\lambda_j(t)$ corresponding to that day. The daily change in the job arrival rate is reflected in the cumulative predicted load $L^p_{i,t}$ value, maintained by GreenHDFS, for every server that is hosting chunks of the file $\tilde{f}_j$. Decaying the predicted job rates is much more realistic than assuming a constant job arrival rate every day, especially in the case of news-server-like job pattern which is typical in the log-processing workloads, one of the biggest use cases of Big Data analytics. This approximation works for almost 70% of the files in the real-world evaluation data set as they receive 90% or more of their share of jobs in the first day of their existence.

5.3.2 Thermal-Aware File Placement Heuristics

GreenHDFS solves the second subproblem in Section 5.2 by using thermal-aware file placement heuristics. GreenHDFS thermal-aware file placement heuristics are classified in three ways: *predictive*, *non-predictive*, and *hybrid*. The predictive heuristics rely on predicted partial information about the file's anticipated computational job profile. Non-predictive heuristics assume no future knowledge about the incoming files and only use run-time information about the servers in the cluster to guide file placement. Hybrid heuristics use a combination of predicted and run-time information about the servers in the cluster in their thermal-aware file placement decision making process. In the next sections, we detail the various thermal-aware file placement heuristics in place in GreenHDFS.

5.3.2.1 Predictive Heuristics

At the time of a new file $\tilde{f}_j$'s creation, T^* uses the predictive models to predict $\mathcal{L}_{active,j}$, and $\lambda_j(t)$, to provide as input to the thermal-aware file placement illustrated in Algorithm 8 which places the file chunks and their replicas on the cluster in a thermal-aware manner. T^* aims to place the first and second replica of a file chunk f_j^k on different servers on the same rack and the third replica on a different rack for fault-tolerance similar to Hadoop or Google clusters [61, 84].

For every file chunk f_j^k, as illustrated in Algorithm 8, for the placement of the first replica, T^* orders the servers in the cluster in an ascending order of $L_{i,t}^e = L_{i,t}^p + \lambda_j(t) \cdot l$, where $L_{i,t}^e$ is the expected load on server i in the interval t if file's chunk f_j^k were to be placed on it, and $L_{i,t}^p$ is the predicted cumulative load on the server in time interval t as shown in Equation 5.6. Please note that $L_{i,t}^p$ is different from the actual run-time load on the servers, and is T^*'s approximation of the predicted future load on servers. T^* updates $L_{i,t}^p$ value for each server in the cluster whenever a new file arrives or a file becomes dormant. T^* then further orders the servers in ascending order of their expected temperatures $T_{i,t}^e = T_{i,t-1} + T_{j,t}^e$; where, $T_{j,t}^e$ is the estimated temperature rise caused by the placement of the chunk on the server and $T_{i,t-1}$ is the run-time temperature of the server in the last monitoring interval. It is important to use the temperature as the second-level server ordering to reduce the sensitivity of the heuristic to system variables.

T^* then picks one of the top-most server from the ordered list of servers whose $L_{i,t}^e < L_{max,i}$, $T_{i,t}^e < T_{max}$, and that has enough free capacity $\hat{\mathcal{D}}_{j,t-1}$ available to host the incoming file chunk

$c^k_{j,i}$. $L_{max,i}$ is the maximum load that can be tolerated by a server i without exceeding the reliability upper-bound of the temperature T_{max} as given by Equation 5.7. The $L_{max,i}$ and T_{max} checks in step 10 of Algorithm 8 are important to ensure thermal-reliability. The first replica of the chunk is placed on the server and the server's predicted cumulative load $L^p_{i,t}$ is updated by adding the predicted load of $c^k_{j,i}$, which is given by $\lambda_j(t) \cdot l$ and its free capacity is decremented by the size of the file chunk. For the placement of the second replica of chunk f^k_j, T^* restricts the selection scope of the suitable server to the rack hosting the first replica and repeats steps 3 till 20 in Algorithm 8. For the third replica placement, T^* excludes the rack hosting the first and the second replica from its search scope. The rest of the server selection steps remain the same.

Every server chosen in Algorithm 8, keeps track of the predicted active life span $\mathcal{L}_{active,j}$ of the file $\tilde{f}_j$. At the end of $\mathcal{L}_{active,j}$, the predicted load of the chunk $c^k_{j,i}$ given by $\lambda_j(t) \cdot l$ is decremented from value of the hosting server's predicted cumulative load $L^p_{i,t}$ maintained by T^*. Subsequently, when a job arrives to a file $\tilde{f}_j$, for each file chunk, T^* finds the lowest temperature server hosting a replica of the chunk for the placement of the sub-job meant for that chunk as shown in Algothirm 9.

Algorithm 8 Thermal-Aware File Placement

Input: $\mathcal{L}_{active,j}$, $\lambda_j(t)$, $\hat{n}_j$, l, s_j

{For all the entire life time TP of the cluster}
{At every new file $\tilde{f}_j$ creation time}
{For each file chunk f_j^k}

1: **for** $k = 1$ to $\hat{n}_j$ **do**
{For the placement of the first replica of chunk f_j^k}
{For each server i in the cluster}
2: $\ddot{n} = |S|$
3: **for** $i = 1$ to $\ddot{n}$ **do**
4: $L_{i,t}^e \leftarrow L_{i,t}^p + \lambda_j(t) \cdot l$
5: $T_{i,t}^e \leftarrow T_{i,t-1} + T_{j,t}^p$
6: **end for**
7: Sort servers in ascending order of expected load $L_{i,t}^e$
8: Further sort servers in ascending order of expected temperature $T_{i,t}^e$
9: **for** $i = 1$ to $\ddot{n}$ **do**
10: **if** $L_{i,t}^e < L_{max,i}$ **then**
11: **if** $T_{i,t}^e < T_{max}$ **then**
12: **if** $\hat{\mathcal{D}}_{i,t} > s_j/\hat{n}_j$ **then**
13: Place chunk f_j^k's first replica on server i
14: $L_{i,t}^p \leftarrow L_{i,t}^p + \lambda_j(t) \cdot l$
15: $\hat{\mathcal{D}}_{i,t} \leftarrow \hat{\mathcal{D}}_{i,t-1} - s_j/\hat{n}_j$
16: break
17: **end if**
18: **end if**
19: **end if**
20: **end for**
{For the placement of the second replica of chunk f_j^k}
{For each server i in the same rack as server hosting first replica}
21: Repeat steps 3 till 20 with a different value of $\ddot{n} = |R| - 1$
{For the placement of the third replica of chunk f_j^k}
{For each server i in racks different from rack hosting first and second replicas}
22: Repeat steps 3 till 20 with a different value of $\ddot{n} = |S| - |R|$
23: **end for**

Algorithm 9 Thermal-Aware Sub-Job Placement

{At arrival of a job targeted to file $\tilde{f_j}$}
{For each file chunk f_j^k}
1: **for** $k = 1$ to $\hat{n}_j$ **do**
  {Find all servers hosting a replica of chunk f_j^k}
2: **for** $r = 1$ to 3 **do**
3: Lookup servers containing $c_{j,i}^k$ in its chunk set $C_{j,i}$
4: **end for**
5: Sort servers hosting chunk f_j^k's replicas in ascending order of their run-time temperature $T_{i,t-1}$ observed in the time interval $t-1$.
6: Select server with lowest $T_{i,t-1}$ to host sub-job for chunk $c_{j,i}^k$
7: Send sub-job to the selected server
8: **end for**

5.3.2.2 Hybrid Heuristics

In the next set of predictive thermal-aware file placement heuristics, GreenHDFS uses predictions in conjunction with run-time knowledge about the cluster in form of the thermal-map and job-count map. The job-count map $\{J_{n,t}^1, J_{n,t}^2, \cdots, J_{n,t}^N\}$ is collected by the cyber-monitor and comprises of the number of the jobs that were targeted to each server in the last monitoring time interval. The thermal-map is collected by the GreenHDFS's monitor component detailed in Section 4.6 comprises of the server outlet temperatures $\{T_1^{out}, T_2^{out}, \cdots, T_N^{out}\}$ of all the servers in the cluster. At the time of file creation, in the Algorithm 8, the servers are ordered both by the run-time information and predicted information.

- **Pred_Temp_JobCount** The servers are ordered first by their value $L_{i,t}^e$ as shown in the Algorithm 8 and are then additionally ordered by the server temperatures present in the thermal-map and the job counts present in the job-count map.

- **Pred_Temp** The servers are ordered first by their value $L_{i,t}^e$ shown in the Algorithm 8 and are then additionally ordered by the server temperatures present in the thermal-map.

- **Pred_JobCount_Temp** The servers are ordered first by their $L_{i,t}^e$ as shown in the Algorithm 8 and are then additionally ordered by the job counts present in the job-count map and

server temperatures present in the thermal-map.

- **Temp_Pred** In this heuristic, the servers are first ordered by the temperatures presented by the thermal-map and then, by the value $L^e_{i,t}$ as shown in the algorithm 8.

- **Pred_JobCount** The servers are ordered first by the value of $L^e_{i,t}$ as shown in the Algorithm 8 and are then additionally ordered by the server job counts present in the job-count map.

These hybrid schemes are semi-sensitive to the sample rate of the monitoring service which collects the run-time information from the physical system. When the run-time information is used as the second-level of ordering, the sensitivity to the sample rate decreases significantly and the benefits of the hybrid schemes increases.

5.3.2.3 Non-Predictive Heuristics

The non-predictive thermal-aware file placement heuristics don't use predictions in making the file placement decision. Instead they rely on the following run-time information about the servers in the cluster collected by GreenHDFS's monitors.

- Thermal-map which comprises of the exhaust temperatures $\{T^{out}_1, T^{out}_2, \cdots, T^{out}_N\}$ of all the servers in the cluster.

- Thermal-map which comprises of the inlet temperatures $\{T^{in}_1, T^{in}_2, \cdots, T^{in}_N\}$ of all the servers in the cluster.

- Free storage capacity map which includes the details on the free space available on each server in the cluster

There are two flavors of the non-predictive heuristics:

- **NonPredInlet** We use $T^{in}_{i,t}$ to get a total order on the servers at the time of file placement and select the n servers with the lowest inlet temperatures $T^{in}_{i,t}$ to store the chunks and replicas of an incoming file $\tilde{f_j}$.

- **NonPredOutlet** GreenHDFS also employs a naive non-predictive, greedy heuristic whereby it orders the servers in ascending order of their run-time outlet temperature $T_{i,t}$. GreenHDFS

selects n servers with the lowest temperatures $T_{i,t}$ to store the chunks and replicas of an incoming file $\tilde{f}_j$. The evaluation results illustrate why this naive heuristic doesn't work well.

- **NonPredRandom** This heuristic was implemented simply to provide a basis for the evaluation. In this case, GreenHDFS just places the incoming file's chunks and replicas randomly over the cluster without any thermal considerations, predictions, or run-time awareness.

Algorithm 10 Data-Placement-Agnostic Thermal-Aware Job Placement.

{At arrival of new job directed to file $\tilde{f}_j$}

Compute number of chunks $\hat{n}_j$ in file $\tilde{f}_j$

Find set $\mathcal{N}$ of $\hat{n}_j$ number of servers which have the lowest run-time temperature $T_{i,j}$, where $|\mathcal{N}| = \hat{n}_j$ and $\mathcal{N} \in S$

{For each chunk f_j^k of $\tilde{f}_j$, see if any of $\mathcal{N}$ servers are either server-local or rack-local to a server hosting a replica of the chunk f_j^k}

for $k = 1$ to $\hat{n}_j$ **do**

  Lookup set $\mathcal{R}$ servers hosting a replica $c^k_{replica,j,i}$ of f_j^k chunk where with $|\mathcal{R}|= 3$ is the replication factor of file $\tilde{f}_j$

  for $r = 1$ to $|\mathcal{R}|$ **do**

    if $r \in \mathcal{N}$ **then**

      {Server-Local Access}

      Send sub-job for file chunk f_j^k to server r

      $r.insertJobQueue(c^k_{replica,j,i})$

      Break, process next chunk

    end if

  end for

  {If none of the servers in $\mathcal{N}$ are server-local to any of the file chunks, see if the servers in $\mathcal{N}$ are atleast rack-local to any of the file chunks.}

  $\{\mathcal{NR}\} = \{\mathfrak{R}_{\mathcal{N}_1}, \mathfrak{R}_{\mathcal{N}_2}, \cdots, \mathfrak{R}_{\mathcal{N}_{|\mathcal{NR}|}}\}$

  for $r = 1$ to $|\mathcal{R}|$ **do**

    {Check if r's rack is same as the racks of any of the servers $\mathcal{N}$}

    if $\mathfrak{R}_r \in \mathcal{NR}$ **then**

      {Rack-Local Access}

      Send sub-job for file chunk f_j^k to server r

      $r.insertJobQueue(f_j^k)$

      Break, process next chunk

    end if

  end for

  {$\mathcal{N}$ servers are neither server-local nor rack-local to any of the servers hosting a replica $c^k_{replica,j,i}$ of chunk f_j^k. Need to perform non-rack-local data access}

end for

5.3.3 Data-Placement-Agnostic Thermal-Aware Job Placement

In this section, we discuss the implementation of data-agnostic, thermal-aware job placement technique which is based on the state-of-the-art cooling reduction techniques and thermal-aware replica selection technique to help us evaluate GreenHDFS's efficacy compared to state-of-the-art approaches. In both these implementations, unlike the thermal-aware placement of GreenHDFS, the files are placed in the cluster in an energy- and thermal-agnostic manner as done in baseline HDFS (v20). The energy- and thermal-awareness is exhibited only at the time of job arrival in these techniques.

5.3.3.1 Thermal-aware Job Placement

In this heuristic, hence referred to as **StateOfArt_Job**, at a analytic job's arrival targeted to a file $\tilde{f_j}$ with $\hat{n}_j$ chunks, $\hat{n}_j$ number of $\mathcal{N}$ servers with the lowest run-time temperatures $T_{i,t}$ are selected for the placement of the sub-jobs as shown in the Algorithm 10.

First, for every chunk f_j^k in $\tilde{f_j}$, an attempt is made to see if any of the $\mathcal{N}$ servers host either this chunk or its replica locally. If the file chunk or its replica is found on any of the $\mathcal{N}$ servers, the sub-job meant for the chunk is added to the server's job queue to take advantage of high server-local performance.

If none of the $\mathcal{N}$ servers contain the chunk or its replica, an attempt is made to see if any of these servers are rack-local to the servers hosting the chunk or its replicas. If a rack-local server is found, the rack-local server in $\mathcal{N}$ is allocated the sub-job targeted to the chunk.

If the $\mathcal{N}$ servers are neither rack-local nor server-local to any of the servers hosting the chunk or its replicas, the sub-job for the chunk is placed on one of the $\mathcal{N}$ servers and the data is accessed using inter-rack bandwidth from the closest server hosting a replica of the chunk. In this case, the overall job completion time is impacted by the latency of the non-rack-local access of this one block. In addition to the performance impact of non-rack-local data access, there is also an additional power cost. The $\mathcal{N}$ servers consume energy in computation which is a combination of the CPU and DRAM energy consumption. In addition, there is a network related power consumption for reading the data non-locally over the network from another server. The server hosting the data also incurs power consumption in network outbound traffic, disk accesses,

as well as in some DRAM and CPU utilization.

5.3.3.2 Thermal-Aware Replica Selection

In this heuristic, hence referred to as **StateOfArt_Replica**, at a job's arrival targeted to a file with n chunks, for each chunk in the file, servers with the least temperature T^i_{out} amongst the servers hosting a replica of that chunk are chosen for running the sub-job relevant to the chunk. Since, these servers have the chunks needed for the computation, the data access is server-local and hence, the performance is high. However, the thermal-aware choice of the servers per chunk is limited to the replication factor in the cluster which is typically 3 in the production clusters. It is possible that none of the three servers hosting the chunk's replicas have a low value of run-time temperature T^i_{out}. Some of them may even have a very high run-time temperature.

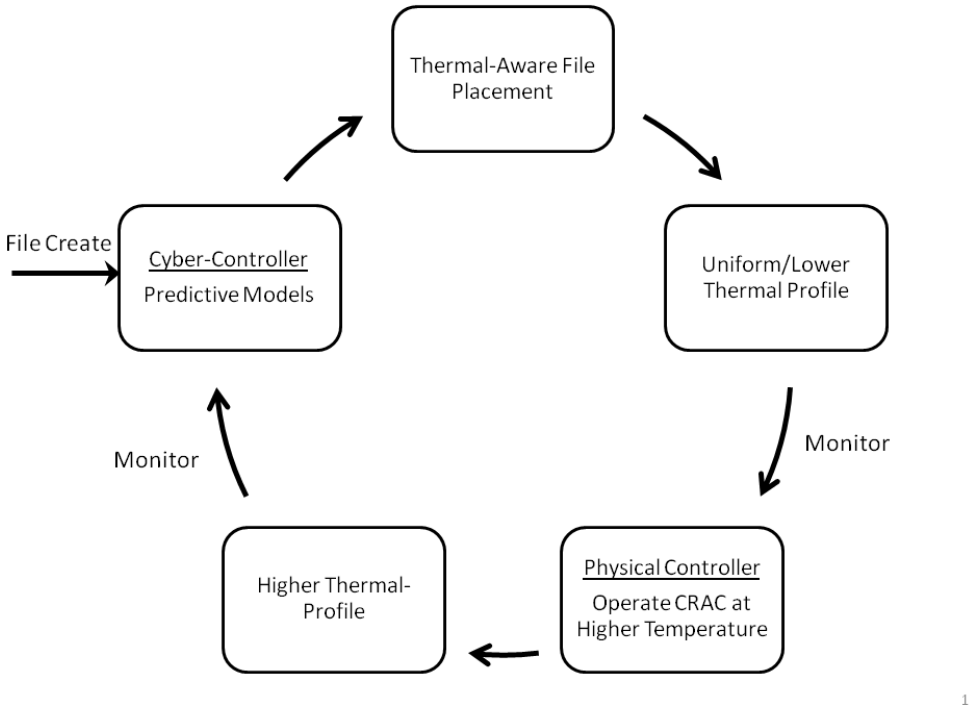

Figure 5.7: **The Cyber-Physical Loop in GreenHDFS.**

5.4 GreenHDFS Thermal-Aware Data Placement Evaluation

In this section, we first present our experimental platform and methodology, followed by a description of the workloads used and our experimental results. Our goal is to answer the following set of high-level questions:

- How does GreenHDFS's predictive thermal-aware file placement compare with state-of-the-art data-agnostic thermal-aware job placement?
- How does predictive thermal-aware file placement compare with non-predictive thermal-aware file placement and random, non-thermal-aware file placement?
- How sensitive are the file placement schemes to variables such as sampling rate of the monitoring service, file creation rate, job duration, delta in cooling-efficiencies of the servers, or file size skew?

We compare the above techniques on five important dimensions: 1) thermal-reliability, 2) cooling energy costs reduction, 3) load-balancing, 4) storage capacity-balancing, and 5) performance.

5.4.1 Traces

We evaluate with one-month long real-world Big Data analytics file system traces generated by a production (2600 servers, 34 million files, 5 petabytes) cluster at Yahoo!. We focus our analysis on the biggest (60% of the used storage capacity) and most important data set (clickstream) in the production cluster. Internet services companies such as Facebook, Google, and Yahoo! [34] rely on clickstream processing [52], an example of log processing (one of the most important use cases of Big Data analytics), to calculate the web-based advertising revenues, and derive user interest models and predictions. For this, daily huge web logs are analyzed in the production environments [135]. The files in the data set have significant skew in sizes, job arrival rates, and evolution life spans. Out of the 34 million files hosted in the cluster, 39% of files are smaller than 1K (similar presence of small files has been noted in other Big Data analytics clusters), 73% of files had an active life span $\mathcal{L}_{active,j}$ of one day, and the statistical job arrival rates vary across files with some files receiving many more jobs than the others.

5.4.2 Platforms

5.4.2.1 floVENT

To simulate a Big Data analytics cluster we use Mentor Graphic's floVENT, a computational fluid dynamics (CFD) simulator [6]. floVENT has been extensively used and validated in several research papers in the past [29, 107, 126]. The cluster under evaluation has 4 rows of 14 industry standard 47U racks arranged in cold- and hot-aisle layout [132]. Each rack contains 46, 1U servers for a total of 2576 servers. The cold air is supplied by CRACs with supply temperature T_{ac} fixed at 15^{o}C. To determine the inlet temperatures of all the servers in the cluster, a floVENT simulation with all the servers in the cluster at idle utilization is performed. The inlet temperatures observed in the floVENT simulation are then used by the GreenHDFS simulator. The power consumption of each server in the cluster in the floVENT simulation of GreenHDFS is directed by the GreenHDFS simulator detailed next.

5.4.2.2 GreenHDFS

The simulator assumes same number and layout of servers as the floVENT simulation. At the time of file creation, the files are divided into 64 MB chunks and are replicated three-way. Using a simulator is critical in understanding the affect of various combination and order of parameters on the heuristics. The simulator implements all predictive and non-predictive thermal-aware file placement heuristics covered in Section 5.3.2.2.

5.4.3 GreenHDFS vs. State-of-the-Art

In this section, we compare cooling costs, performance, and thermal-profile of the cluster with GreenHDFS's predictive thermal-aware file placement vs. state-of-the-art, data-placement-agnostic thermal-aware job placement.

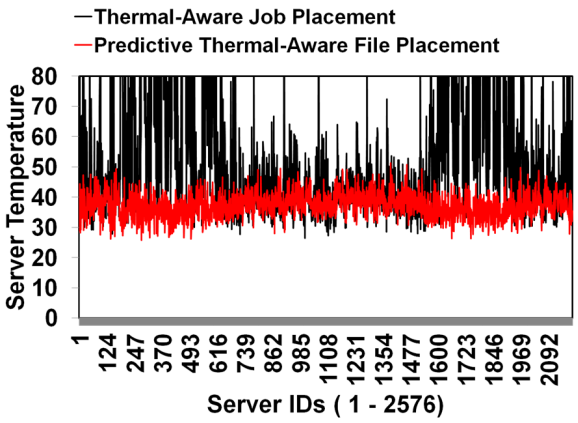

Figure 5.8: **Maximum temperature observed by servers in the cluster.**

5.4.3.1 Thermal-Profile

Maximum temperatures experienced by the servers throw light on the thermal-reliability efficacy of an energy management technique. Lower the maximum temperatures observed, higher is the thermal-reliability assurance of the technique. Figure 5.8 shows the maximum temperature experienced by each of the 2576 servers in the cluster during the one-month simulation run. The monitoring service's sampling rate is assumed to fine-grained which results in lower thermal-profile for the thermal-aware job placement technique. Still, maximum temperatures observed by the servers are much lower with GreenHDFS's thermal-aware file placement than with thermal-aware job placement. Figure 5.12 compares the temperature snapshot of the servers in the cluster at the same time instant during the simulation run with fine-grained and coarse-grained sampling rates for the two schemes. Predictive thermal-aware file placement in GreenHDFS results in lower and more uniform temperature than thermal-aware job placement. Predictive thermal-aware file placement is not sensitive to the sampling rate of the monitoring service and fares almost the same with fine- and coarse-grained sampling, whereas thermal-aware job placement fares much better with fine-grained sampling rate. The thermal-profile of data-agnostic thermal-aware job placement technique is also sensitive to the delta in the cooling-efficiencies of the servers in the cluster, and the job durations. On the other hand, the predictive thermal-aware file placement is minimally sensitive to the job durations, or delta in cooling-efficiencies.

Predictive thermal-aware file placement results in a reduction in the temperature $T_{c,t}^{out}$ of the

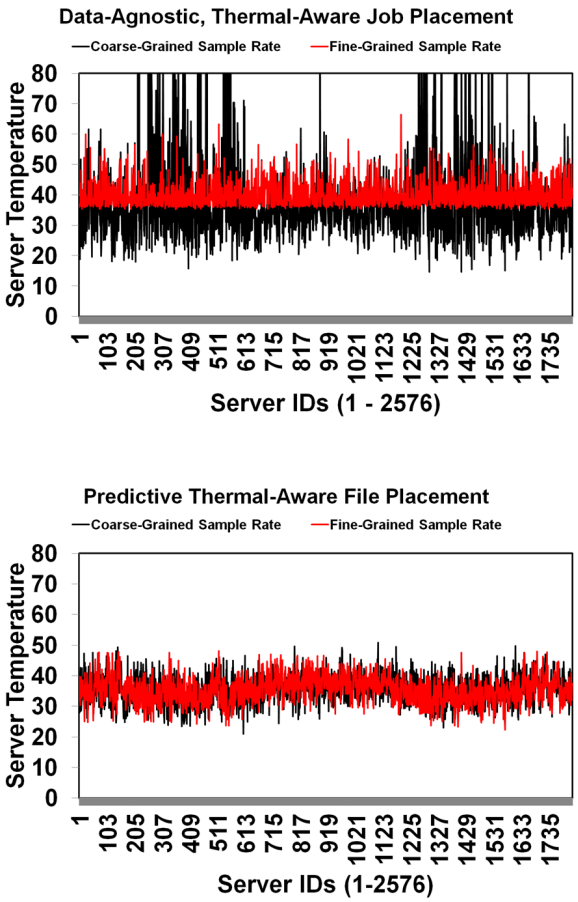

Figure 5.9: **Temperature snapshot per server with different sampling rate of monitoring service.**

exhausted air in the cluster, and hence air conditioner (CRAC) can be operated at a higher set point temperature T_{ac}. Increase in T_{ac} increases the efficiency (COP) of the CRAC allowing it to remove more heat with less work and thus, reduces cooling energy costs. For example, operating the CRACs at 5^{o}C higher supply temperature of 20^{o}C increases COP from 5 to 6, resulting in additional cooling costs savings.

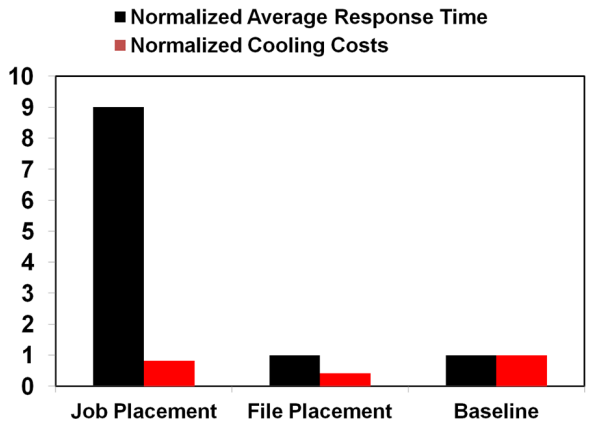

Figure 5.10: **Normalized performance and cooling energy costs.**

5.4.3.2 Cooling Energy Costs

Figure 5.11, shows the normalized cooling energy costs during the one-month simulation run. Since, the thermal-profile is sensitive to various system parameters in case of thermal-aware job placement, we compare with the best-case cooling energy costs savings possible with thermal-aware job placement. The predictive thermal-aware file placement saves 42% cooling energy costs vs. 18% in case of thermal-aware job placement.

5.4.3.3 Performance

As shown in Figure 5.10, GreenHDFS has 9x times lower average response time than the thermal-aware job placement. In case of data-placement-agnostic thermal-aware job placement technique, the sub-jobs of the incoming job are placed on the servers with the lowest run-time temperatures. These servers may not all have the data relevant to the computation resident on them, requiring high latency rack-local or inter-rack data access, which results in degraded performance. The overall response time of a job is equivalent to the maximum completion time of its sub-jobs; even one straggler sub-job can increase the response time. In case of data-placement-agnostic thermal-aware job placement technique, presence of a large number of stragglers results in much

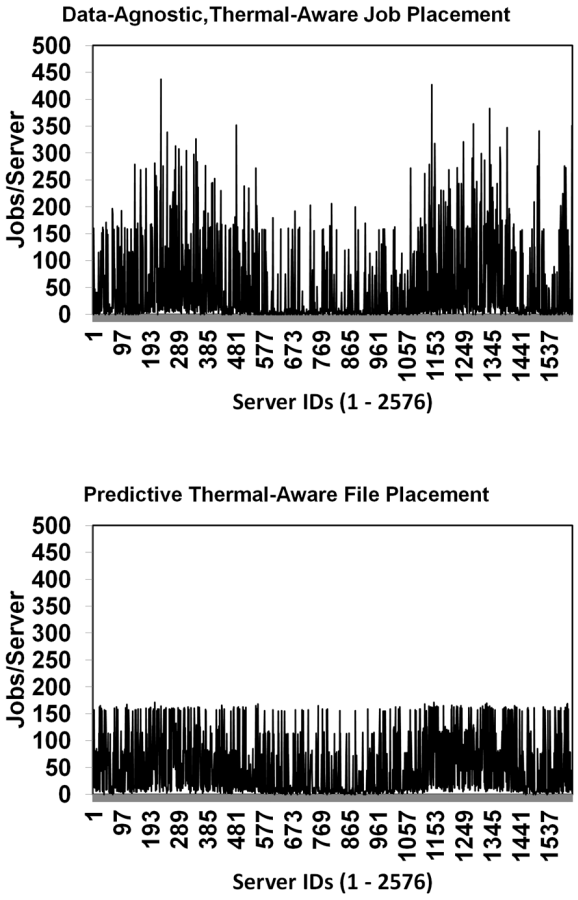

Figure 5.11: **Snapshot of jobs received per server at a time instant.**

lower overall performance. On the other hand, GreenHDFS is fully data-placement-aware and doesn't suffer performance impact.

Next, we look at load-balancing of the jobs under the two schemes. As shown in Figure 5.11, jobs are uniformly distributed in case of GreenHDFS, whereas, jobs are more skewed towards some servers in case of thermal-aware job placement. Thus, GreenHDFS achieves better job load-balancing than state-of-the-art thermal-aware job placement technique, and fares better in yet another aspect of performance. At higher loads, GreenHDFS does show a skew towards the cooling-efficient servers as they have higher load tolerance $L_{max,i}$ values.

5.4.4 Predictive Thermal-Aware File Placement Heuristics

In constant rate heuristics, using statistical value of maximum job arrival rate per minute $\lambda_{max,j}$ as an approximation results in the lowest values of the maximum temperature ever observed by a server during the 30-day simulation run. However, using $\lambda_{max,j}$ is an over-approximation and results in server bins getting artificially over-packed, which leads to an occasional inability to find servers for the incoming file at high cluster utilization. Overall, approximation of file's job arrival rate using average job arrival rate per minute $\lambda_{avg,j}$ works best. It allows cooling costs savings, thermal-reliability, efficient server bin packing, load-balancing, and storage capacity balancing and is not sensitive to job durations, sampling rate of the monitoring service, file size skew, and file creation rate. Statistical analysis of the data set further confirms the results as the standard deviation in the job arrival rates at a minute granularity is low for majority of the files. Please note that $\lambda_{avg,j}$ is not a true average and is calculated only for the minutes with non-zero jobs. Variable rate heuristic is advantageous when files are active for longer periods of time. Majority of the files in the data set have a short active lifespan $\mathcal{L}_{active,j}$ of less than a day, thereby limiting the advantage of variable rate heuristic.

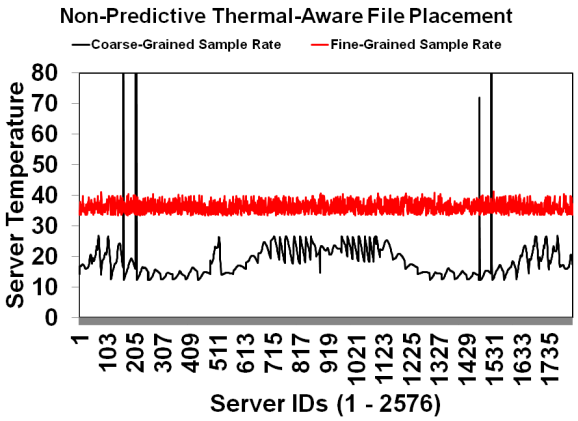

Figure 5.12: **Temperature snapshot per server with different sampling rate of monitoring service.**

5.4.5 Predictive vs. Non-Predictive Heuristics

The non-predictive heuristics are very sensitive to job durations, sampling rate of the monitoring service, file size skew, file creation arrival rate and inter-arrival time, and the delta in the cooling-efficiencies of the servers. If the job durations are short, the sampling rate is very fine-grained, and delta in servers' cooling-efficiencies is high, non-predictive thermal-aware file placement results in a better (i.e., more uniform and lower thermal-profile) than even predictive thermal-aware file placement as shown in Figure 5.12.

Non-predictive scheme greedily fills servers with lower run-time temperatures with the incoming files. If the job durations are very short and the sampling rate is coarse-grained, the last interval reading of the temperature sensors by the monitoring service is not an accurate representation of the actual run-time temperatures. If the delta in the cooling-efficiencies of the servers is high, the most cooling-efficient servers tend to be lowest temperature servers in the last monitoring interval. Given, the significant skew in the file sizes in the evaluation dataset, a lot of new files get placed on the most cooling-efficient servers, resulting in a load- and storage-unbalance. Unfortunately, such a greedy placement can increase the temperature of these servers to very high values when computational jobs are directed to the files residing on these servers resulting in a thermally-unreliable system. A very fine-grained sampling rate of the monitoring service helps make the technique much less sensitive to system variables. However, in Big Data analytics clusters such as Hadoop cluster, the master server is a single entity and gets overwhelmed processing heartbeats from 4000+ worker servers if heartbeats are sent at a very fine time-granularity. Thus, non-predictive techniques can not be used in an ad hoc manner and need careful understanding of the workload and dataset characteristics.

5.4.6 GreenHDFS vs. Random File Placement

In this section we compare the best-performing GreenHDFS predictive thermal-aware file placement scheme with random, non-thermal-aware file placement. Figures 5.13 and 5.14 show the thermal contour plots at different planes in the cluster with random, non-thermal-aware file placement and with predictive thermal-aware file placement in GreenHDFS respectively. The GreenHDFS thermal profiles are much more uniform and have lower temperatures than the random placement thermal plots. The maximum temperatures experienced by the servers with random

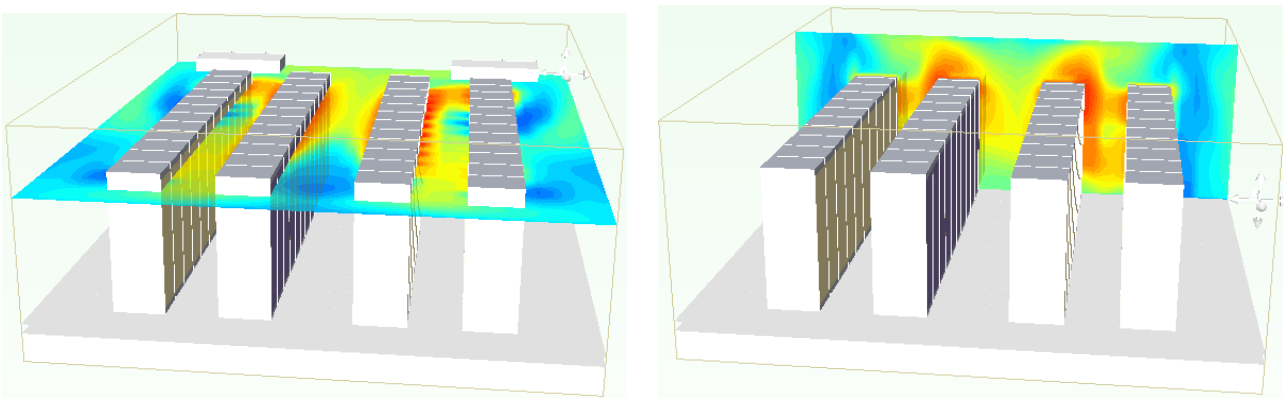

Figure 5.13: **Thermal contour plots of the servers and maximum temperatures observed in the cluster with random, non-thermal-aware file placement**

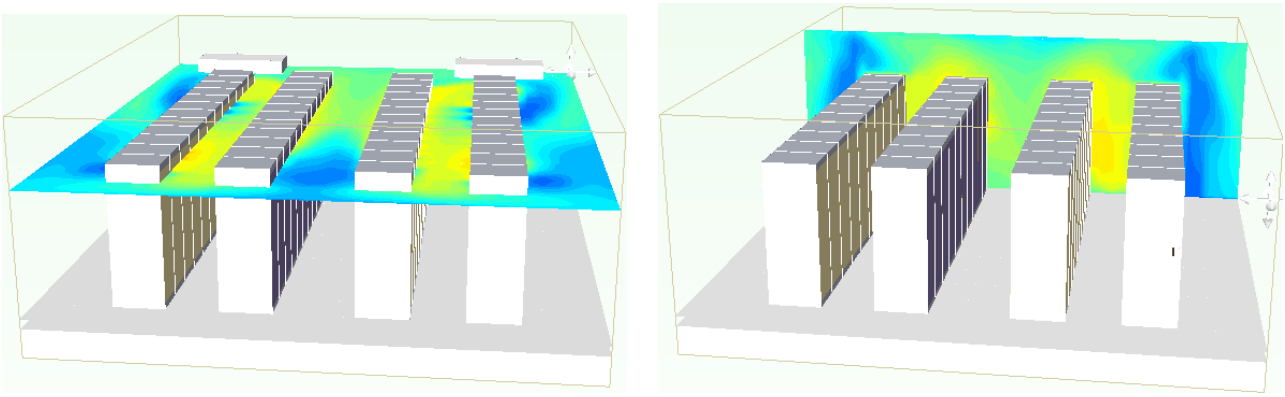

Figure 5.14: **Thermal contour plots of the servers in the cluster managed by GreenHDFS.**

file placement, as shown in Figure 5.13, are higher than GreenHDFS, leading to thermal-reliability concern under high load. Load-balancing of jobs is more uniform with predictive technique than with random placement. Thus, an informed, predictive placement is much better than a random placement for reducing cooling energy costs and for thermal-reliability assurance. Good features of random placement are that it is not sensitive to system variables such as sampling rate and observes data-locality just like the predictive technique.

5.4.7 Discussion

Free-cooling or air- and water-side economization (i.e., use outside air or natural water resources to cool the data center) is gaining popularity as it can result in significant cooling energy costs

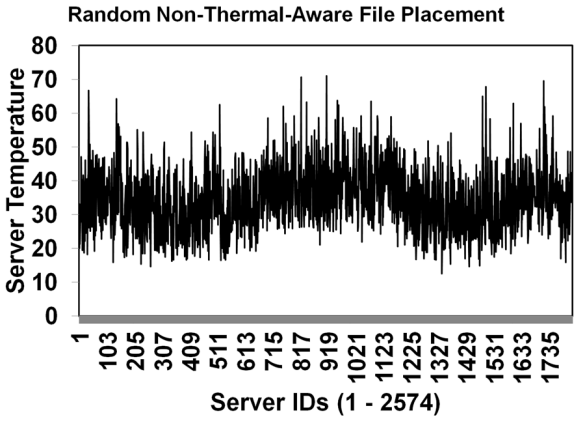

Figure 5.15: **Maximum temperatures observed in the cluster with random, non-thermal-aware file placement**

savings [110]. There is also a drive towards increasing the cooling set point of the cooling systems to make them more efficient [27]. If the ambient temperature of the outside air or the cooling set point temperature is high, the inlet temperatures of the servers get high which reduces their ability to dissipate computational heat, resulting in an increase in server temperatures. The servers are rated to operate safely only with a certain temperature range, beyond which the failure rates increase. GreenHDFS considers the differences in the thermal-reliability-driven load-tolerance upper-bound of the servers in its predictive thermal-aware file placement and places file chunks in a manner that ensures that temperatures of servers don't exceed T_{max}. Thus, by ensuring thermal-reliability at all times and by lowering the overall temperature of the servers, GreenHDFS enables data centers to enjoy energy-saving economizer mode for longer periods of time and also enables an increase in the cooling set point.

There are a substantial number of data centers that still rely fully on traditional air-conditioning. These data centers can not always be retrofitted with the economizer modes or hot- and cold-aisle air containment as incorporation of the economizer and air containment may require space for ductwork, and heat exchangers which may not be available in the data center. Existing data centers may also not be favorably located geographically; air-side economization is more viable in geographic locations where ambient air temperatures are low for most part of the year and humidity is in the tolerable range. GreenHDFS provides a software-based approach to enhance the cooling-efficiency of such traditional data centers as it lowers the overall temperature in the cluster, makes the thermal-profile much more uniform, and reduces hot air recirculation, resulting

in lowered cooling energy costs.

Table 5.1: **Table of Notations Used in the Thermal-Aware File Placement.**

Variable	Description	Units
t	Time interval	Seconds
i	A server in the cluster	
$T_{i,t}$	Temperature of server i at time t	Kelvin (K)
$T_{i,t}^{in}$	Server i's inlet temperature	Kelvin (K)
T_{out}^{i}	Server i's exhaust temperature	Kelvin (K)
π	File chunks to servers mapping function	
$P_{i,t}$	Power consumption of server i in t	Watts (W)
$L_{i,t}$	Computational Load of Server i in t	
T_{max}	Reliability driven upper-bound on the temperature that can be tolerated by a server as specified in the data sheet	Kelvin (K)
c_i	Factor of server i's heat exchange rate, it's air flow, and heat capacity density of the air	Kelvin.Second/Joule (Ks/J)
w_{i_1}	Power coefficient of server i	Joules/Second (J/s)
w_{i_2}	Power coefficient of server i	Joules/Second (J/s)
T_{ac}	Air supply set point temperature of the air conditioner	Kelvin (K)
$P_{c,t}$	Cooling power consumed in cluster at t	
c_{air}	Heat capacity constant	Joules/Kelvin.meter^3 (J/Km^3
f_{ac}	Flow rate of the cooling unit	Meter^3/Second (m^3/s)
$\|S\|$	Total number of the servers in the cluster	
$\|Z\|$	Total number of the files in the cluster	
$\|R\|$	Total number of servers in a rack	
$L_{max,i}$	Maximum load that can be subjected to a server i without exceeding T_{max}	
ρ_i	Average utilization of server i	
l	Average job duration	Seconds
P_{total}	Overall compute and cooling power	Watts (W)
$J_{n,t}^{i}$	Job count in the last monitoring interval at server i	
$\hat{\mathcal{D}}_i, t$	Free storage capacity in the last monitoring interval at server i	
θ_i	Heat exchange rate of server i	Joules/Kelvin.Second (J/Ks)

Table 5.2: **Table of Notations Used in the Thermal-Aware File Placement.**

Variable	Description	Units
$\tilde{f}_j$	A file in the cluster	
f_j^k	Chunk k of file $\tilde{f}_j$	
$\hat{n}_j$	Number of chunks in file $\tilde{f}_j$	
$\tilde{f}_j$	Set of all chunks of file $\tilde{f}_j$	
$c_{j,i}^k$	Chunk k of file $\tilde{f}_j$ assigned to server i	
$C_{j,i}$	Set of file $\tilde{f}_j$'s chunks assigned to server i	
$n_{j,i}$	Number of file $\tilde{f}_j$'s chunks assigned to server i	
$\tilde{C}_i$	Set of chunks of different files assigned to server i	
$\mathcal{L}_{active,j}$	File life span between the file creation and the last job received by the file $\tilde{f}_j$	Days
$J_{T,j}$	Total number of computational jobs targeted to a file $\tilde{f}_j$ over its active lifespan $\mathcal{L}_{active,j}$	
$\lambda_{avg,j}$	Average arrival rate of computational jobs targeted to a file $\tilde{f}_j$	
$\lambda_{med,j}$	Median arrival rate of computational jobs targeted to a file $\tilde{f}_j$	
$\lambda_{max,j}$	Maximum arrival rate of computational jobs targeted to a file $\tilde{f}_j$	

CHAPTER 6

CONCLUSION

State-of-the-art energy management techniques rely on thermal- and energy-aware job placement, migration, or consolidation to garner energy costs savings. These techniques are not applicable in the Big Data analytics environment as they are inherently data-placement-agnostic in nature; thereby, provide energy savings at significant performance impact in the Big Data environment. GreenHDFS is based on the observation that data needs to be a first-class object in energy-management in the Big Data environments to allow high performance, given the data-intensive nature of Big Data analytics. GreenHDFS takes a novel data-centric, cyber-physical approach to reduce cooling and compute energy costs. On the cyber-side, GreenHDFS is aware of the differences in the data-semantics (i.e., computation job profile, job rates, and life spans) of the Big Data placed in the compute cloud. On the physical-side, GreenHDFS is cognizant of the uneven thermal-profile in the data centers. Armed with this cyber-physical knowledge, and coupled with its insights, predictive data models, and run-time information GreenHDFS does proactive, cyber-physical, thermal- and energy-aware file placement and scale-down, which implicitly results in thermal- and energy-aware job placement in the Big Data Analytics cloud compute model. GreenHDFS aims to reduce server operating and cooling energy costs, enhance energy-proportionality, and ensure thermal-reliability of the servers in the Big Data analytics cluster without impacting performance. The energy management is done purely at the storage management layer, keeping the file system interfaces to the Big Data analytics applications the same. Thus, Big Data analytics applications can take advantage of energy costs savings without requiring any code rewrite. The next sections summarize the findings, lessons learnt, and the future work of GreenHDFS.

6.1 Data-Centric Compute Energy Management

GreenHDFS takes a *data-centric*, cyber-physical scale-down approach to reduce server operating energy costs. Instead of energy-aware consolidation of jobs or replicas as was done earlier, GreenHDFS focuses on energy-aware consolidation of files to realize sufficient idleness to enable scale-down. On the cyber-side, GreenHDFS is cognizant that all-files-are-not-alike on the cluster. For example, files in the cluster can be in different evolution phases: some files may be in their active phase; whereby, the files are actively being computed upon and accessed, and other files may be in their dormant phase; whereby, the files are past their active phase and are now just lying dormant in the system without receiving any computations or accesses for regulatory, compliance, disaster recovery, or .

On the physical-side, GreenHDFS is aware that even in a data center with free-cooling, economization, or air containment, uneven server inlet thermal-profile exists because of distance from and varying ability of the cooling system to cool different parts of the data center. In traditional air-cooled data centers, uneven thermal-profile is further aggravated because of hot air recirculation and air bypass. The higher the inlet temperature of a server, the lower is its cooling-efficiency (i.e., ability to remove the heat generated). Lower cooling-efficiency compromises the ability of a server to dissipate the heat generated by the computational load, resulting in higher server temperatures. Such thermal hot-spots are the main cause of high cooling energy costs as the cooling system needs to run at lower supply set point temperature to alleviate the thermal hot-spots. Running cooling system at lower temperature reduces its efficiency, further increasing the cooling energy costs.

Storage and compute models of Big Data analytics present several scale-down challenges. The underlying storage system distributes file chunks and replicas across the servers for higher performance, load-balancing and resiliency. Any server may be participating in the reading, writing, or computation of a file chunk at any time. Such a data placement makes it hard to generate significant periods of idleness in the Big Data analytics clusters and renders usage of inactive power modes infeasible [91]. GreenHDFS is able to achieve significant periods of idleness in the very same environment by recognizing that all-the-files-are-not-alike, and that files are in varying stages of their evolution lifespans in the cluster. Some files are in their active lifespan, i.e., are being actively computed upon and accessed, while other files are past their active lifespan and are now simply lying dormant in the system without receiving any computational jobs or accesses. Our study of a production Big Data analytics Hadoop cluster at Yahoo! [79] found that 56%

of the storage in the cluster was in a dormant state (i.e., was not accessed at all in the entire one-month long analysis duration). A majority of the dormant data needs to exist for regulatory, compliance, and historical trend analysis purposes and cannot just be deleted [79]. An IDC study found that almost 70% of the data in the data centers is actually dormant in nature.

Based on its cyber-physical awareness, GreenHDFS separates cluster servers and files into logical $Active$ and $Inactive$ zones in a cost-, performance-, temperature-, and power-differentiated manner. $Active$ zone servers are used to host new, incoming files and files that are actively being computed upon; on the other hand, GreenHDFS identifies, separates, and consolidates dormant files, i.e., files that are past their active lifespan and now, have either none or very infrequent computations targeted to them, on the $Inactive$ zone servers. Since computations follow data to capitalize on high server-local data access performance, a majority of the computations get directed to $Active$ zone, allowing significant idleness in the $Inactive$ zone servers, which can then be scaled-down.

GreenHDFS assigns the most inherently cooling-inefficient servers in the cluster to the $Inactive$ zone. The dormant files are best suited for the cooling-inefficient servers as these servers have low load tolerance $L_{max,i}$ values as discussed in Section 5.1.6. $L_{max,i}$ is the maximum load that can be tolerated by a server without exceeding the reliability-driven temperature upper-bound T_{max} defined in the server's data sheet. $L_{max,i}$ is inversely proportional to the inlet temperature of a server, and since cooling-inefficient servers have high inlet temperatures, they have lower load tolerance. By placing dormant files on these servers, GreenHDFS ensures that the cooling-inefficient servers receive negligible computations, don't generate much heat, and as a result, their exhaust temperature remain bounded. Such a thermal-aware data zone partitioning reduces the thermal hot-spots in the cluster leading to overall even and lower server temperatures in the cluster which in turn reduces the cooling energy costs in addition to reducing server energy costs.

Big Data analytics is data-intensive in nature and hinges on high data access performance. Hence, cluster zone partitioning in GreenHDFS is done in an energy- and performance-aware manner. GreenHDFS uses zone partitioning optimization algorithm detailed in Section 4.5.1 to determine the optimal number of servers to be assigned to the $Inactive$ zone subject to performance (i.e., throughput and response time) and capacity constraints. Evaluation results with real-world Big Data analytics data set show that 18%-33% of the servers in the 2600 server evaluation cluster can be allocated to the $Inactive$ zone without any transfer bandwidth or average response time degradation. Once the number of servers to assign to $Inactive$ zone is decided, GreenHDFS does

either cluster-level or rack-level thermal-aware zone partitioning of the servers between the two zones. Cluster- and rack-level schemes make different energy savings and performance trade-offs as discussed in Section 4.5.2.

GreenHDFS takes benefit of zoned bit recording (ZBR) commonplace in today's hard disks to place active files on the outer hard disk cylinder zones which have higher transfer bandwidth than the inner hard disk cylinder zones. Such a placement reduces the service time of the data accesses, and hence, alleviates the load-unbalancing caused by scale-down. We did a feasibility study of such high performance file layout with the clickstream dataset and realized that $Active$ zone server disks get only 26% full with the active and new incoming files. The low storage capacity requirements and short active life spans of the active files noticed in the clickstream dataset in the Big Data analytics cluster at Yahoo! makes high performance layout of the active files on the outer disk cylinder zones feasible. Since, clickstream processing is the most important use-case of Big Data analytics and is prevalent across Internet services companies, the feasibility study can be extrapolated to work across companies.

GreenHDFS's data-centric scale-down approach hinges on accurate determination of dormancy of the files as both premature and late classification of the dormancy of the files have different repercussions. A *File Migration Policy* running in GreenHDFS performs file dormancy determination in two ways: reactive and predictive. Reactive GreenHDFS covered in Section 4.6.7.1 relies on insights gleaned reactively from the observed access patterns of the files to drive its energy-aware file policies and file dormancy determination. Predictive GreenHDFS covered in Section 4.6.7.2 uses predictive models which it builds from supervised machine learning of historical traces to predict file attributes to guide file placement and determine file's dormancy in a proactive and finely-tuned manner instead of relying on one-size-fits-all file policies. The *File Migration Policy* not only aids in scale-down, but also results in much more available storage capacity in the $Active$ zone which can be used for better active file layout, and more aggressive replication for obviating performance hot-spots.

There are several advantages of using a Predictive variant of GreenHDFS as opposed to using a Reactive variant of GreenHDFS:

- Higher accuracy in identifying truly dormant files. As a result, files are moved to the $Inactive$ zone only when they have truly become dormant and are not just observing a lull in their jobs arrival rate. Accurate dormancy determination significantly reduces accesses to the files residing on the $Inactive$ zone which in turn results in:

- Higher amount of idleness in the $Inactive$ zone leading to longer periods of scale-down resulting in higher energy savings.

- Reduced performance impact courtesy of reduced accesses to the $Inactive$ zone. Accesses to the $Inactive$ zone files in GreenHDFS may suffer performance degradation because of two reasons: 1) they may require an inactive-to-active power state transition of the scaled-down server resulting in a transition penalty as high as ten seconds, and 2) files are laid out in a performance-sacrificing, high energy saving manner in the $Inactive$ zone.

- Timely reclamation of the storage space in the $Active$ zone servers given the fine-grained migration of dormant files, which is very important for the feasibility of high performance active file placement on the outermost cylinder zones of the disks covered in Section 4.5.1

- Higher scalability as instead of enumerating the entire file name space for identifying dormant files, GreenHDFS now just needs to look at its data structures for the list of migration file candidates on the day of the policy run. The candidate migration files on any day are a very small fraction of the total name space.

- Lower runtime overhead as there is no need to record the last access time of the files.

GreenHDFS predictive models, as detailed in Section 4.6.1, are meant for the clickstream processing, an example of log processing, which is one of the most important use cases of Big Data analytics and has significant business value. Internet services companies such as Facebook, Google, Twitter, LinkedIn, and Yahoo! [34] rely on clickstream processing [34, 52, 133] of huge web logs daily to calculate the web-based advertising revenues (source of a majority of their income), and to derive user interest models and predictions.

GreenHDFS is not tied to a particular predictive modeling technique as different workloads and datasets has different feature sets, and hence, need to be trained differently in order to generate predictive models. In the case of clickstream processing, there is correlation between the hierarchical file path of a file and its attributes, hence using subdirectories in the file path as the feature set works well. Other datasets may not have any correlation between the file path and instead may display correlation between file's meta data such as file's owner, and file's attributes. The clickstream workload falls in the category of predictable workloads and hence, quite accurate

predictive models can be generated for such a workload. On the other hand, there are several datasets/workloads that are not be predictable at all and an attempt to figure out predictive models for such datasets/workloads may be futile. An example of unpredictable workload/dataset is a research dataset which may get accessed in a totally ad hoc manner at anytime and hence, is not a good candidate for predictive capability.

Predictive GreenHDFS results in higher energy costs savings, storage-efficiency, and performance than Reactive GreenHDFS; however, making predictive capability work for different workloads is an involved process. While Reactive GreenHDFS allows lower energy costs savings, performance, and storage-efficiency than Predictive GreenHDFS, Reactive GreenHDFS can work across multiple workloads as long as its policy thresholds are chosen judiciously. And, Reactive GreenHDFS performs almost as well as Predictive GreenHDFS with news-server-like-access pattern workloads. In the news-server-like workloads, a majority of the computational jobs on the files happen soon after the files' creation and the jobs go down in number as the files age. With such workloads, once a file is deemed dormant and is migrated to the $Inactive$ zone, the chances of it getting accessed again in the future are low. This results in less server wake ups leading to higher energy cost savings, and lower performance impact. A majority of the log processing workloads such as clickstream processing, and multimedia workloads such as videos-on-demand, and movies-on-demand have a news-server-like access pattern and are a very good fit for Reactive GreenHDFS. Reactive GreenHDFS does result in energy savings and storage space savings even with non-news-server-like access patterns as long as there is a definite skew in the access patterns of the files, their popularity, and life spans. GreenHDFS is also a great fit for archival, disaster recovery and backup data which is common even in the context of Big Data Analytics.

In the Big Data clusters, the lowest energy-efficiency region corresponds to their most common operating mode, as shown by a six-month average CPU utilization study done at Google [28]. Per this study, servers operate most of the time between 10%-50% of their maximum utilization levels. Thus, there is a significant opportunity to consolidate computational load on the $Active$ zone and push the servers to operate closer to their energy-efficient range (i.e., 70%-80% of peak utilization). In the $Inactive$ zone, on the other hand, scaling-down servers aids in deriving energy-proportionality during idle utilization. Since the power draw in the inactive power states is very close to zero, by transitioning the servers to inactive power state, GreenHDFS provides a mechanism to have an energy-proportional behavior in data centers built with non-energy-proportional components during periods of average load. And, the compute capacity of the $Inactive$ zone can always be harnessed under peak load scenarios by waking up the sleeping

servers.

Summary: There are several advantages of GreenHDFS's data-centric, data-classification-driven cluster zone partitioning:

- Allows scale-down even in the Big Data analytics environment which presents significant scale-down challenges courtesy of its different storage, compute, and network model; thereby, resulting in server operating energy costs savings.

- Big Data analytics cloud mandates a different compute model whereby, *Data-locality* is a really important consideration for high performance of computational jobs. Existing research is job-centric in nature and attempts to consolidate jobs on few servers during periods of low load so that the rest of the servers can be scaled-down. This brings forth *challenging* performance and energy trade-offs: respecting data-locality makes workload consolidation very constrained or almost impossible; ignoring data-locality results in power savings at high performance cost. GreenHDFS's data-centric energy management approach does not involve any such energy savings and performance trade-offs and computations always enjoy data-local high performance.

- Results in few power state transitions which is very important as frequent inactive-to-active power state transitions can impact hardware reliability; server components such as disks only have limited start/stop cycles.

- Allows thermal hot-spots reduction courtesy of its thermal-aware zone partitioning (Section 4.5.2) resulting in lower cooling energy costs in addition to lower server operating energy costs.

- Ensures high read and write data access performance to active and new incoming files. Big Data analytics is a data-intensive application in nature and hinges on high read access performance. Write performance is equally important as significant number of files are written daily and read right afterwards. GreenHDFS results in high data access performance as it uses: 1) zoned bit recording aware active file placement on outer disk cylinder zones which have higher transfer bandwidth than the inner disk cylinder zones (Section 4.5.1), 2) separation of small and large files (Section 4.6.2), 3) performance-aware zone partitioning (Section 4.5.2), and 4) proactive replication of high heat files (Section 4.6.4).

- Ensures long periods of idleness in the $Inactive$ zone servers by curtailing the need to

access data on the scaled-down servers courtesy of its energy-aware $Inactive$ zone chunking (Section 4.6.3), and file placement (Section 4.6.6) policies. This allows scale-down of the servers for long durations, resulting in server operating energy costs savings and lower performance impact.

- Allows a software-based method to achieve energy-proportionality even with non-energy-proportional components such as disks, network, and memory.

6.2 Data-Centric Cooling Energy Management

The surge in Big Data analytics cloud data centers brings in its wake burgeoning cooling energy costs. And, the impact is not just monetary, but also environmental given the significant water usage and carbon dioxide emissions involved. A majority of the existing research on run-time reduction of cooling energy costs relies on thermal-aware job placement/migration to reduce the cooling energy costs. These techniques face challenging performance and energy trade-offs in the Big Data analytics environment where data-locality is mandated for high performance: respecting data-locality in thermal-aware job placement constrains placement only to the servers that host the data resulting in lower cooling energy savings; neglecting data-locality results in cooling energy savings at performance cost.

GreenHDFS takes a different data-centric, cyber-physical cooling energy management approach which is fully data-placement-aware and doesn't involve energy savings and performance trade-offs. GreenHDFS is aware that the cooling and server power consumption depends on the load on the servers, and since jobs follow the files' chunks, the load on a server is directly dependent on the way the files' chunks are distributed across the servers. Thus, file distribution directly affects the load distribution in the cluster. In GreenHDFS, files are placed first in the cluster in a thermal- and energy-aware manner so that the computational jobs can then automatically enjoy energy costs savings in addition to high data-local performance by following files placed in a proactive, thermal- and energy-efficient manner.

Now, thermal-aware file placement can not be done in a naive fashion as discussed in Section 5.1.4 and there is a need for more sophisticated and predictive thermal-aware file placement algorithms and heuristics that can somehow glean information about the "future" computational

load profile of an incoming file and the "future" thermal-profile of the servers in the cluster and then place the file "now" on the most thermally-conducive server in the "future". The problem of optimally placing a collection of files in an informed, thermal-aware manner on a cluster can be decomposed into two subproblems.

The first subproblem relates to the prediction of each file's future computational load profile. GreenHDFS uses supervised machine learning on historical traces to predict the file's computational load profile. Since predicting the exact profile for each and every file in the cluster is simply not feasible both in terms of predictor complexity and accuracy, GreenHDFS uses predictive, simple, and yet accurate approximation for the computational load profile as discussed in Section 5.3.1. The choice of the approximation involves thermal-reliability, and efficient server packing trade-offs and requires a study of the statistical properties of the dataset. For the evaluation clickstream dataset at Yahoo!, average jobs arrival rate at per-minute granularity turned out to be a very good approximation as the standard deviation in the observed jobs arrival rate at different time intervals in the life time of the files was low.

The second subproblem relates to the actual optimal placement of files in the cluster subject to energy costs, temperature and capacity constraints of the cluster servers, given knowledge of each file's future computation job profile. Unfortunately, file placement problem is known to be NP-Hard [53]. The computational complexity of an idealized thermal-aware file placement problem suggest the need for an approximation algorithm to solve the general problem and one can come up with fairly complex approximation schemes. But in the context of the design and implementation of a robust file system, algorithmic simplicity is usually preferred. In that spirit, GreenHDFS uses heuristics for thermal-aware file placement.

Two variants of thermal-aware file placement heuristics, predictive and non-predictive are explored and compared with non-thermal-aware random file placement and data-agnostic thermal-aware job placement. Predictive file placement heuristic uses a combination of predicted and run-time cluster information to guide its thermal placement as elaborated in Section 5.3.2.2. The non-predictive heuristics rely only on run-time information from the cluster to guide thermal-aware file placement as discussed in Section 5.3.2.3.

Predictive heuristic works better than other heuristics in all dimensions: cooling costs savings, thermal-reliability, load-balancing, and storage utilization. Predictive thermal-aware file placement obviates the need to do reactive, thermal-reliability driven migration of load or files by ensuring that the files are placed upfront in such a manner that at no point in time, temperature of a server

exceeds its reliability-driven temperature upper-bound. Predictive heuristics are not sensitive to system and dataset parameters such as file size skew, job durations, delta in inlet temperatures of the servers, and file creation rate.

With the right combination of system variables, non-predictive technique can result in as good a behavior as predictive technique; however, it is very sensitive to monitoring service's sampling rate, file size skew, server utilization, and delta in cooling-efficiencies of the servers. If the sampling rate is fine-grained, the sensitivity of the non-predictive heuristic to other dataset and cluster parameters reduces significantly and the non-predictive heuristic starts showing the same advantages predictive scheme. Now, in the typical Big Data analytics clusters such as Hadoop or Google Map Reduce clusters, the master server is a single entity. In the case of GreenHDFS, master server is where the cyber-controller is incorporated. If the other servers start sending information to the master at a very fine-grained level, the master may get overwhelmed and may not be able to fulfill its main responsibilities such as that of answering metadata queries in a timely fashion.

Both the non-predictive and predictive thermal-aware file placement techniques require analysis of the dataset and an understanding of the cluster parameters. The predictive technique requires analysis to determine the predictability of the dataset and the associated workload. If the dataset is found to be predictable, as was the case with the evaluation dataset, relevant representative feature sets and supervised machine learning method need to be identified. The accuracy of predictions, retraining requirements, and costs of prediction need to be evaluated. In case of non-predictive technique, dataset analysis needs to be done from sensitivity angle. For example, if the dataset doesn't have a file skew and all the files are uniformly big, non-predictive technique will fare well and it can be used as is.

Summarizing, if the workload is predictable, predictive thermal-aware file placement should be used given its various benefits. If a workload is not predictive, but does have cluster and dataset characteristics that are conducive to the non-predictive technique, it is better to use the non-predictive technique instead of random, non-thermal-aware file placement technique as discussed and evaluated in the Section 5.4.

While several companies such as Microsoft, Google, and Yahoo! have announced new data centers that rely only on free-cooling or air- and water-side economization, there are a substantial number of traditional data centers that still rely on air- or liquid-cooling either entirely or in-part coupled with economization. The existing data centers may not be located in economizer

friendly geographic locations and can not always be retrofitted with the economizer modes as incorporation of the economizer may require space for ductwork, and addition of heat exchangers which may not be available in the data center. Power usage effectiveness (PUE) is the ratio of the total building power to the IT power. A majority of the data centers still have a high PUE of 1.8 as per a 2011 survey of 500 data centers [2]. The high PUE is because of the power overheads such as 42% overhead of the CRAC and the chiller. GreenHDFS aids in reducing the thermal hot-spots significantly courtesy of its thermal-aware zone partitioning whereby cooling-inefficient servers are reserved for dormant files. Thermal hot-spots are the number one cause of high cooling energy costs in the traditionally cooled data centers that can not be retrofitted with air containment techniques. Alleviation of the thermal hot-spots reduces cooling energy costs significantly. GreenHDFS also makes it possible to raise the cooling set point temperature of the cooling system which increases the efficiency of the cooling system; thereby, further reducing the cooling costs.

The cooling energy costs of the data centers can be significantly cut down if the data center is capable of supporting economizer mode and can operate for extended periods of time in the economizer mode. The cooling-efficacy in the economizer mode depends on the temperature and the quality of the outside air. If the ambient temperature of the air is high in the economizer mode, the inlet temperatures of the servers may get high, thereby reducing their cooling-efficiency and resulting in a temperature rise of the servers. Now, the servers can safely operate only within their rated, allowable temperature range. For example, the operating temperature of Dell Blade is $50^oF - 95^oF$, IBM Blade is $50^oF - 95^oF$, and NetAppStorage is $50^oF - 104^oF$. If the server temperature goes out of the allowable range, failure rates increase. Usage of economizer mode for extended hours and an increase in the set point of the cooling system hinge on the system's ability to ensure thermal-reliability. GreenHDFS thermal-aware file placement is done in a manner that ensures that servers remain within their thermally-reliable temperature range. GreenHDFS is cognizant of the differences in the load tolerance of each server and places files on the servers accordingly as discussed in the Section 5.1.6. GreenHDFS results in much lower thermal-profile compared with random, non-thermal-aware file placement or the state-of-the-art thermal-aware job placement techniques even when all the techniques are subjected to the same load. Thus, GreenHDFS thermal-aware file placement enables the usage of free-cooling and economization modes for longer periods of time resulting in cooling energy costs savings.

6.3 Evaluation Results

Evaluation results with one-month long real-world traces from production Big Data Analytics cluster show up to a 59% reduction in the cooling energy costs, up to 26% reduction in the compute energy costs, a 9x better average response time than thermal-aware job placement techniques which are inherently data-placement agnostic in nature, more uniform thermal-profile, thermal-reliability assurance, and lower overall temperature in the cluster. Data needs to be a first-class object in Big Data Analytics and GreenHDFS's evaluation results show that its data-centric energy management approach is a step in the right direction. GreenHDFS also throws light on the importance of historical trace analysis (a Big Data analytics problem in itself given the humongous size of the traces) to guide the data-placement and energy-management policies in the Big Data analytics cluster. There is a wealth of information about the files' access patterns, jobs arrival rates, evolution life spans, and sizes hidden in the system traces that can be leveraged for higher energy costs savings and performance.

6.4 Future Work

GreenHDFS allows overall energy costs savings, ensures thermal-reliability, provides a software-based mechanism for incorporation of energy-proportionality with non-energy-proportional components in a self-adaptive and automated manner while allowing high data access performance. GreenHDFS can be extended in a variety of ways in the future to provide even higher energy costs savings. GreenHDFS circle of influence can also be expanded to include a wider variety of Big Data analytics workloads and datasets.

In its current version, GreenHDFS doesn't do any power management in the $Active$ zone. $Active$ zone hosts active files and the computational jobs targeted to these files may have strict service level agreements and completion deadlines; thereby, requiring performance guarantees. Power management in the $Active$ zone needs to be done in a manner that doesn't degrade performance. A variety of active-mode power management schemes such as dynamic voltage and frequency scaling (DVFS) for processors, multi-speed disks that are capable of operating in different rotational speeds for storage, and energy-aware caching and data layouts for the memory can be applied to the $Active$ zone servers to further save on server operating energy costs.

GreenHDFS uses statistical distributions to approximate computational load-profile of the files in its thermal-aware file placement which is geared towards reducing cooling energy costs and ensuring thermal-reliability of the servers. More sophisticated distributions can be considered to represent files' jobs arrival rate in the thermal-aware file placement, which could result in even higher cooling energy costs savings by providing a more accurate match between the incoming file's computational load-profile and anticipated thermal-profile of the servers.

In addition to the Reactive and Predictive mode of operation in which the onus is on GreenHDFS to learn about files' relevant attributes, GreenHDFS may also work very well with user-provided hints and, such an hints-based approach can be further explored in the future. If the user is aware of the active life span of the file and its expected heat, and supplies the same as hints, GreenHDFS can override its own insights about the file and utilize the user-provided hints in its various policies such as file migration, and zone placement. However, hints-based approach will require careful analysis as one of the downsides of a purely hints based system centers around fairness. While several checks-and-bounds can be put in place, a user may still provide a fallacious hint that his/her file has very high heat; such a file will get special treatment in GreenHDFS and enjoy high performance. On the other hand, the insightful reactive and predictive modes in GreenHDFS are unbiased by user hints as they rely on self-generated insights and are inherently fairer. Also, hints-based system won't work well in workloads whereby the files are fed into the system by applications. For example, in the case of clickstream processing, daily thousands of web logs are copied over to the cluster directly from web servers and there is no user in the middle in this scenario to provide file-level hints. Given the large number of files created everyday, user-driven hints at a fine-grained per-file granularity are not even feasible. In such scenarios, GreenHDFS's capability to self-generate file insights becomes very important, and hints may need to be used in conjunction with self-generated file insights.

In the dissertation, GreenHDFS has been evaluated with real-world traces from Big Data analytics clickstream processing workload, which is one of the most important workload in the Internet services companies because of the monetary and business value. In the future, it would be worthwhile to expand GreenHDFS's scope to other Big Data analytics workloads and evaluate its efficacy with different workloads and datasets. The scope expansion requires an in-depth analysis of the dataset characteristics such as its evolution lifespans, sizes, job patterns, and jobs arrival rates to throw light on the behavior of the underlying files. An analysis of the predictability of the dataset/workload is also required. If the dataset/workload is found to be predictable, generation of predictive models and identification of relevant feature sets needs to happen.

REFERENCES

[1] http://www.gartner.com/resId=1818517.

[2] http://www.datacenterknowledge.com/archives/2011/05/10/uptime-institute-the-average-pue-is-1-8/.

[3] http://www.datacenterknowledge.com/archives/2009/04/09/data-centers-move-to-cut-water-waste/.

[4]

[5] EC2 Pricing. http://aws.amazon.com/ec2/pricing/.

[6] floVENT: A Computational Fluid Dynamics Simulator. http://www.mentor.com/products/mechanical/products/flovent.

[7] IDC. http://www.idc.com.

[8] Pig. http://pig.apache.org/.

[9] Seagate ES.2. http://www.seagate.com/staticfiles/support/disc/manuals/NL35Series&BCESSeries/BarracudaES.2Series/100468393e.pdf, 2008.

[10] SMSC, 2008.

[11] Google tco spreadsheet. http://spreadsheets.google.com/pub?key=phRJ4tNx2bFOHgYskgpoXAA&output=xls, 2009.

[12] Datacenter locations in usa, 2010.

[13] Energy Rates, 2010.

[14] Personal Communication with Hong Tang, 2010.

[15] Cost of Power in Large-Scale Data Centers. http://perspectives.mvdirona.com, November, 2008.

[16] Abbasi, Z., Varsamopoulos, G., and Gupta, S. K. S. Thermal Aware Server Provisioning and Workload Distribution for Internet Data Centers. In *HPDC '10: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing* (New York, NY, USA, 2010), ACM, pp. 130–141.

[17] Aboughazaleh, N., Childers, B., Moss, D., and Melhem, R. Energy conservation in memory hierarchies using poweraware cached-dram. In *in Proc. of the Dagstuhl Seminar on Power-aware Computing Systems. Dagstuhl Research Online Publication Server* (2005).

[18] AbouGhazaleh, N., Childers, B. R., Mossé, D., and Melhem, R. G. Near-memory caching for improved energy consumption. *IEEE Trans. Comput. 56*, 11 (Nov. 2007), 1441–1455.

[19] Agrawal, N., Prabhakaran, V., Wobber, T., Davis, J. D., Manasse, M., and Panigrahy, R. Design tradeoffs for ssd performance. In *ATC'08: USENIX Annual Technical Conference*.

[20] Aguilera, M. K., Keeton, K., and Merchant, A. Improving recoverability in multi-tier storage systems. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2007).

[21] Akyürek, S., and Salem, K. Adaptive block rearrangement. *ACM Trans. Comput. Syst. 13* (May 1995), 89–121.

[22] Alpaydin, E. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2004.

[23] Amur, H., Cipar, J., Gupta, V., Ganger, G. R., Kozuch, M. A., and Schwan, K. Robust and Flexible Power-Proportional Storage. In *Proceedings of the 1st ACM Symposium on Cloud Computing* (New York, NY, USA, 2010), SoCC'10, ACM, pp. 217–228.

[24] Amur, H., and Schwan, K. Achieving power-efficiency in clusters without distributed file system complexity.

[25] Ananthanarayanan, G., and Katz, R. H. Greening the Switch. Tech. Rep. UCB/EECS-2008-114, EECS Department, University of California, Berkeley, Sep 2008.

[26] Hadoop. `http://hadoop.apache.org/`.

[27] ASHRAE. Thermal Guidelines for Data Processing Environments Expanded Data Center Classes and Usage Guidance.

[28] Barroso, L. A., and Hölzle, U. The Case for Energy-Proportional Computing. *Computer 40*, 12 (2007).

[29] Bash, C., and Forman, G. Cool Job Allocation: Measuring the Power Savings of Placing Jobs at Cooling-Efficient Locations in the Data Center. In *USENIX Annual Technical Conference* (2007), ATC'07, pp. 363–368.

[30] Bash, C., Patel, C., and Sharma, R. Dynamic Thermal Management of Air Cooled Data Centers. In *Thermal and Thermomechanical Phenomena in Electronics Systems, 2006. ITHERM '06. The Tenth Intersociety Conference on* (30 2006-june 2 2006), pp. 8 pp. –452.

[31] Beitelmal, M. H., and Patel, C. D. Model-Based Approach for Optimizing a Data Center Centralized Cooling System. Tech. rep., Hewlett Packard, 2006.

[32] Belady, C. In the Data Center, Power and Cooling Costs more than the IT Equipment it Supports, February, 2010.

[33] Bell, G., Hey, T., and Szalay, A. Beyond the data deluge.

[34] Blanas, S., Patel, J. M., Ercegovac, V., Rao, J., Shekita, E. J., and Tian, Y. A Comparison of Join Algorithms for Log Processing in MapReduce. In *Proceedings of the 2010 International Conference on Management of Data* (New York, NY, USA, 2010), SIGMOD '10, ACM, pp. 975–986.

[35] Bohrer, P., Elnozahy, E. N., Keller, T., Kistler, M., Lefurgy, C., McDowell, C., and Rajamony, R. The case for power management in web servers, 2002.

[36] Borthakur, D. Largest Hadoop Cluster at Facebook. `http://hadoopblog.blogspot.com/2010/05/facebook-has-worlds-largest-hadoop.html`.

[37] Borthakur, D. Hierarchical Directory Organization at Facebook, 2011.

[38] Breen, T., Walsh, E., Punch, J., Shah, A., and Bash, C. From Chip to Cooling Tower Data Center Modeling: Part I Influence of Server Inlet Temperature and Temperature Rise Across Cabinet. In *Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 2010 12th IEEE Intersociety Conference on*, pp. 1 –10.

[39] Brill, K. G. Data Center Energy Efficiency and Productivity., 2007.

[40] Brill, K. G. The Invisible Crisis in the Data Center: The Economic Meltdown of Moore Law, 2007.

[41] Buckinx, W., and Poel, D. V. d. Predicting online purchasing behavior. Working papers of faculty of economics and business administration, ghent university, belgium, Ghent University, Faculty of Economics and Business Administration, 2003.

[42] Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., and Doyle, R. P. Managing Energy and Server Resources in Hosting Centers. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles* (New York, NY, USA, 2001), SOSP '01, ACM, pp. 103–116.

[43] Chen, G., He, W., Liu, J., Nath, S., Rigas, L., Xiao, L., and Zhao, F. Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation* (Berkeley, CA, USA, 2008), NSDI'08, USENIX Association, pp. 337–350.

[44] Chen, S., Joshi, K. R., Hiltunen, M. A., Schlichting, R. D., and Sanders, W. H. CPU Gradients: Performance-Aware Energy Conservation in Multitier Systems. *International Conference on Green Computing 0* (2010), 15–29.

[45] Chen, Y., Das, A., Qin, W., Sivasubramaniam, A., Wang, Q., and Gautam, N. Managing Server Energy and Operational Costs in Hosting Centers. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems* (New York, NY, USA, 2005), SIGMETRICS '05, ACM, pp. 303–314.

[46] Chen, Y., Gmach, D., Hyser, C., Wang, Z., Bash, C., Hoover, C., and Singhal, S. Integrated Management of Application Performance, Power and Cooling in Data Centers. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pp. 615 –622.

[47] D. Blaauw, S. D., and Lee, Y. Managing Variations through Adaptive Design Techniques, 2010.

[48] Data-intensive computing. `http://research.yahoo.com/files/BryantDISC`.

[49] Dean, J., Ghemawat, S., and Inc, G. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design and Implementation*, OSDI'04, USENIX Association.

[50] Deng, Q., Meisner, D., Ramos, L., Wenisch, T. F., and Bianchini, R. Memscale: active low-power modes for main memory. *SIGPLAN Not. 46*, 3 (Mar. 2011), 225–238.

[51] Deng, Y., Wang, F., and Helian, N. Eed: Energy efficient disk drive architecture. *Information Sciences 178*, 22 (November 2008), 4403–4417.

[52] Department, P. C., Chatterjee, P., Hoffman, D. L., and Novak, T. P. Modeling the Clickstream: Implications for Web-Based Advertising Efforts. *Marketing Science 22* (2000), 520–541.

[53] Dowdy, L. W., and Foster, D. V. Comparative Models of the File Assignment Problem. *ACM Comput. Surv. 14*, 2 (June 1982), 287–313.

[54] Economist. Data, data everywhere. `http://www.economist.com/node/15557443`.

[55] Ellard, D., Mesnier, M., Thereska, E., Ganger, G. R., and Seltzer, M. Attribute-based prediction of file properties, 2003.

[56] EMC. Emc greenplum. `http://www.emc.com/microsites/bigdata/why-big-data-analytics.htm`.

[57] EPA. EPA Report on Server and Data Center Energy Efficiency. Tech. rep., U.S. Environmental Protection Agency, 2007.

[58] Essary, D., and Amer, A. Predictive Data Grouping: Defining the Bounds of Energy and Latency Reduction through Predictive Data Grouping and Replication. *Trans. Storage* (May 2008).

[59] Fan, X., Weber, W.-D., and Barroso, L. A. Power Provisioning for a Warehouse-Sized Computer. In *ISCA '07: Proceedings of the 34th Annual International Symposium on Computer Architecture* (New York, NY, USA, 2007), ACM, pp. 13–23.

[60] Ganesh, L., Weatherspoon, H., Balakrishnan, M., and Birman, K. Optimizing power consumption in large scale storage systems. In *Proceedings of the 11th USENIX workshop on Hot topics in operating systems* (Berkeley, CA, USA, 2007), USENIX Association, pp. 9:1–9:6.

[61] Ghemawat, S., Gobioff, H., and Leung, S.-T. The Google File System. In *Proceedings of the Nineteenth ACM Smposium on Operating Systems Principles* (New York, NY, USA, 2003), SOSP '03, ACM, pp. 29–43.

[62] Google. Official Google Blog. `http://googleblog.blogspot.com/2008/11/sorting-1pb-with-mapreduce.html`.

[63] Griffioen, J., and Appleton, R. Reducing file system latency using a predictive approach. In *Proceedings of the USENIX Summer 1994 Technical Conference on USENIX Summer 1994 Technical Conference - Volume 1* (Berkeley, CA, USA, 1994), USTC'94, USENIX Association, pp. 13–13.

[64] Gunda, P. K., Ravindranath, L., Thekkath, C. A., Yu, Y., and Zhuang, L. Nectar: Automatic management of data and computation in datacenters. In *Proceedings of the 9th Conference on Symposium on Operating Systems Design and Implementation* (2010), OSDI'10.

[65] Gurumurthi, S., Sivasubramaniam, A., Kandemir, M., and Franke, H. Drpm: Dynamic speed control for power management in server class disks. In *In Proceedings of the International Symposium on Computer Architecture (ISCA* (2003), pp. 169–179.

[66] Hamilton, J. Cooperative Expendable Micro-Slice Servers (CEMS): Low Cost, Low Power Servers for Internet-Scale Services, 2009.

[67] Heath, T., Centeno, A. P., George, P., Ramos, L., Jaluria, Y., and Bianchini, R. Mercury and Freon: Temperature Emulation and Management for Server Systems. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems* (New York, NY, USA, 2006), ASPLOS-XII, ACM, pp. 106–116.

[68] Hoelzle, U., and Barroso, L. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan and Claypool Publishers, May 29, 2009.

[69] Horvath, T., Abdelzaher, T., Skadron, K., Member, S., and Liu, X. Dynamic Voltage Scaling in Multitier Web Servers with End-to-End Delay Control. In *IEEE Transactions on Computers* (2007), press.

[70] HP. Hp vertica. `http://www.vertica.com/`.

[71] Huang, H., Pillai, P., and Shin, K. G. Design and implementation of power-aware virtual memory. In *Proceedings of the annual conference on USENIX Annual Technical Conference* (Berkeley, CA, USA, 2003), ATEC '03, USENIX Association, pp. 5–5.

[72] Hutton, R. Hadoop in Europe. `http://www.cloudera.com/blog/2010/03/why-europes-largest-ad-targeting-platform-uses-hadoop`, 2010.

[73] IBM. Ibm infosphere biginsights. `http://www-01.ibm.com/software/data/infosphere/biginsights/`.

[74] Intel. Intel Atom Processor Z560. `http://ark.intel.com/Product.aspx?id=49669\&processor=Z560\&spec-codes=SLH63`.

[75] Intel. Quad-Core Intel Xeon Processor 5400 Series. `http://www.intel.com/Assets/en_US/PDF/prodbrief/xeon-5400-animated.pdf`, 2008.

[76] Intel. Quad-core intel xeon processor 5400 series price. `http://ark.intel.com/ProductCollection.aspx?series=33905`, 2008.

[77] Jian-Hui, Z., and Chun-Xin, Y. Design and simulation of the cpu fan and heat sinks. *Components and Packaging Technologies, IEEE Transactions on 31*, 4 (dec. 2008), 890 –903.

[78] Kaushik, R. T., and Bhandarkar, M. GreenHDFS: Towards an Energy-Conserving, Storage-Efficient, Hybrid Hadoop Compute Cluster. In *Proceedings of International Conference on Power-Aware Computing and Systems* (Berkeley, CA, USA, 2010), HotPower, USENIX Association, pp. 1–9.

[79] Kaushik, R. T., Bhandarkar, M., and Nahrstedt, K. Evaluation and Analysis of GreenHDFS: A Self-Adaptive, Energy-Conserving Variant of the Hadoop Distributed File System. In *Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science* (2010), CloudCom, IEEE.

[80] Kaushik, R. T., Cherkasova, L., Campbell, R., and Nahrstedt, K. 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* (New York, NY, USA, 2010), HPDC '10, ACM, pp. 332–335.

[81] Kaushik, R. T., Egashira, R., Abdelzaher, T., and Nahrstedt, K. Predictive Data and Energy Management in GreenHDFS. In *Proceedings of International Conference on Green Computing* (2011), IGCC, IEEE.

[82] Kaushik, R. T., and Nahrstedt, K. T^*: A Data-Centric Cooling Energy Costs Reduction Approach for Big Data Analytics Cloud. In *Appear in Proceedings of 2012 International Conference for High Performance Computing, Networking, Storage and Analysis* (New York, NY, USA, 2012), SC '12, ACM.

[83] Kist, A. A., and Aldraho, A. Dynamic topologies for sustainable and energy efficient traffic routing. *Comput. Netw. 55*, 9 (June 2011), 2271–2288.

[84] Konstantin, S., Kuang, H., Radia, S., and Chansler, R. The Hadoop Distributed File System. *MSST* (2010).

[85] Lang, W., and Patel, J. M. Energy Management for MapReduce Clusters. *Proceedings VLDB Endow. 3* (September 2010), 129–139.

[86] Lang, W., Patel, J. M., and Shankar, S. Wimpy Node Clusters: What about Non-Wimpy Workloads? In *Proceedings of the Sixth International Workshop on Data Management on New Hardware* (New York, NY, USA, 2010), DaMoN '10, ACM, pp. 47–55.

[87] Le, K., Bianchini, R., Martonosi, M., and Nguyen, T. Cost- and Energy-Aware Load Distribution Across Data Centers. In *HotPower* (2009).

[88] Le Sueur, E., and Heiser, G. Dynamic voltage and frequency scaling: the laws of diminishing returns. In *Proceedings of the 2010 international conference on Power aware computing and systems* (Berkeley, CA, USA, 2010), HotPower'10, USENIX Association, pp. 1–8.

[89] Lebeck, A. R., Fan, X., Zeng, H., and Ellis, C. Power aware page allocation. *SIGOPS Oper. Syst. Rev. 34*, 5 (Nov. 2000), 105–116.

[90] Lee, L.-W., Scheuermann, P., and Vingralek, R. File assignment in parallel i/o systems with minimal variance of service time. *IEEE Trans. on Computers 49* (2000), 127–140.

[91] Leverich, J., and Kozyrakis, C. On the Energy (In)efficiency of Hadoop Clusters. *SIGOPS Operating Systems Review 44* (March 2010), 61–65.

[92] Li, D., and Wang, J. Eeraid: energy efficient redundant and inexpensive disk array. In *EW11: Proceedings of the 11th workshop on ACM SIGOPS European workshop* (New York, NY, USA, 2004), ACM, p. 29.

[93] Li, S., Le, H., Pham, N., Heo, J., and Abdelzaher, T. Joint Optimization of Computing and Cooling Energy: Analytic Model and A Machine Room Case Study, 2012.

[94] Li, X., Gupta, R., Adve, S. V., and Zhou, Y. Cross-component energy management: Joint adaptation of processor and memory. *ACM Trans. Archit. Code Optim. 4*, 3 (Sept. 2007).

[95] Lin, C.-H., Yang, C.-L., and King, K.-J. Ppt: joint performance/power/thermal management of dram memory for multi-core systems. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design* (New York, NY, USA, 2009), ISLPED '09, ACM, pp. 93–98.

[96] Lombardo, A., Panarello, C., and Schembra, G. Achieving energy savings and qos in internet access routers. *SIGMETRICS Perform. Eval. Rev. 38*, 3 (Jan. 2011), 76–80.

[97] Luca Parolini, Bruno Sinopoli, B. H. K., and Wang, Z. A Cyber-Physical-System Approach to Data Center Modeling and Control for Energy Efficiency. In *Proceedings of the IEEE, Special Issue on Cyber-Physical Systems* (December 2011).

[98] Lustre. `www.lustre.org`, 2010.

[99] Madhyastha, T. M., and Reed, D. A. Input/output access pattern classification using hidden markov models. In *Proceedings of the fifth workshop on I/O in parallel and distributed systems* (New York, NY, USA, 1997), IOPADS '97, ACM, pp. 57–67.

[100] Mahajan, R., pin Chiu, C., and Chrysler, G. Cooling a microprocessor chip. *Proceedings of the IEEE 94*, 8 (aug. 2006), 1476 –1486.

[101] McKinsey. Big data: The next frontier for innovation, competition, and productivity. `http://www.mckinsey.com/insights/mgi/research/technology_and_innovation/big_data_the_next_frontier_for_innovation`.

[102] Meisner, D., Gold, B. T., and Wenisch, T. F. PowerNap: Eliminating Server Idle Power. In *Proceeding of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems* (New York, NY, USA, 2009), ASPLOS '09, ACM, pp. 205–216.

[103] Merkel, A., and Bellosa, F. Memory-Aware Scheduling for Energy Efficiency on Multicore Pocessors. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems* (Berkeley, CA, USA, 2008), HotPower'08, USENIX Association, pp. 1–1.

[104] Micron. Ddr2 sdram sodimm, 2004.

[105] Miyoshi, A., Lefurgy, C., Hensbergen, E. V., Rajamony, R., and Rajkumar, R. Critical power slope: Understanding the runtime effects of frequency scaling. In *In Proceedings of the 16th Annual ACM International Conference on Supercomputing* (2002), pp. 35–44.

[106] Moore, J., Chase, J., and Ranganathan, P. Weatherman: Automated, Online and Predictive Thermal Mapping and Management for Data Centers. In *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on* (june 2006), pp. 155 – 164.

[107] Moore, J., Chase, J., Ranganathan, P., and Sharma, R. Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers. In *Proceedings of the Annual Conference on USENIX Annual Technical Conference* (Berkeley, CA, USA, 2005), ATEC '05, USENIX Association, pp. 5–5.

[108] Narayanan, D., Donnelly, A., and Rowstron, A. Write Off-Loading: Practical Power Management for Enterprise Storage. *ACM Transaction of Storage 4*, 3 (2008), 1–23.

[109] Niccolini, L., Iannaccone, G., Ratnasamy, S., Chandrashekar, J., and Rizzo, L. Building a power-proportional software router. In *Proceedings of the 2012 USENIX conference on Annual Technical Conference* (Berkeley, CA, USA, 2012), USENIX ATC'12, USENIX Association, pp. 8–8.

[110] Niemann, J., Bean, J., and Avelar, V. Economizer modes of data center cooling systems.

[111] Oly, J., and Reed, D. A. Markov model prediction of i/o requests for scientific applications. In *Proceedings of the 16th international conference on Supercomputing* (New York, NY, USA, 2002), ICS '02, ACM, pp. 147–155.

[112] Oracle. Oracle exadata. `http://www.oracle.com/us/products/database/exadata/overview/index.html`.

[113] Patel, C., Bash, E., Sharma, R., and Beitelmal, M. Smart Cooling of Data Centers. In *Proceedings of PacificRim/ASME International Electronics Packaging Technical Conference and Exhibition* (2003), IPACK'03.

[114] Patel, C., Sharma, R., Bash, C. E., and Beitelmal, A. Thermal Considerations in Cooling Large Scale High Compute Density Data Centers. In *Eight Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems* (2002), ITherm'02, pp. 767–776.

[115] Patel, C. D., Bash, C. E., Belady, C., Stahl, L., and Sullivan, D. Computational Fluid Dynamics Modeling of High Compute Density Data Centers to Assure System Inlet Air Specifications.

[116] Patel, C. D., and Shah, A. J. Cost Model for Planning, Development and Operation of a Data Center. Tech. rep., HP Labs, 2005.

[117] Introduction to power supplies, 2002.

[118] Riska, A., Mi, N., Casale, G., and Smirni, E. Feasibility Regions: Exploiting Tradeoffs between Power and Performance in Disk Drives. In *Second Workshop on Hot Topics in Measurement and Modeling of Computer Systems (HotMetrics'09), Seattle, WA, 2009* (2009).

[119] Ryan, T. P. *Modern Regression Methods*. Wiley Series in Probability and Statistics, 1997.

[120] Sarood, O., and Kale, L. V. A 'Cool' Load Balancer for Parallel Applications. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis* (New York, NY, USA, 2011), SC '11, ACM, pp. 21:1–21:11.

[121] Scheuermann, P., Weikum, G., and Zabback, P. Data partitioning and load balancing in parallel disk systems. *The VLDB Journal 7*, 1 (Feb. 1998), 48–66.

[122] Schmidt, R. R., Karki, K. C., Kelkar, K. M., Radmehr, A., and Patankar, S. V. Measurements and Predictions of the Flow Distribution Through Perforated Tiles in Raised-Floor Data Centers. In *Proceedings of The Pacific Rim/ASME International Electronic Packaging, IPACK* (2001).

[123] Seidman, J. Hadoop at Orbitz. http://www.cloudera.com/resource/hw10_hadoop_and_hive_at_orbitz, 2010.

[124] Shang, Y., Li, D., and Xu, M. Energy-aware routing in data center network. In *Proceedings of the first ACM SIGCOMM workshop on Green networking* (New York, NY, USA, 2010), Green Networking '10, ACM, pp. 1–8.

[125] Sharma, N., Barker, S., Irwin, D., and Shenoy, P. Blink: Managing Server Clusters on Intermittent Power. In *Proceedings of the Sixteenth International Conference on Architectural Support for Programming Languages and Operating Systems* (New York, NY, USA, 2011), ASPLOS '11, ACM, pp. 185–198.

[126] Sharma, R. K., Bash, C. E., Patel, C. D., Friedrich, R. J., and Chase, J. S. Balance of Power: Dynamic Thermal Management for Internet Data Centers. *IEEE Internet Computing 9* (2005), 42–49.

[127] Snowdon, D., Ruocco, S., and Heiser, G. Power management and dynamic voltage scaling: Myths and facts, 2005.

[128] Snowdon, D. C., Le Sueur, E., Petters, S. M., and Heiser, G. Koala: a platform for os-level power management. In *Proceedings of the 4th ACM European conference on Computer systems* (New York, NY, USA, 2009), EuroSys '09, ACM, pp. 289–302.

[129] Intel X25-E SATA Solid-State Drive. http://www.intel.com/design/flash/nand/extreme/index.htm, 2008.

[130] Storer, M. W., Greenan, K. M., Miller, E. L., and Voruganti, K. Pergamum: Replacing tape with energy efficient, reliable, disk-based archival storage. In *In FAST-2008: 6th Usenix Conference on File and Storage Technologies* (2008).

[131] Sudan, K., Chatterjee, N., Nellans, D., Awasthi, M., Balasubramonian, R., and Davis, A. Micro-pages: increasing dram efficiency with locality-aware data placement. In *Proceedings of the fifteenth edition of ASPLOS on Architectural support for programming languages and operating systems* (New York, NY, USA, 2010), ASPLOS '10, ACM, pp. 219–230.

[132] Sullivan, R. Alternating Cold and Hot Aisles Provides more Reliable Cooling for Server Farms.

[133] Sweiger, M., Madsen, M. R., Langston, J., and Lombard, H. *Clickstream Data Warehousing*. John Wiley and Sons, Inc., 2002.

[134] Tang, Q., Gupta, S., and Varsamopoulos, G. Energy-Efficient Thermal-Aware Task Scheduling for Homogeneous High-Performance Computing Data Centers: A Cyber-Physical Approach. *Parallel and Distributed Systems, IEEE Transactions on 19*, 11 (nov. 2008), 1458 –1472.

[135] Thusoo, A., Shao, Z., Anthony, S., Borthakur, D., Jain, N., Sen Sarma, J., Murthy, R., and Liu, H. Data Warehousing and Analytics Infrastructure at Facebook. In *Proceedings of the International Conference on Management of Data* (New York, NY, USA, 2010), SIGMOD '10, ACM, pp. 1013–1020.

[136] Tjioe, J., Widjaja, R., Lee, A., and Xie, T. Dora: A dynamic file assignment strategy with replication. *Parallel Processing, International Conference on 0* (2009), 148–155.

[137] Tolia, N., Wang, Z., Marwah, M., Bash, C., Ranganathan, P., and Zhu, X. Delivering Energy Proportionality with Non Energy-Proportional Systems: Optimizing the Ensemble. In *Proceedings of the 2008 Conference on Power Aware Computing and Systems* (Berkeley, CA, USA, 2008), HotPower'08, USENIX Association, pp. 2–2.

[138] Varma, A., Ganesh, B., Sen, M., Choudhury, S. R., Srinivasan, L., and Jacob, B. A Control-Theoretic Approach to Dynamic Voltage Scheduling. In *Proceedings of the 2003 International Conference on Compilers, Architecture and Synthesis for Embedded Systems* (New York, NY, USA, 2003), CASES '03, ACM, pp. 255–266.

[139] Vasudevan, V., Franklin, J., Andersen, D., Phanishayee, A., Tan, L., Kaminsky, M., and Moraru, I. Fawndamentally Power-Efficient Clusters. In *Proceedings of the 12th Conference on Hot Topics in Operating Systems* (Berkeley, CA, USA, 2009), HotOS'09, USENIX Association, pp. 22–22.

[140] Wang, Z., Tolia, N., and Bash, C. Opportunities and Challenges to Unify Workload, Power, and Cooling Management in Data Centers. In *Proceedings of the Fifth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks* (New York, NY, USA, 2010), FeBiD '10, ACM, pp. 1–6.

[141] Weddle, C., Oldham, M., Qian, J., and i Andy Wang, A. PARAID: The Gear Shifting Power-Aware RAID. In *Proceedings of USENIX Conference on File and Storage Technologies (FAST'07)* (2007), USENIX Association.

[142] Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D. E., and Maltzahn, C. Ceph: A Scalable, High-Performance Distributed File System. In *Proceedings of the 7th Conference on Symposium on Operating Systems Design and Implementation* (2006), OSDI '06.

[143] Weiser, M., Welch, B., Demers, A., and Shenker, S. Scheduling for reduced cpu energy. In *Proceedings of the 1st USENIX conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 1994), OSDI '94, USENIX Association.

[144] White, T. *Hadoop: The Definitive Guide*. O'Reilly Media, May, 2009.

[145] Wilkes, J., Golding, R., Staelin, C., and Sullivan, T. The hp autoraid hierarchical storage system. *ACM Transactions of Computer System 14*, 1 (1996), 108–136.

[146] Zheng, H., Lin, J., Zhang, Z., Gorbatov, E., David, H., and Zhu, Z. Mini-rank: Adaptive dram architecture for improving memory power efficiency. In *Proceedings of the 41st annual IEEE/ACM International Symposium on Microarchitecture* (Washington, DC, USA, 2008), MICRO 41, IEEE Computer Society, pp. 210–221.

[147] Zhu, Q., Chen, Z., Tan, L., Zhou, Y., Keeton, K., and Wilkes, J. Hibernator: helping disk arrays sleep through the winter. *SIGOPS Operating Systems Review 39*, 5 (2005), 177–190.