Future Generation Computer Systems 88 (2018) 40–54

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Towards thermal-aware Hadoop clusters

Yi Zhou [a], Shubbhi Taneja [a,*], Gautam Dudeja [a], Xiao Qin [a,1], Jifu Zhang [b], Minghua Jiang [c], Mohammed I. Alghamdi [d]

[a] *Department of Computer Science and Software Engineering, Samuel Ginn College of Engineering, Auburn University, AL 36849-5347, United States*
[b] *School of Computer Science and Technology, Taiyuan University of Science and Technology, Taiyuan 030024, China*
[c] *College of Mathematics and Computer Science, Wuhan Textile University, Wuhan 430073, China*
[d] *Department of Computer Science, Al-Baha University, Al-Baha City, Saudi Arabia*

HIGHLIGHTS

- With the exponential increase in cooling costs of large-scale data centers, thermal management must be adequately addressed.
- Recent trends have discovered one of the critical reasons behind the temperature rise turns out to be heat re-circulation within data center.
- In this study, we proposed a new resource- and thermal-aware scheduler in Hadoop clusters; our scheduler aims at minimizing peak inlet temperature across all nodes to reduce power consumption and cooling cost in data centers.
- The proposed dynamic scheduler makes job scheduling decisions based on current CPU/disk utilization and number of tasks running as well as the feedback given by all slave nodes at run-time.
- We deploy a thermal model to project respective temperature of each slave node in addition to neighbors heat contribution.
- The thermal-aware scheduler is integrated with the Hadoops scheduling mechanism.
- We test our schedulers by running a set of Hadoop benchmarks (e.g., WordCount, DistributedGrep, PI and TeraSort) under various temperature conditions, utilization thresholds, and cluster sizes.
- The experimental results show that our scheduler achieves an average inlet temperature reduction by 2.5 °C over the default FIFO scheduler; our scheduling solution saves approximately 15% of cooling cost with marginal performance degradation.

ARTICLE INFO

Article history:
Received 19 January 2017
Received in revised form 16 December 2017
Accepted 25 April 2018
Available online 8 May 2018

Keywords:
Data centers
Thermal energy
Hadoop
Map-Reduce

ABSTRACT

With the exponential increase in cooling costs of large-scale data centers, thermal management must be adequately addressed. Recent trends have discovered one of the critical reasons behind the temperature rise turns out to be heat re-circulation within data center. In this study, we proposed a new resource- and thermal-aware scheduler in Hadoop clusters; our scheduler aims at minimizing peak inlet temperature across all nodes to reduce power consumption and cooling cost in data centers. The proposed dynamic scheduler makes job scheduling decisions based on current CPU/disk utilization and number of tasks running as well as the feedback given by all slave nodes at run-time. We deploy a thermal model to project respective temperature of each slave node in addition to neighbor's heat contribution. The thermal-aware scheduler is integrated with the Hadoop's scheduling mechanism. We test our schedulers by running a set of Hadoop benchmarks (e.g., WordCount, DistributedGrep, PI and TeraSort) under various temperature conditions, utilization thresholds, and cluster sizes. The experimental results show that our scheduler achieves an average inlet temperature reduction by 2.5 °C over the default FIFO scheduler; our scheduling solution saves approximately 15% of cooling cost with marginal performance degradation.

1. Introduction

Data centers have been one of the fastest growing power consumption unit in recent years. With the rapid increase usage of distributed processing frameworks (e.g., Yahoos's Hadoop), the energy cost of maintaining datacenters has become a vital economically factor [1]. Energy cost spent on cooling systems turns into a huge fraction of a data center's maintenance cost. Unsurprisingly,

* Corresponding author.
E-mail addresses: yzz0074@auburn.edu (Y. Zhou), shubbhi@auburn.edu (S. Taneja), gzd0004@auburn.edu (G. Dudeja), xqin@auburn.edu (X. Qin), zjf@tyust.edu.cn (J. Zhang), jmh@wtu.edu.cn (M. Jiang), mialmushilah@bu.edu.sa (M.I. Alghamdi).
1 Senior Member, IEEE.

https://doi.org/10.1016/j.future.2018.04.084
0167-739X/© 2018 Elsevier B.V. All rights reserved.

the cooling cost nowadays is much higher than the cost of building data centers housing IT equipments [2]. A handful of studies were conducted to reduce the energy cost in light of improving the server operational efficiency [3]. However, improving the energy efficiency of cooling systems is still an open issue. Few techniques were proposed to cut energy waste in order to boost the energy efficiency of cooling systems. Employing proper energy-efficient techniques, one may conserve 26% [4] of energy consumption without dramatically affect system performance.

Improving thermal efficiency of computing clusters is a popular way of reducing cooling cost of data centers [5,6]. The endless increasing demand of data-intensive computing leads to hardware capacity at a large scale (e.g., fast CPUs coupled with large storage systems), which in turn gives rise to excessive heat. A hot computing environment not only leads to high cooling cost, but also increases cooling-failure risks [7]. In this study, we show that lowering overall node temperatures helps in minimizing the energy consumption of cooling systems in the data centers.

1.1. Motivations

Three reasons motivate us work towards thermal-aware Hadoop clusters:

- Endless increasing energy cost of cooling systems for temperature maintenance in data centers.
- The lack of thermal-aware Hadoop schedulers that consider heat re-circulation, CPU utilization, and disk utilization.
- The pressing need of thermal models applied to estimate thermal impact and manage task assignments in order to improve the energy efficiency of cooling system.

Energy consumption has grown to be an enormous concern in maintaining data centers [8]. The energy consumption of information technology (IT) equipment (e.g., the computing servers, network cables, power supplies, security devices, thermostats, AC ducts) is approximately 40% of total energy consumption. Energy consumed by cooling systems accounts for approximately 45% in data centers [9]. Coupled with the fast growth of data centers, thousands of server racks are fitted in to floor areas as large as 9000 m^2. Data centers with high density of server racks augment the energy consumption incurred by the heat dissipation effect [10]. For example, due to the heat re-circulation, a rack server's heat dissipation requirements is roughly 30 times larger that those of servers in 1990 [11]. In 2013, U.S. data centers consumed an estimated 91 billion kilowatt-hours of electricity, which is the amount of electricity to power all entire households in the New York City twice over for a year. Meanwhile, power consumption of data centers is projected to increase to roughly 140 billion kilowatt-hours annually by 2020, costing American businesses $13 billion annually in electricity bills.

Energy conservation techniques at the application-level received much attention. For example, Li and Zhang proposed a power-aware MapReduce application model tailored for energy-efficient MapReduce programming [12]. Most existing energy consumption models ignore the correlations between heat re-circulation and hardware workload (e.g., CPU utilization and disk utilization) [13]. In this study, we aim to improve energy efficiency by seamlessly integrating multiple factors like heat re-circulation and resource utilization. Hardware resources such as CPU, disks, network switches emit enormous heat under heavy workload conditions. Cooling systems must control the overall temperature of data centers by supplying cool air to the air inlets.

Theoretically, heat is blown away by cool air generated from cooling systems. Heat re-circulation effect exists, because a portion of heat emitted from a server's outlet is recirculated into neighboring servers' inlets, thereby leading to temperature increase. Moreover, heavily utilized computing components such as CPU chips and disks heat up during the heat transfer from the inlet side to the outlet one, it becomes a main factor driving up temperatures. These combined effects give rise to an extra cost of cooling down the overall temperature in data centers [14]. Although static and dynamic approaches were proposed to alleviate the heat re-circulation effect, it remains an open issue of systematically minimizing heat re-circulation in Hadoop clusters. Therefore, reducing heat generated from computing components by distributing tasks in a reasonably thermal-aware manner becomes an effective way to reduce the cooling cost. Due to the lack of a heat-flow model characterizing the heat re-circulation in terms of server placements, thermal-aware scheduling in Hadoop becomes a challenging task [15]. We are motivated by this challenge to develop the thermal-aware Hadoop scheduler by incorporating an effective heat model, which effectively reduces the cooling cost by distributing tasks in a thermal-efficient way.

1.2. The basic idea and contributions

We design and implement a thermal-aware scheduling module in Hadoop. Our novel scheduler employs a thermal model to characterize heat-flow behaviors affecting cooling cost. The thermal model takes into account hardware resources (e.g., CPU and disk contribution to outlet temperatures) as well as the heat re-circulation effect. To make thermal awareness possible in the scheduler, our module keeps track of CPU/disk utilization and CPU/disk temperatures.

In our thermal-aware Hadoop scheduler, we leverage the lightweight heartbeat mechanism to communicate the thermal-related data from the Task Tracker to the Job Tracker. Heartbeat is referred to as a signal communicated between Task Trackers running on a data node and the Job Tracker on a name node. In case a node failed in responding to the heartbeat signal, the node will be treated as a faulty one. More specifically, heartbeat messages containing CPU/disk utilization and temperatures are periodically delivered from the Task Tracker to the Job Tracker. And in the Job Tracker, our scheduler relies on the thermal model to estimate heat emissions according to a given server related information (e.g., server placement). In case the Job Tracker receives no heartbeat message from a Task Tracker for an excessive amount of time, the task Tracker's node is labeled as unhealthy; tasks originally assigned to unhealthy nodes should be passed on to other healthy Task Trackers. Because the minimum interval of heartbeat is three seconds, we argue that this heartbeat mechanism is perfectly suited for real-time thermal data propagation.

In our thermal model, heat re-circulation effect is represented by a cross interference matrix coupled with rack server placements that are expressed as a height vector (i.e., the height of each server with respect to CRAC at floor). Thanks to the cross interference matrix and the height vector, the deployed thermal model is in the right position to estimate thermal temperatures associated with the rack servers.

Offered thermal information from heartbeat messages, the thermal model estimates inlet temperatures of computing nodes in a Hadoop cluster. If the estimated peak temperature of the specific server is blow a redline, our scheduler will assign map tasks to the server. In doing so, the scheduler is capable of dynamically balancing load across all the nodes while reducing heat emissions. Please note that throughout this manuscript, we use terms nodes and servers interchangeably.

Contributions. We make the following six contributions in this study.

- We introduce a thermal model, which takes into account CPU/disk utilization and CPU/disk temperatures.

- We propose a novel thermal-aware scheduler in Hadoop clusters aiming to minimize peak inlet temperatures across all nodes. The thermal-aware scheduler assigns jobs across the data nodes based on thermal feedback to reduce cooling cost in data centers.
- We run Hadoop benchmarks (i.e., *Word Count*, *Distributed Grep*, *Pi*, *Tera Sort*) to verify the effectiveness of our thermal-aware Hadoop. We conduct a series of experiments comparing our thermal-aware Hadoop with the traditional one.
- Our experimental indicates that compared with the existing solution, our thermal-aware Hadoop can effectively conserve energy consumption by anywhere between 7.2% to 20.7% while rejecting only 3% jobs. We show that our scheduler adequately reduces energy cost with a marginal performance overhead.

In summary, we devise a dedicated Hadoop dynamic thermal-aware scheduler, which seeks to take into account multiple thermal-affecting factors. As a result, our novel scheduler enables Hadoop clusters to manipulate task assignments in a thermal-efficient way to reduce the overall power consumption.

Paper Organization. The rest of the paper is organized as follows. Section 2 summarizes the related work. The design of our thermal-aware Hadoop as well as implementation issues are stated in Section 3. Section 4 evaluates the energy efficiency of our thermal-aware Hadoop with the non-thermal-aware one running various benchmarks. We discuss open issues and future research directions in Section 5. Finally, Section 6 concludes this research.

2. Related work

2.1. Thermal-efficient computing

Thermal-efficient computing has increasingly become prevalent in data centers, where cooling cost should be minimized. For example, Beloglazov et al. proposed an architectural framework for energy-efficient cloud computing, in which power usage characteristics of devices and energy-efficient resource allocation policies are investigated [16]. Abbasi et al. developed the *TACOMA* system, which decreases heat distribution by server provisioning [17]. *TACOMA* uses computational fluid dynamics or CFD to formulate the heat re-circulation model. *TACOMA* chooses active servers in a way to minimize total energy. Patel et al. built an online measurement and control techniques to enhance energy efficiency [18,19]. Supply and return heat index (i.e., RHI and SHI) characterizes the energy efficiency of computer room air conditioning (CRAC) in a data center. The measurement tool can pinpoint hot spots and non-uniformity in heat distribution. Given RHI and SHI readings, a scheduler dynamically distributes workload to improve thermal efficiency. Aforementioned techniques take into consideration of thermal efficiency in order to cut back energy consumption; however, these schemes are inadequate for mitigating adverse thermal impacts at a high level in data centers. Our **TAH** is distinctly different from the aforementioned approaches, because **TAH** conserves cooling cost of Hadoop clusters by taking into account multiple thermal-affecting factors.

2.2. Thermal management

An increasing number of thermal management approaches can be found in the literature. Guo et al. proposed a thermal storage management in data centers by considering geographical load balancing, opportunistic scheduling of delay-tolerant workloads [20]. Shuja et al. designed a data-center-wide energy-efficient resource scheduling framework, which schedules resources according to the current workload of a data center. Shuja et al. implemented a resource scheduling algorithm minimizing a subset of resources to service computing workload [21]. Guo et al. proposed a dynamic flow scheduling scheme, which achieves excellent power efficiency in DCNs with an improved QoS [22].

Recently studies show that heat re-circulation has significant impacts on cooling cost. For example, Guo et al. investigated a joint inter- and intra-datacenter workload management strategy to cut the electricity cost of geographically distributed datacenters [23]. And Moore et al. proposed a power budgeting policy called *MinHR*, the goal of which is to minimize the total amount of heat recirculated in a data center before returning to CRAC. *MinHR* applies the hear re-circulation factor or HRF as a parameter modeling the heat re-circulation of one chassis on all other servers. When it comes to homogeneous servers, HRF of each server is equal to the sum of the correspond row of a heat re-circulation matrix. *MinHR* is a two-phase heuristic solution. First, MinHR ranks chassis based on the ratio between each individual chassis HRF to the sum of all HRFs. Then, MinHR assigns tasks to a chassis with the lowest HRF ratio. The existing thermal management approaches facilitate the improvement of the energy efficiency of general-purpose data-center platforms. Unfortunately, none of approaches of the above techniques are specifically tailored for modern big-data platforms like Hadoop, Storm, and Spark clusters. Unlike the aforementioned thermal management approaches that are designed for general data center, our solution is focused on thermal management in Hadoop clusters.

2.3. Thermal-aware scheduling

Much attention has been paid to a wide range of energy-saving techniques in the realm of thermal-aware scheduling. A handful of algorithms were introduced to deal with various workload types and programming models. For instance, Tang et al. proposed the *XInt-GA* and *X-Int-SQP* algorithms to minimize peak inlet temperatures to reduce cooling power [24]. *XInt-GA* is a genetic algorithm, whereas *XInt-SQP* is a sequential quadratic approach. *XInt-GA* and *X-Int-SQP* are capable of minimizing peak inlet temperatures through task assignments. Sharma et al. developed the Inverse-temperature algorithm [25], which heuristically solves temperature imbalances by redistributing workload inversely proportional to chassis outlet temperatures. Thermal load balancing is achieved by providing cooling inlet air for each rack below a redline temperature, maintaining uniformity in inlet temperatures. The load balancing mechanism dynamically responds to thermal emergencies that cause uneven temperatures.

All the aforementioned algorithms are capable of balancing temperatures in data centers. Unfortunately, the existing thermal-efficient solutions are inadequate boost thermal efficiency of Hadoop clusters. Our proposed *TAH* addresses this issue by the virtue of a dynamic thermal-aware scheduler dedicated to MapReduce computing frameworks like Hadoop. Our *TAH* minimizes the peak inlet temperature by incorporating the heat circulation matrix into the Hadoop framework. Table 1 summarizes the major differences between TAH and the existing schemes found in the literature. Let us compare the discrepancy between TAH and the alternative solutions from the perspectives of thermal awareness, Hadoop computing platform, load balancing, and scheduling, respectively.

3. TAH: Thermal-Aware Scheduler

The Hadoop scheduling mechanism allows multiple MapReduce jobs to share computing resources in a cluster. Hadoop scheduling policies act as plugins, among which the Hadoop system can readily switch the schedulers. By default, Hadoop manages three schedulers, namely, the FIFO, Fair and Capacity schedulers.

Table 1
Comparisons between TAH and the other existing schemes.

Comparisons among the most closely related studies on thermal-aware scheduling techniques				
Schemes	Thermal awareness	Hadoop computing platform	Load balancing	Scheduling
TACOMA [17]	✓	✗	✓	✗
Power-Aware Scheduling [12]	✗	✓	✓	✗
Thermo-fluids Provisioning [19]	✓	✗	✗	✓
TAWP [26]	✓	✗	✗	✗
DCEERS [21]	✗	✗	✓	✓
TAH	✓	✓	✓	✓

All these schedulers are structured to optimally handle resources without addressing the thermal management issue. Our *TAH* – a thermal-aware scheduler – is incorporated into the Hadoop scheduling mechanism to improve thermal efficiency of Hadoop clusters. In this section, we first shed light on the scheduling mechanism (see Section 3.1, into which the *TAH* is integrated. Next, in Section 3.2, we present the thermal model, a centerpiece in *TAH*. Last, we detail the implementation of *TAH* in Section 3.3.

3.1. The Hadoop scheduling mechanism

Hadoop jobs are divided into tasks based on split size of input data. The Hadoop system manages tasks stored in a task queue. The *Job Tracker* assigns tasks from the task queue to one of the task trackers in a Hadoop cluster; the candidate task trackers should be healthy and free to perform the assigned tasks. The *HeartBeat* mechanism is a communication interface between Job Tracker and Task Trackers. The scheduling mechanism performs the following procedure.

Step 1. After a client submits a job, the job is placed into an internal queue, from which the job scheduler picks jobs to be executed (see Fig. 1). During the course of job initialization, the job is partitioned into a list of small tasks stored in a task queue. Recall that the large input data is divided into small splits, each of which is handled by a task in the task queue.

Step 2. After the splits is originated, the job client launches map tasks for the splits. Each task is assigned a unique identifier (i.e., task ID)

Step 3. The *Task Trackers* runs a simple loop that periodically sends heartbeats to the *Job Tracker*. Note that the heartbeat period can be customized by system administrators. Heartbeat messages offers a means of communication between the Task Trackers and the Job Tracker. The key fields of the heartbeat data structure contains maximum numbers of mappers and reducers, total amount of physical and virtual memory, CPU frequency and time, response ID, Task Tracker's health state. The heartbeat message is periodically delivered from the Task Tracker to the Job Tracker. In case that the Job Tracker does not receive a heartbeat message from the Task Tracker for an interval of time, then the corresponding node is labeled as unhealthy. Tasks assigned to the unhealthy node will be passed to other healthy Task Trackers.

Step 4. In acknowledgment of heartbeat, *Job Tracker* further assigns map and reduce tasks based on the number of free slots available. The Task Trackers usually have two map and reduce slots, meaning that each Task Tracker only run two mappers or two reducers at a time. Generally, the number of Map/Reduce slots in a Task Tracker are configured depending on the number of computing cores, each of which is in charge of one task. If a node has at least one Map slot available, then the node will accommodate one mapper; otherwise, a Reduce task will be executed by the task Tracker.

Step 5. Before processing a heartbeat, *Job Tracker* takes the following steps to invoke task-tracker profiling on the basis of its previous heartbeat information.

- **Processing HeartBeat**: Job Tracker, receiving a HeartBeat from a Task Tracker with all the above mentioned data, handles data only if the data is delivered by an allowed Task Tracker. If a HeartBeat is duplicated, then Job Tracker will ignore the heartbeat. Otherwise, the HeartBeat is processed; tasks are examined prior to their execution.
- **Obtaining a task list:** When the Task Tracker is ready to run a task, Job Tracker obtains a list of tasks that are either a setup or clean-up task.
- **Choosing a task from the list:** The scheduler iterates through the set-up task list, before choosing a runnable task; non-runnable tasks are marked as failed ones. The scheduling mechanism removes the chosen task from the list after a task is scheduled, killed, completed, or failed on this Task Tracker before.
- **Checking flaky Task Trackers:** After retrieving a map or reduce task, the scheduling mechanism checks if an excessive number of tasks have failed on this Task Tracker. If the case is confirmed, then the chosen tasks will not be assigned to the Task Tracker; otherwise, the tasks are marked as schedulable.
- **Assigning the created Mappers/Reducers:** The scheduling mechanism calculates a Map-Reduce load factor by counting the Task Tracker's total Map and Reduce slots as well as the total Map and Reduce slots across the pool. A newly selected task from the task queue is supplied with resources. The chosen tasks are executed in the order of failed tasks, non-running tasks, speculative tasks, no-location-information tasks.
- **Launching the task:** After the task is assigned, the mechanism launches the task in the Task Tracker while kicking off a timer to keep track of the task's progress. If the Task Tracker does not respond to this task for a long time period, then the task will be pinpointed as a failed task.

Step 6. *Job Tracker* further checks for any job to be killed, cleaned up, or tasks that needs to be committed. In the HeartBeat response, the mechanism checks if any restart information should be included prior to sending the signal to the Task Tracker.

3.2. Thermal model

In this section, we present our simple yet effective thermal model. We also show how to obtain the thermal related information governed by our thermal model.

3.2.1. The thermal model

The objective of *TAH* thermal model is to keep inlet temperature T_{in}^{i} below redline temperature T_{red}, so that our *TAH* scheduler can make thermal-friendly scheduling decisions by using this critical criterion(see Eqs. (1), (2) and Section 3.3). To characterize the thermal impacts in Hadoop clusters, we propose a thermal model as well as the building blocks necessary for the model. We also present the following assumptions and the notation used in the model.

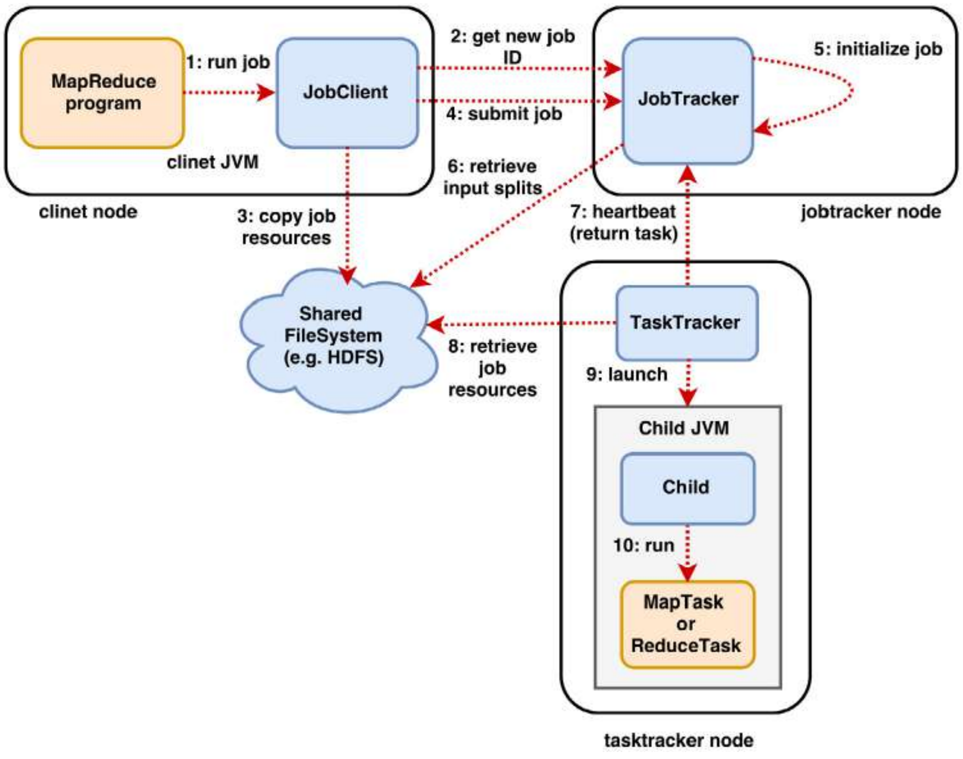


Fig. 1. An anatomy of the Hadoop scheduling mechanism.

Table 2
Model notation.

Variables	Description
i	Number of Server Node
Q	Heat generated (J)
p	Density of air (kg/m^3)
f	Flow rate (m^3/s)
T_{sup}	Air temperature as supplied from cooling unit
Anxn	Heat Re-circulation/ cross-interference matrix
T_{out}	Outlet Temperature (c)
T_{in}	Inlet Temperature (°C)
T	Change/Rise in temperature (°C)
T_{red}	Red Line/Threshold temperature
P_c	Computational Power
R	Ratio of distance
$d_{i,j}$	Cross Interference matrix
d_i	Distance of the server from AC vent (m)
d	Height of room (m)
$T_{initial}$	Room Initial Temperature (°C)
K_{disk}	Constant temperature rise by disk
$T_{in}^{possible}$	Possible Inlet temperature if job assigned
C_{util}	CPU utilization
D_{util}	Disk Utilization
P_{AC}	Cooling cost/Power consumption by AC unit
COP	Coefficient of performance

(1) An initial temperature is always consistent with the room temperature in a data center.
(2) The air flow is static in all parts of the data center.
(3) Supplied temperature strength is linearly proportional to the distance from the vent.
(4) disks are assumed to be in two states (i.e., either active or idle).
(5) Active disks contribute to a constant rise in outlet temperature.

A temperature simulator class gathers CPU and disk utilization information from the hash structure maintained by Job Trackers. For a server, the outlet temperature is sum of temperatures $T_{initial}$ and T_{rise}. Temperature T_{rise} is linearly related to the CPU Utilization of the server (see Eq. (2) where K_{disk} is a constant contribution of an active disk to outlet temperature T_{out}). (See Table 2.)

The inlet temperature of a server node (a.k.a., Task Tracker) is affected by CRAC temperature T_{sup} as well as the outlet temperature of the server itself and the contribution of heat re-circulation effect from other server nodes. Eq. (1) shows the actual representation of Inlet Temperature of a server at any given point of time. T_{in} depends on T_s and a vector, which models the exact strength of T_s at a given height [27].

$$R = 1 - \frac{d^i}{d}$$

$$T_{in}{}^{i} = T_{initial}^{i} - RT_{sup}{}^{i} + AT_{out} \tag{1}$$

Eq. (1) can be reorganized to obtain outlet temperatures.

R in Eq. (1) represents the height factor of server from CRAC. We calculate the height by using ratio of server's height from CRAC to ceiling height from CRAC. This implies the fact that the higher the server's placement from CRAC which is at floor, the less cooling effect received from a cooling device. In Eq. (1) A is a heat re-circulation or cross-interference vector, which sums up the temperature rise of a server node. Such a summation is reasonable, because the node's neighbors sitting at placement outlet temperature.

In our study, heat re-circulation vector or matrix is calculated by using a thorough evaluation of a data center's physical layout with the CFD tools (see, for example, [28] and [26]). Fig. 2 depicts the cross interference of the heat re-circulation effects among a group of neighboring nodes. In Fig. 2, A is a nxn matrix where n is total number of cluster nodes in the cluster. Each element $\alpha_{i,j}$ from the matrix represents the fraction contribution of heat from server $node_i$'s outlet to $node_j$'s inlet temperature. The outlet temperature of a node (e.g., node i) affects the inlet temperatures

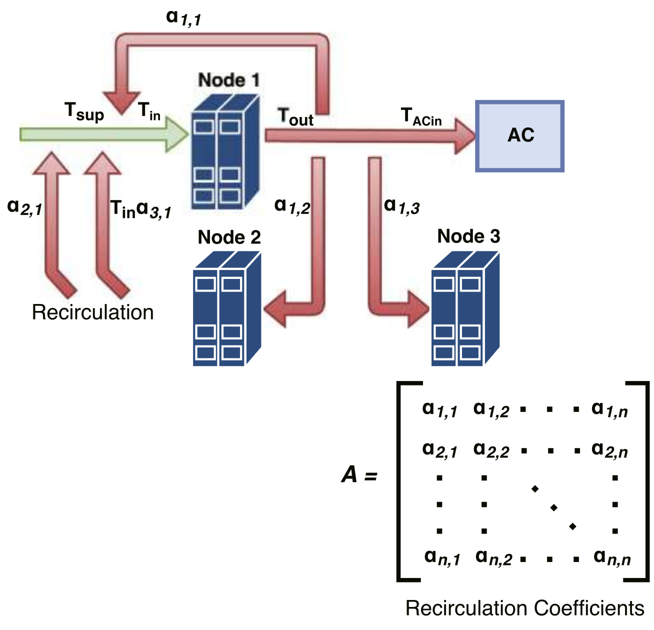


Fig. 2. Cross-interference between neighbor servers.

of itself (e.g., factor $\alpha_{i,i}$) as well as its neighboring nodes (e.g., factor $\alpha_{i,j}$, node j is node i's neighbor).

$$T_{rise}{}^{i} = K_{disk}{}^{i} + aC_{util}{}^{i} + b$$
$$T_{out}{}^{i} = T^{i}_{initial} + T_{rise} \tag{2}$$

The main factor driving temperature T_{out} up is the heat transfer in the system from the inlet side to the outlet side. Computing components such as CPU chips and disks heat up during the course of heavy workloads. Air heat transfer leads to an increased outlet temperature T_{out}.

The overarching goal of our proposed *TAH* is to keep inlet temperature T^{i}_{in} below redline temperature T_{red}. Such a goal becomes a critical criterion on making thermal-friendly scheduling decisions; thus, the thermal-aware scheduler to keep inlet temperature below the redline. In addition, distributing tasks by *TAH* imposes no requirement of reducing CRAC temperature T_{sup}, which in turn conserves massive energy cost. The objective function of *TAH* can be expressed as

$$\begin{aligned} &\text{Minimize: } T^{Avg}_{in} = \frac{\sum_{i=1}^{n} T^{i}_{in}}{n}, \\ &\qquad\text{subject to } T^{i}_{in} \leq T_{red}, \end{aligned} \tag{3}$$

where T^{Avg}_{in} is the average inlet temperature across all the computing nodes.

3.2.2. HeartBeat mechanism modification

There are two options to implement a channel to transfer thermal information among Task Trackers and Job Tracker in Hadoop. The first choice is to develop the communication channel from ground up. The second strategy is to make use of the existing communication mechanism. We choose to take the second approach by extending the HeartBeat mechanism to retrieve thermal information acquired from task trackers. The intuitive advantage of applying the HeartBeat mechanism to obtain thermal data is two-fold. First, the HeartBeat channel allows thermal data to be transferred between Task Trackers and the Job Tracker in a timely manner. Second, the implementation of the Heartbeat mechanism is light weight nature. There are two Heartbeat interfaces in Hadoop. The first interface connects Job Trackers and Task Trackers, whereas another interface is sitting between DataNode and NameNode. In our study, we used the MapReduce interface. In our *TAH*, a Task Tracker maintains an instance of a class called ResourceStatus, which is composed of all resource information pertinent to the task trackers resources (e.g., virtual memory, physical memory, and the number of map/reduce slots).

We extend the ResourceStatus class by introducing a few new methods keeping track of CPU Utilization, Disk Utilization, CPU core temperatures, and disk temperature. Since Hadoop is a Java based framework running on *JVM*, Hadoop has no direct access to any machine's resources. To solve this problem, we employ the Linux resource calculator plugins, /proc/stat, which provides the CPU and disk status information. To monitoring disk temperatures in *TAH*, we take a full advantage of *hddtemp*, a Linux utility program to extract disk temperatures.

Job Tracker maintains a HashMap of Task Tracker name to last TaskTrackerStatus received from that Task Tracker. By doing so, the Job Tracker is able to identify the health Task Trackers. A benefit of leveraging the heartbeat to communicate data between Task Trackers and a Job Tracker is to make use of an existing lightweight mechanism that is very well suited for real-time data propagation. We also recognize that the minimum interval between two heartbeat message is minimum three seconds, which serves the purpose of real-time monitoring at reasonably low cost.

3.3. Architectural framework and implementation

To incorporate thermal awareness into Hadoop schedulers, we implement our *TAH* by modifying the source code of the existing Hadoop system. Specifically, we extend the HeartBeat mechanism by offering an array of new methods calculating CPU utilization, Disk utilization, CPU core temperatures, and disk temperatures within the ResourceStatus class instance, which is maintained by Task Trackers. In order to leverage these thermal relevant information, we develop a thermal estimator deployed to extract heartbeat messages and to project peak inlet temperatures using the thermal model. Moreover, the *TAH*-Scheduler makes decisions on whether or not to assign jobs to corresponding nodes to maintain the balance in Hadoop clusters from the perspective of thermal awareness. In this section, we discuss the design and implementation issues of *TAH*. Fig. 3 depicts the framework of *TAH*. In addition to modifying the Hadoop's default scheduling scheme, we implement a set of thermal information processing modules including a thermal simulator and the *TAH*-scheduler. During the implementation phase, we have the following reasonable assumptions.

- Each Task Tracker works on one task at a time.
- All map tasks in a job are identical in terms of their resource requirement, because all the tasks process the same operations.
- Blocks processed by all the map tasks share the same size.
- Each task is assigned to utmost one Task Tracker.
- Above constraints implies that temperature increase per map task is also be same across all the servers.
- Heat re-circulation effect is more profound to servers mounted on the same rack.

3.3.1. HeartBeat

Task Trackers periodically send HeartBeat messages containing CPU and disk temperatures and utilization along with the other HeartBeat fields. The HeartBeat message interval sets the accuracy of the real time information we have on the Job Tracker.

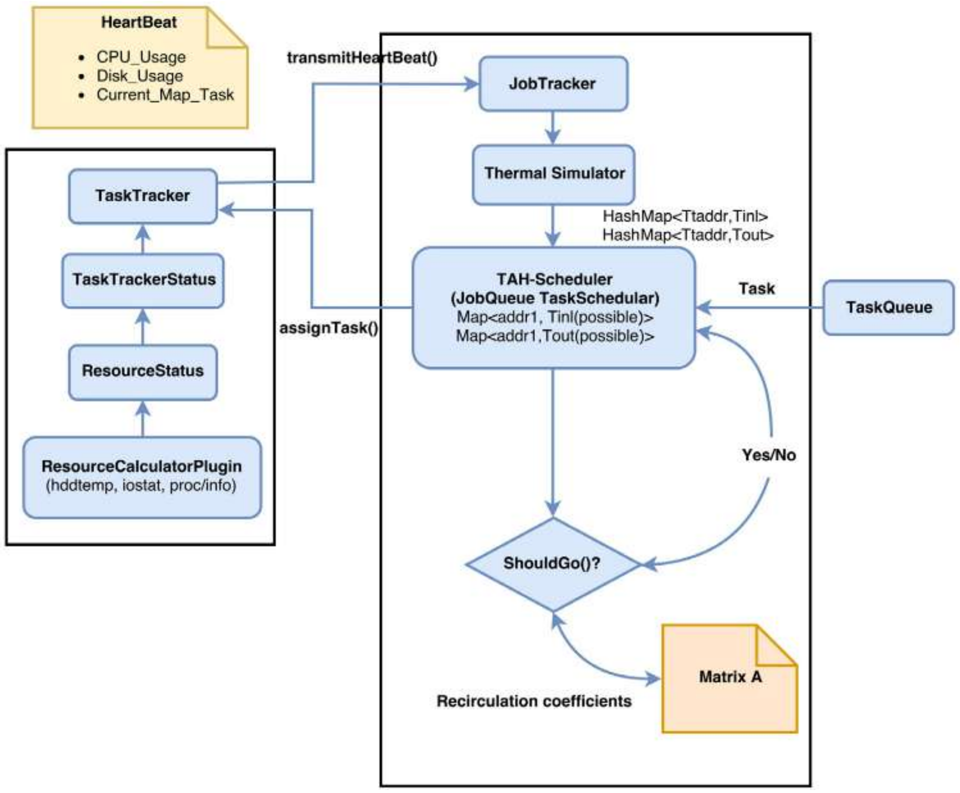


Fig. 3. TAH design implementation.

3.3.2. *Task tracker*

In the initial phase, each Task Tracker – having no assigned task – is available to accept a task for execution. Each time a Task Tracker requests for a task by updating its availability through a heartbeat message; the Task Tracker also registers itself in the Thermal Simulator. In other words, if a Task Tracker has just sent a heartbeat requesting for a task, it will register itself in the Thermal Simulator.

3.3.3. *Thermal simulator*

The Thermal Simulator is in charge of extracting heartbeat messages and estimating the peak inlet temperature using the thermal model. The *thermal simulator* (see the following 11 steps) is invoked by Job Tracker to extracts the Task Tracker status upon the arrival of the heartbeat message.

(1) The simulator extracts a heartbeat message and creates a map structure to keep the heartbeat required fields (CPU_{util} and $Disk_{util}$) with each information being mapped to the Task Trackers address i and their field values.
(2) Now on wards each time i task tracker sends a heartbeat, the simulator updates its map structure to keep information in a real-time manner.
(3) The simulator takes the three steps below to estimate temperatures using the thermal model:
    - The CPU usage and disk usage are applied to calculate the outlet temperature of server i and maintains another map structure for storing the outlet and inlet temperatures for Task Tracker.
    - The simulator makes use of the model (see Eq. (2) in Section 3.2) to calculate the outlet Temperature. We adopt a linear outlet temperature model, where CPU and disks are in either the active or idle mode.
    - From outlet temperatures, the simulator derives inlet temperatures and those of neighboring task trackers using Eq. (1). T_{sup} is a constant temperature and A is the cross interference matrix offering factor $\alpha_{i,j}$ (i.e., an indicator of server i's contribution to the inlet temperature of i's neighbor Task Trackers j). If servers i and j are far from each other, then factor $\alpha_{i,j}$'s value becomes fairly small. The most prominent heat re-circulation affect could be noticed among servers mounted in the same rack.
(4) The scheduler iterates through the task list, where a task is chosen. A task may be removed from the list if it is completed or killed.
(5) The simulator calls a boolean method named *shouldGo* to determine whether the scheduler should assign task to the task tracker seeking a task. The method here uses difference between the inlet and outlet temperatures. Let us consider an example where task tracker i has requested for task (see Eq. (4)).

$$\triangle T^i = {T_{out}}^i - {T_{in}}^i \tag{4}$$

(6) Now a method calculates the temperature rise per map task by using Eq. (5). This expression relies on an earlier assumption that all map tasks are similar, meaning that all the tasks have equal contribution to the outlet temperature of server i.

$$\triangle T^{Avg} = \frac{\triangle T}{map_i}, \tag{5}$$

where $\triangle T^{Avg}$ is the temperature increase per mapper.
(7) We employ a method to obtain and maintain estimated outlet temperatures if a job is assumed to be assigned to that tracker $T_{out}^{possible}$.

(8) Given server i, we introduce a method calculating the average of $d_{i,j}$ where $j\varepsilon[1\dots n]$. Here n is number of Task Trackers in cluster as in Eq. (6).

$$\alpha_i^{Avg} = \frac{\sum_{j=1}^{n} \alpha_{i,j}}{n}, \tag{6}$$

where α_i^{Avg} is the average factor for server i.

(9) From the given matrix, our scheduler then looks for neighboring nodes for which the cross-interference matrix value is greater than the average for that server (see Eq. (6)). Once known how the neighbor servers could be critically affected by the re-circulation effect of heat, the scheduler then applies $T_{out}^{possible}$ to calculate the inlet temperatures of all its neighbor nodes; the temperatures are kept as $T_{in}^{possible}$.

(10) Now it is time for the scheduler to make a decision on whether or not it is a wise idea to assign a job to server i. The criteria stated in Eq. (7) is for server i and all its neighbors, estimated temperature $T_{in}^{possible}$ is lower than redline temperature T_{red}.

$$\text{criteria} = T_{in}^{possible} < T_{red} \tag{7}$$

(11) If server i does not meet the criteria in the previous step, no task is assigned to the server; rather, the scheduler waits for the next server's heartbeat message to evaluate future candidate servers.

In summary, tasks are scheduled to a server that has the least thermal impact on the server's neighbors. The system parameters considered by the scheduler include CPU/disk usage and the number of running tasks on the server. Technically speaking, the scheduler incorporates the heat re-circulation effect by taking into account a data center's physical layout, which is in form of a cross-interference matrix. We also take into consideration of the factor of the physical placement of servers (i.e., rack placement and the proximity to a cooling system). These factors affect inlet temperatures over the course of duration. Our scheduler offers thermal management to govern the data center with little adverse impact on system performance.

4. Experimental results

We evaluate the energy efficiency of a ten-node Hadoop cluster, where the prototype of *TAH* is implemented. Similarly, we scale down the size of our testing data sets to match the capacity of the cluster. In this section, we first show system and workload parameters (see Section 4.1). Then, we discuss the thermal and cooling cost of the tested Hadoop cluster managed by *TAH* (see Section 4.5.

4.1. Setup and benchmarks

We build a Hadoop cluster as a testbed using the off-the-shelf commodity hardware.

Hardware. The Hadoop cluster is composed of a NameNode and ten DataNodes. Tasks running on each DataNode is managed by a task tracker. Table 3 shows the hardware configuration of the computing nodes (i.e., NameNode and DataNode) in the Hadoop cluster. The switch employed to interconnect all the nodes is TP-Link's TL-SG1024D the bandwidth of which is 10,100 Mbps.

Software. We install the Linux Ubuntu 10.04 (i.e., Linux kernel 2.6.32–25.45-generic) operating system on all the nodes. We implement TAH as an extension to the existing Hadoop system, where Java 1.6 is running. The nodes have password free accesses to launch tasks and to exchange of intermediate data. We attach temperature sensors to each node; we apply the hddtemp utility program to measure CPU and disk temperatures.

Cluster Size and Data Set. To evaluate the performance our *TAH*, we vary the cluster size and data sizes to resemble various system and workload conditions. More specifically, experiments are conducted with a cluster size of 5, 10, and 12, respectively. The data size is configured to 5 GB, 10 GB, and 20 GB, respectively. All the nodes follow the default map/reduce slot settings, block size, and input split size. The replication factor is, by default, set to three in all the experiments.

Benchmarks. We evaluate the schedulers by running the popular Hadoop Benchmarks chosen to stress test a cluster with a wide range of CPU and disk workload conditions. The benchmarks tested in this study include *WordCount*, *Distributed Grep*, *PI estimation*, and *Tera Sort*.

4.2. Temperature reduction

We start the experiments by focusing on node temperature reduction; we compare our *TAH* with the existing Hadoop FIFO scheduler in terms of load distribution among the nodes. Fig. 4(a) and (b) show the number of *WordCount* task distribution among the five nodes processing data of 10 GB and 15 GB, respectively.

The results indicate that the existing Hadoop scheduler distributes tasks without concerning server's rack location in data centers. In other words, the default Hadoop scheduler randomly assign map tasks to the data nodes. Consequently, the hadoop scheduler inevitably increases peak inlet temperature non-uniformly, which leads to a pressing need to repeatedly lower supply temperature T_{sup}.

In contrast, *TAH* evidently dispatches tasks by being aware of server locations to improve thermal efficiency. *TAH* strives to maximize utilization of nodes that are located close to a cooling system (e.g., close to floor), thereby cooling down the hotspot in a Hadoop cluster. For example, *TAH* pushes workload of node 5 higher than that of the other nodes, because node 5's ambient temperature is lower than those of the others. *TAH* maintains a light load on node 1, which is the least thermal-friendly node in the system. It is noteworthy that Node 1, located far away from CRAC, contributes the most heat re-circulation effect. Furthermore, node 1's inlet temperature is the highest among all the nodes during the course of data processing. *TAH* judiciously cools down node 1 by reducing the number of running tasks.

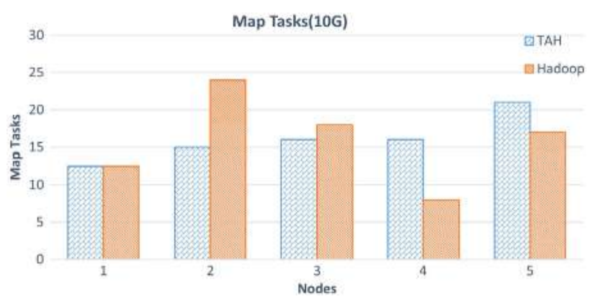
Comparing Fig. 4(a) and (b), we observe that regardless or the applied scheduler, increasing the data size from 10 G to 15 G pushes up the number of assigned tasks on all the nodes. For example, in the case of the existing Hadoop the number of assigned tasks on node 4 is increased from 9 to 11; in the *TAH* case, the number of tasks handled by node 4 is changed from 16 to 23. *TAH* makes node 4 share increased heavy load, because node 4 along with node 5 contribute less adverse impacts on heat re-circulation than the other nodes.

Fig. 5(a) and (b) depict the inlet temperature distributions when the Hadoop and *TAH* scheduler is employed, respectively. We observe from the figures that compared with the Hadoop scheduler, *TAH* significantly reduces the nodes' peak inlet temperatures by 1.9 °C. The results confirm that *TAH* offers prominent reduction in peak inlet temperatures of the nodes running WordCount tasks. In turn, such inlet temperature reductions help in keeping the supply temperature at a relatively high level to conserve cooling energy consumption.

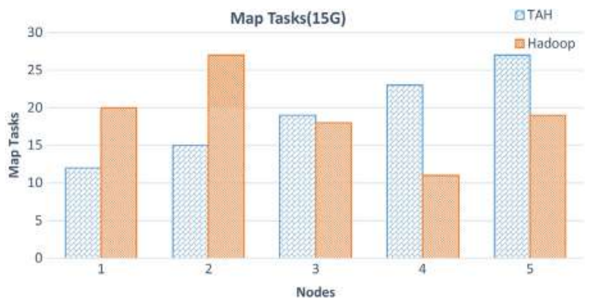
After comparing Fig. 5(a) with Fig. 5(b), we conclude that the amount of processed data affect peak inlet temperatures. This trend is applicable to the two evaluated schedulers. For instance, when we increase the data size from 10 GB to 15 GB, node 2's peak temperature moves from 29.6 °C to 33.2 °C in the Hadoop case and from 29.0 °C to 31.4 °C. An intriguing observation is that a large amount input data allows the *WordCount* benchmark take further

Table 3
Server specifications.

Node type	Number of cores	RAM	Storage
HP proliant ml110	Intel X3430 4 cores@2.8 GHz	2 GB ECC DDR3 SDRAM	143 GB
Dell Optiplex360	Intel E7400 4 cores@2.8 GHz	2 GB Non-ECC DDR2 SDRAM	143 GB

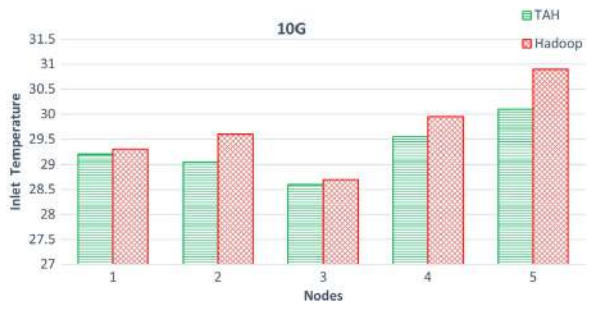


(a) Number of map tasks vs node (10 GB).

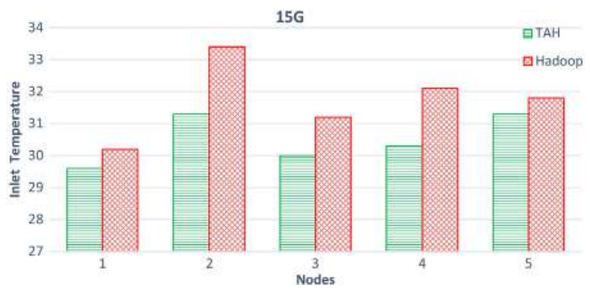


(b) Number of map tasks vs node (15 GB).

Fig. 4. Map task distribution for wordCount.

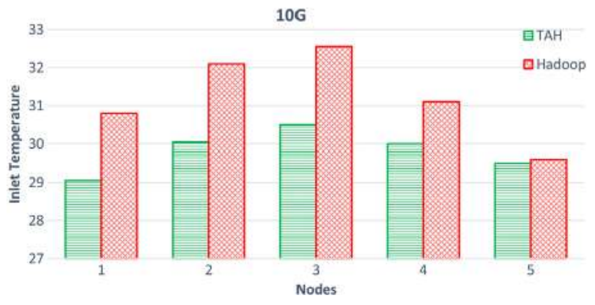


(a) Peak inlet temperature vs node (10 GB).

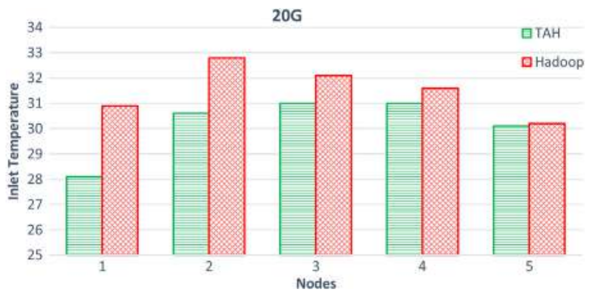


(b) Peak inlet temperature vs node (15 GB).

Fig. 5. Peak inlet temperature for wordCount.



(a) Peak inlet temperature vs node (10 GB).



(b) Peak inlet temperature vs node (20 GB).

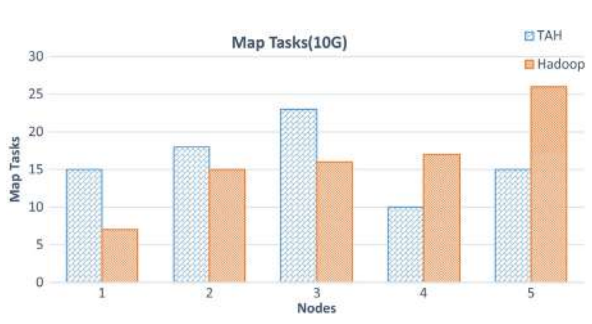
Fig. 6. Peak inlet temperature for Grep.

thermal benefit from *TAH*. The experimental results suggest that data-intensive applications like *WordCount* can take full thermal advantage from our *TAH*.

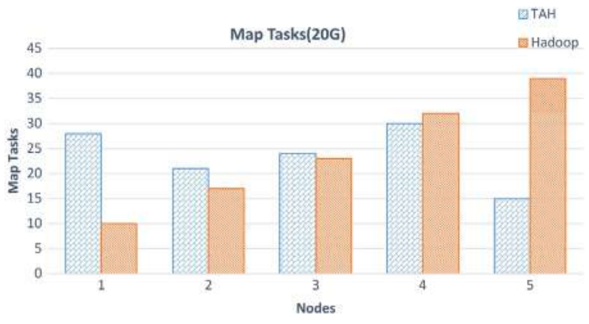
Now we are positioned to conduct a set of experiments to evaluate the performance of *TAH* driven by the *Grep* benchmark in terms of load distribution and peak-inlet-temperature reduction among the nodes. Fig. 6(a) and 6(b) shows that *TAH* reduces the Peak inlet temperatures of the nodes running Grep by 1.5 °C. The peak temperature reductions are affected by Grep's heavy disk usage (i.e., Grep is an I/O intensive job) as well as the thermal-aware task assignments. Fig. 7(a) and 7(b) displays that the nodes overly utilized in the Hadoop scheduler are underutilized in the *TAH* case, where the hotspot node's peak temperature along with those of the other nodes are noticeably cut down.

The *TeraSort* application [29] is capable of sorting a massive amount of data (e.g., 1 TB) in a fast manner. The *TeraSort* benchmark facilitate an integrated testing of the HDFS and MapReduce layers in Hadoop clusters. The TeraSort benchmark suite offers added benefits allowing us to measure thermal behavior of our Hadoop cluster governed by *TAH*.

In this group of experiments, we make use of the *TeraSort* benchmark to compare capability to reduce peak inlet temperature and load-distribution efficiency of our *TAH* with default Hadoop's. Fig. 8(a) and (b) illustrate that *TAH* lowers the peak inlet

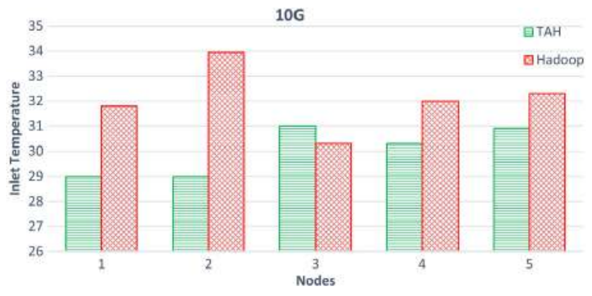


(a) Number of map tasks vs node (10 GB).

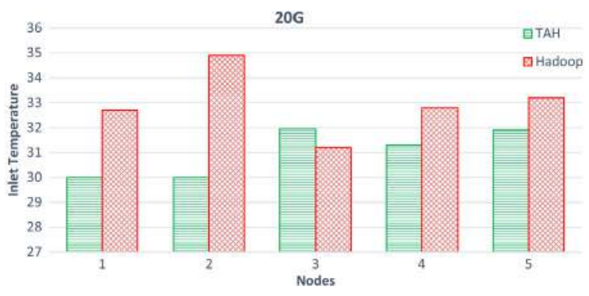


(b) Number of map tasks vs node (20 GB).

Fig. 7. Map task distribution for Grep.

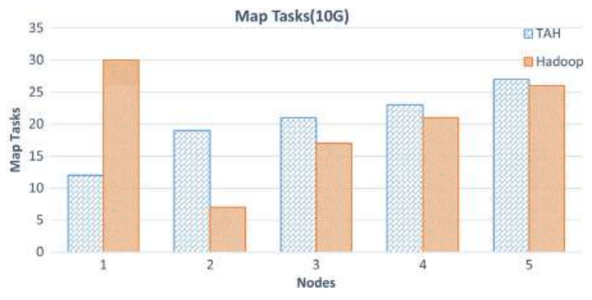


(a) Peak inlet temperature vs node (10 GB).

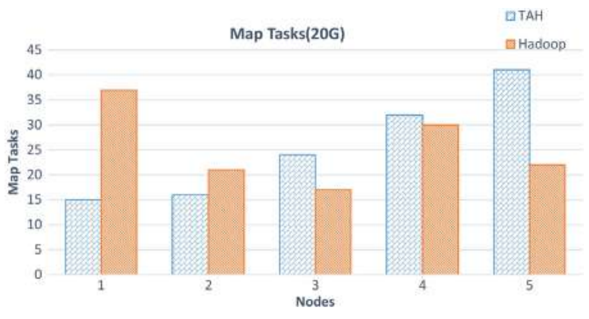


(b) Peak inlet temperature vs node (20 GB).

Fig. 8. Peak inlet temperature for Tera Sort.



(a) Number of map tasks vs node (10 GB).



(b) Number of map tasks vs node (20 GB).

Fig. 9. Map task distribution for Tera Sort.

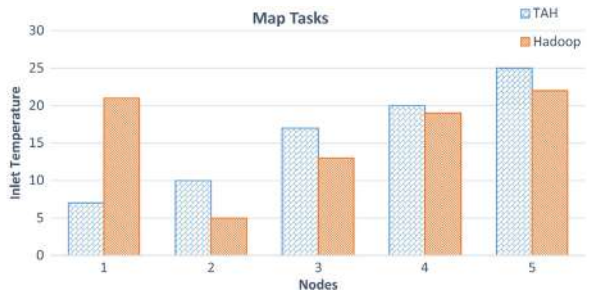
temperatures of the nodes running TeraSort. More specifically, *TAH* reduces the peak inlet temperatures of the cluster managed by the existing Hadoop scheduler by an average of 2.3 °C and 2.2 °C for the input data size of 10 GB and 20 GB, respectively. In the 10 GB case, the maximal and minimal peak-temperature reductions are 4.9 °C and 0.8 °C; whereas the maximal and minimal peak-temperature reductions of the 20 GB case are 4.8 °C and 0.7 °C. This thermal performance improvement with the perspective of TeraSort is consistent with those of the WordCount and Grep benchmarks.

Again, we attribute the peak-temperature reductions offered by *TAH* to task load distributions, which are plotted in Fig. 9(a) and (b). The results show that *TAH* suppresses peak inlet temperatures of hot spots by minimizing the number of running tasks. The task distribution patterns observed in Fig. 9 agree with those of Figs. 4 and 7.

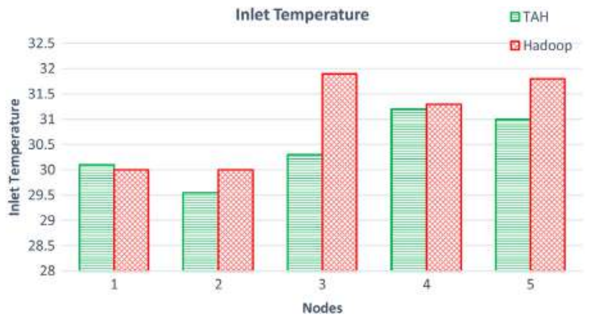
Let us exam the last benchmark - *PI*, which is a Hadoop program computing the PI value. Unlike WordCount and Grep, the *PI* benchmark is a CPU-intensive job. *PI* loads no data from HDFS on the cluster. Fig. 10(a) plots the *PI* task distributions among the data nodes, whereas Fig. 10(b) shows the temperature reductions achieved by *TAH*. Specifically, *TAH* assigns the least number of tasks to node 1 - a hotspot that make the most significant impact to heat re-circulation (see Fig. 10(a)). Consequently, *TAH* lowers the peak inlet temperatures by up to 4.5 °C, with an average of 2.2 °C (see Fig. 10(b)). The task distribution trend and the peak-temperature trends are similar to those of the previous three benchmarks.

4.3. Performance evaluation

We conduct a group of experiments to compare performance of TAH with the original Hadoop using the three machine-learning applications (i.e., Fuzzykmeans, Canopy and Kmeans) implemented in the Apache MAHOUT. We configure these machine learning algorithms using the default arguments. Fig. 11 indicates that the running times of the three MAHOUT algorithms are almost identical in both TAH and Hadoop. For example, for the Kmeans case, TAH slowdowns the original Hadoop's performance by as



(a) Map task distribution vs node.



(b) Peak inlet temperature vs node.

Fig. 10. Pi estimation map distribution and peak inlet temperature reduction.

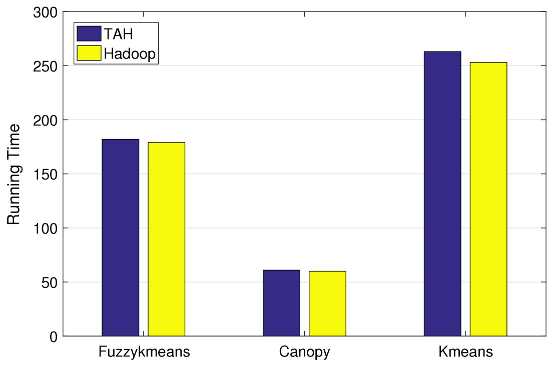


Fig. 11. Performance comparisons between TAH and Hadoop running the Fuzzykmeans, Canopy, and Kmeans applications.

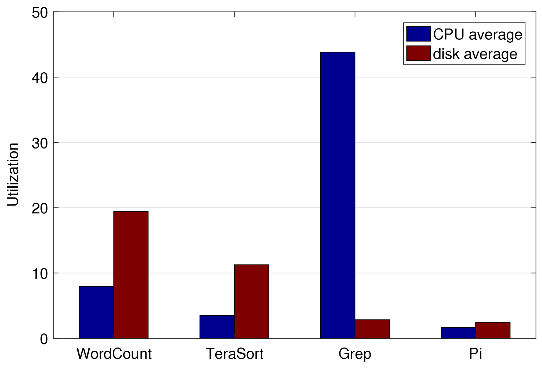


Fig. 13. Performance of TAH's CPU and disk utilization.

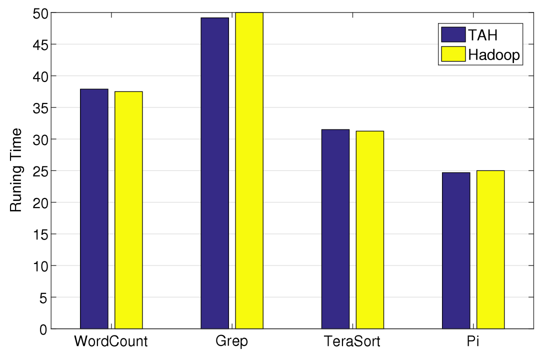


Fig. 12. Performance comparison between TAH and Hadoop. Four tested applications are WordCount, Grep, TeraSort, and Pi.

little as 4.8%. This evidence strongly shows that TAH boots thermal efficiency of Hadoop clusters with marginal performance degradation.

After comparing the performance of TAH with the original Hadoop using the three machine-learning applications, we conduct a set of experiments to illustrate the performance impacts of TAH on Hadoop applications (i.e., WordCount, Grep, TeraSort, and Pi). Again, the system configurations (e.g., number of nodes and input data size) are the same as those of the previous experiments. Fig. 12 indicates that the running times of the four Hadoop benchmarks are almost identical on the clusters governed by TAH and Hadoop's scheduler, respectively. For example, for the WordCount and TeraSort cases, TAH merely slowdowns system performance by 2.5% and 3.2%. The empirical results strongly suggest that TAH improve thermal efficiency of Hadoop clusters without adversely imposing performance degradation.

Now we conduct a set of experiments to evaluate the performance of CPU and disk utilization of *TAH* on Hadoop applications (i.e., WordCount, Grep, TeraSort, and Pi). The number of nodes is set to ten and input data size is 15 GB. Fig. 13 indicates that our TAH can deal with these stress tests without under imposing hefty load on CPU and disk resources. The results show that TAH processes map/reduce tasks in an efficient way. This observation is consistent with those drawn in the previous experiments. In other words, the empirical results strongly suggest that TAH improve thermal efficiency of Hadoop clusters without adversely imposing performance degradation.

4.4. Put it all together

Now we run the benchmarks on a seven-node Hadoop cluster handling a total of 20 GB data. The redline temperature *Tred* is set to 33.5 °C. Fig. 14 illustrates the peak temperatures of the seven nodes running the *TeraSort*, *Grep*, and *PI*.

Again, the results indicate that *TAH* is conducive to reducing peak temperatures the Hadoop cluster managed by the existing scheduler. The peak-temperature reduction offered by *TAH* are summarized in Fig. 15.

TAH's salient feature of lowering peak temperatures is attributed to thermal-aware task scheduling. For example, Fig. 16 confirms that regardless of the benchmarks, *TAH* makes the nodes (e.g., node 7) being closer to the cooling system process more tasks than the other ones (e.g., node 1) whose peak temperatures are more sensitive to the number of assigned tasks. In the cluster

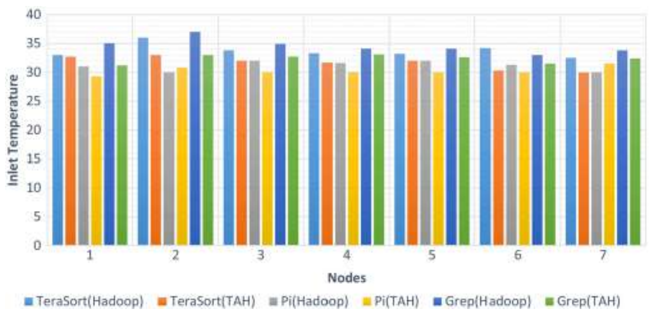


Fig. 14. Peak inlet vs nodes.

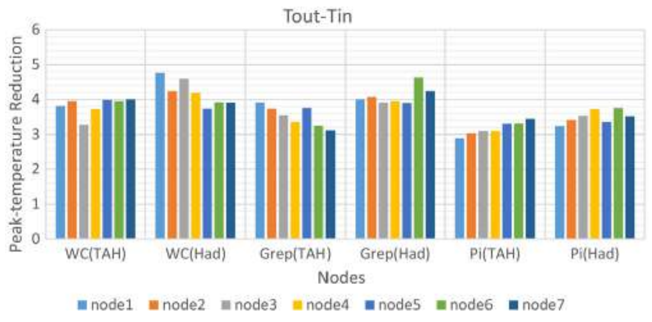


Fig. 15. Peak inlet vs nodes.

governed by *TAH*, the servers on the top are processing fewer tasks because (1) temperature rises in these servers are more significantly affected by heat re-circulation effect and (2) the servers are likely to become a hot spot.

Fig. 16(a) the level of non-uniform distribution of jobs managed by the existing scheduler as well as our thermal-aware *TAH* scheduler. Fig. 16(b) plots the task distributions of all the tested benchmarks. Please note that Node 7 and Node 1 represent the top and bottom servers in the rack, respectively.

When we compare the load distribution trends among all the benchmarks, we immediately pinpoint an outlier - *PI Estimation*. Thus, the number of map tasks of *PI* is noticeably fewer than those of the others, because the other Hadoop applications are data-intensive ones (e.g., the input data size is 20 GB), whereas *PI* is a CPU-intensive job that has no massive amount of input data.

4.5. Cooling cost savings

Now we estimate cooling cost savings offered by *TAH*. To measure cooling cost incurred by power consumption of the cooling system, we apply a validated *COP* model (see details in [24]) to derive the power consumption from the average CPU usage of each data node in the Hadoop cluster. Specifically, this model (see Eq. (8)) represents the cooling power consumption as a function of computing power consumption and coefficient the coefficient *COP*(*Tsup*).

$$P_{AC} = \frac{P_c}{COP(T_{sup})} \tag{8}$$

where coefficient *COP*(*Tsup*) is a function of the air supply temperature *Tsup*.

In our cooling cost estimation, we keep CRAC's temperature T_{sup} at 20 °C. Function *COP*(*Tsup*), adopted to calculate cooling power, is expressed as

$$COP(T_{sup}) = (0.0068T_{sup}^2 + 0.0008T_{sup} + 0.458) \tag{9}$$

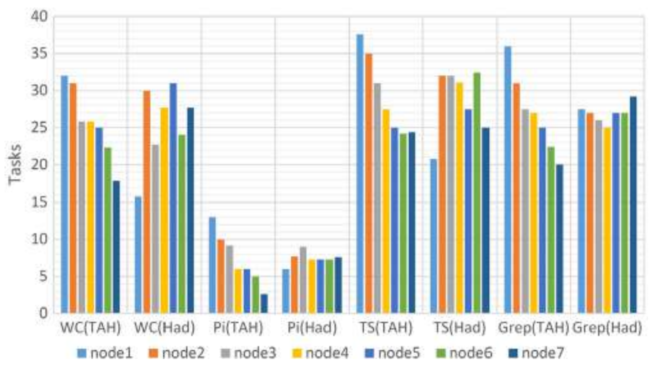
We employ function P_c (see Eq. (10) below) to estimate power consumed by the Hadoop cluster. P_c is a function of idle power P_{idle}, active power P_{active}, and CPU utilization C_{util}. Thus, we have

$$P_c = P_{idle} + P_{active} \times C_{util}, \tag{10}$$

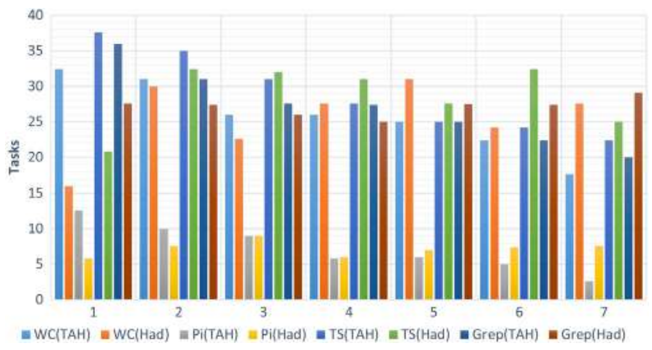
where we set idle power P_{idle} and active power P_{active} to 120 and 50 watts, respectively. We choose these power values to resemble real-world servers like Dell PowerEdge M610. When there is no assigned task running on a data node, the power consumption equals to P_{idle}. If CPU is busy, power consumption P_c is linearly dependent of CPU utilization C_{util}.

The overall power consumption is a summation of computing power and cooling power. Recall that Fig. 15 shows the differences between outlet temperature T_{out} and inlet temperature T_{in}. It is noteworthy to mention that a large difference between temperaturesT_{out} and T_{in} leads to high cooling cost [24]. Table 4 summarizes the cooling cost of the seven-node cluster running all the four Hadoop benchmarks.

We apply the empirical data reported in Table 4 to project cooling cost of a large-scale data center housing a total of 50,000 computing nodes processing Hadoop applications. Fig. 17(a) compares the cooling cost of the data center managed by the existing scheduler and that governed by *TAH*. We observe that *TAH* offers the cooling-cost savings of 20.7%, 12.2%, 20.5%, and 7.2% for *WordCount*, *TeraSort*, *Grep*, and *PI*, respectively. We conclude that *TAH* is more beneficial to *WordCount* and *Grep* than *PI*. This comparison study suggests that *TAH* is more energy efficient under data-intensive workload than under computing-intensive workload.



(a) Map task distribution vs application.

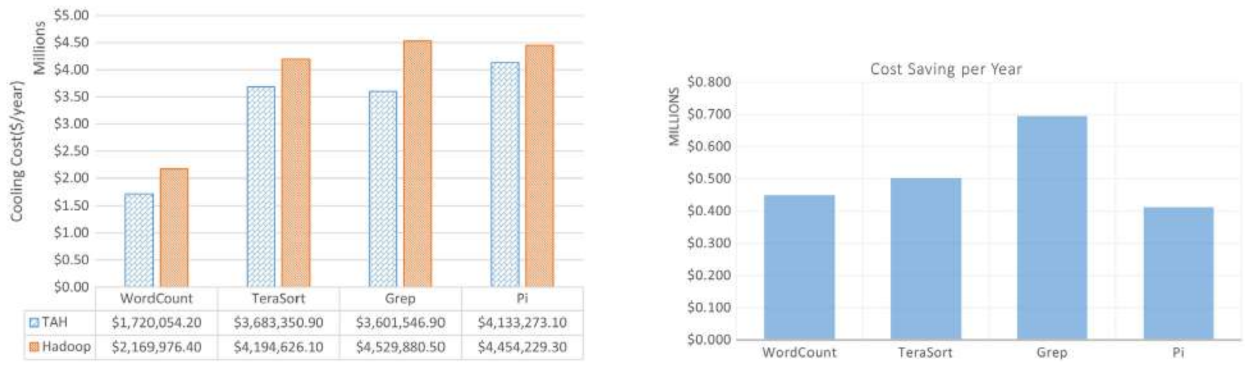


(b) Map-distribution vs node.

Fig. 16. Map distribution vs node.

Table 4
Cooling cost reduction comparison for different benchmarks.

Node	Cooling cost (in Watt)							
	WC(R)	WC(H)	TS(R)	TS(H)	Gr(R)	Gr(H)	Pi(R)	Pi(H)
1	60.734	48.587	61.294	53.259	59.799	60.734	59.799	56.996
2	59.239	56.062	59.799	60.548	56.996	60.360	58.865	59.239
3	56.062	55.688	58.865	60.734	56.956	59.789	58.304	57.930
4	54.380	56.996	57.557	57.930	56.062	59.052	57.552	58.304
5	53.259	56.249	56.062	57.744	56.060	57.865	56.996	57.557
6	50.455	54.567	53.259	57.557	56.435	57.928	56.622	57.549
7	47.092	57.193	52.324	56.062	56.001	57.183	55.127	59.425
Avg	54.460	55.047	57.023	58.690	56.916	59.132	57.610	58.144



	WordCount	TeraSort	Grep	Pi
▨ TAH	$1,720,054.20	$3,683,350.90	$3,601,546.90	$4,133,273.10
■ Hadoop	$2,169,976.40	$4,194,626.10	$4,529,880.50	$4,454,229.30

(a) Cooling cost comparison.

(b) Cooling cost savings.

Fig. 17. Cooling cost comparison.

Fig. 17(b) indicates that *TAH* achieves a cooling-cost saving anywhere from $400,000 to $700,000, assuming the electricity unit price resembles those in the states like Iowa, California, and New York. Again, the results confirm that *TAH* offers the *Grep* benchmark the highest cooling-cost saving rate compared to the other three counterparts.

Interestingly, we observe that out of all the jobs requested by the task trackers *TAH* rejects 3% jobs, which is considered minimal in case of running an enormous number of jobs. A handful of jobs are black listed in Hadoop's typical processing due to unavailable resources for all the jobs. Nevertheless, the *TAH* powered cluster exhibits little performance degradation in terms of execution time.

If some jobs should be processed in a timely manner, then the jobs can be assigned high priorities so that *TAH* favors high performance over thermal efficiency while running these time-sensitive jobs on Hadoop clusters. We notice that the number of rejected jobs is reduced by increasing the cluster size. We conclude that deploying *TAH* to large-scale Hadoop clusters can achieve tremendous cooling-cost without adversely affecting system performance.

5. Discussions and future work

In this section, we discuss a few open research issues related to *TAH*. We shed some light on possibilities of extending this work to further enhance thermal and energy efficient of next-generation data centers.

The thermal model applied in our study incorporates CPU and disk contributions to outlet temperatures of data nodes in Hadoop clusters. We model disk power state using two modes — active and idle modes. This two-state power model of disks can be replaced by more comprehensive power models. An enhanced power model will improve impact measurements of disks on outlet temperatures. We plan to conduct disk profiling to tune the disk power model, thereby accurately quantifying disk contributions to outlet temperatures.

We will integrate the thermal module with the other schedulers (e.g., the Fair and Capacity scheduler). We have implemented the thermal module in the FIFO scheduler, which maintains a single task queue. The schedulers developed by Facebook and Yahoo manage multiple queues to offer parallel scheduling capabilities. Seamlessly integrating our thermal management module with these schedulers may be interesting both on thermal management and performance optimization fronts.

To boost energy efficiency of data centers, we plan to design a dynamic power management strategy to adjust cluster size in a way to reduce the overall power consumption. We intend to implement the power manager in the Hadoop's HDFS data flow.

The proposed manager will dynamically scale up or down the number of data nodes depending on workload conditions. Specifically, the manager will scale up a cluster when CPU/disk utilization of data nodes is beyond a threshold; the cluster will be scaled down if load drops below a given level. During the scaling process, data blocks may be re-balanced among inter-rack or Intra-rack nodes.

HDFS maintains three replicas for each data block; two copies are located on one rack whereas the third one is placed on another rack. The power manager will rely on replicas and their locations to judiciously determine data nodes to be power off or on under given load.

6. Conclusion

In this study, we developed a thermal-aware scheduler called *TAH* for Hadoop clusters. *TAH* deploys a thermal model to capture heat-flow behaviors affecting cooling cost in Hadoop-enabled data centers. *TAH* takes into account CPU/disk contributions to outlet temperatures in addition to heat re-circulation impacts. Keeping track of CPU/disk utilization and temperatures, *TAH* fosters thermal awareness during the course of dispatching tasks to data node in a Hadoop cluster.

We took the advantage of the lightweight heartbeat mechanism to communicate the thermal-related data (i.e., CPU/disk utilization and temperatures) from the *Task Tracker* to the *Job Tracker* in *TAH*.

We adopted a cross interference matrix coupled with rack server placements (i.e., a height vector) to represent heat re-circulation effect, which is a centerpiece of estimating temperatures associated with rack servers.

TAH is capable of dynamically balancing load across all the nodes while reducing heat emissions. For example, if projected peak temperature of a node is blow redline temperature, *TAH* will continue assign tasks to the server. If it does make any of its neighbor server to exceed threshold temperature it skips assigning job to the task. Thus, *TAH* schedules jobs across the data nodes based on thermal feedback to minimize peak temperatures and to reduce cooling cost in data centers.

We evaluated the effectiveness of our *TAH* by running four Hadoop benchmarks - *WordCount*, *Grep*, *PI*, and *TeraSort*. The experimental results show that *TAH* adequately reduces energy cost with a marginal performance overhead. Compared with the existing scheduler, our thermal-aware *TAH* effectively conserves energy consumption by up to 20.7% with an average of 15.2%.

Acknowledgments

Xiao Qin's work was supported by the US National Science Foundation under Grants IIS-1618669, CCF-0845257, CNS-0757778, CCF-0742187, CNS-0917137, CNS-0831502, CNS-0855251, OCI-0753305, DUE-0837341, and DUE-0830831.

References

[1] J. Shuja, K. Bilal, S.A. Madani, M. Othman, R. Ranjan, P. Balaji, S.U. Khan, Survey of techniques and architectures for designing energy-efficient data centers, IEEE Syst. J. 10 (2) (2016) 507–519.
[2] E. Naserian, S.M. Ghoreyshi, H. Shafiei, P. Mousavi, A. Khonsari, Cooling aware job migration for reducing cost in cloud environment, J. Supercomput. 71 (3) (2015) 1018–1037.
[3] A.-C. Orgerie, M.D.d. Assuncao, L. Lefevre, A survey on techniques for improving the energy efficiency of large-scale distributed systems, ACM Comput. Surv. 46 (4) (2014) 47.
[4] R.T. Kaushik, M. Bhandarkar, Greenhdfs: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster, in: Proceedings of the USENIX Annual Technical Conference, 2010, p. 109.
[5] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P.P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q.M. Malluhi, N. Tziritas, A. Vishnu, et al., A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems, Computing 98 (7) (2016) 751–774.
[6] H. Sun, P. Stolf, J.-M. Pierson, G. Da Costa, Energy-efficient and thermal-aware resource management for heterogeneous datacenters, Sustain. Comput.: Inf. Syst. 4 (4) (2014) 292–306.
[7] X. Gao, Z. Xu, H. Wang, L. Li, X. Wang, Why some like it hot too: Thermal attack on data centers, in: Proceedings of the 2017 ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems, ACM, 2017, pp. 23–24.
[8] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, A.Y. Zomaya, Energy-efficient data replication in cloud computing datacenters, Clust. Comput. 18 (1) (2015) 385–402.
[9] D. Kliazovich, P. Bouvry, S.U. Khan, Dens: data center energy-efficient network-aware scheduling, Clust. Comput. 16 (1) (2013) 65–75.
[10] K. Ebrahimi, G.F. Jones, A.S. Fleischer, A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities, Renew. Sustain. Energy Rev. 31 (2014) 622–638.
[11] C.D. Patel, C.E. Bash, C. Belady, L. Stahl, D. Sullivan, Computational fluid dynamics modeling of high compute density data centers to assure system inlet air specifications, in: Proceedings of IPACK, vol. 1, 2001, pp. 8–13.
[12] Y. Li, H. Zhang, K.H. Kim, A power-aware scheduling of mapreduce applications in the cloud, in: Dependable, Autonomic and Secure Computing, DASC, 2011 IEEE Ninth International Conference on, IEEE, 2011, pp. 613–620.
[13] Y. Yang, J. Wei, G. Wei, L. Ping, Z. Yongmin, Modeling on energy consumption of cloud computing based on data center, 2015.
[14] Q. Tang, S.K. Gupta, G. Varsamopoulos, Thermal-aware task scheduling for data centers through minimizing heat recirculation, in: 2007 IEEE International Conference on Cluster Computing, IEEE, 2007, pp. 129–138.
[15] C.D. Patel, A.J. Shah, Cost model for planning, development and operation of a data center, 2005.
[16] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. 28 (5) (2012) 755–768.
[17] Z. Abbasi, G. Varsamopoulos, S.K. Gupta, Tacoma: server and workload management in internet data centers considering cooling-computing power trade-off and energy proportionality, ACM Trans. Archit. Optim. 9 (2) (2012) 11.
[18] C.D. Patel, R. Sharma, C.E. Bash, A. Beitelmal, Thermal considerations in cooling large scale high compute density data centers, in: Thermal and Thermomechanical Phenomena in Electronic Systems, 2002 ITHERM 2002. The Eighth Intersociety Conference on, IEEE, 2002, pp. 767–776.
[19] A.H. Beitelmal, C.D. Patel, Thermo-fluids provisioning of a high performance high density data center, Distrib. Parallel Databases 21 (2–3) (2007) 227–238.
[20] Y. Guo, Y. Gong, Y. Fang, P.P. Khargonekar, X. Geng, Energy and network aware workload management for sustainable data centers with thermal storage, IEEE Trans. Parallel Distrib. Syst. 25 (8) (2014) 2030–2042.
[21] J. Shuja, K. Bilal, S.A. Madani, S.U. Khan, Data center energy efficient resource scheduling, Clust. Comput. 17 (4) (2014) 1265–1277.
[22] Z. Guo, S. Hui, Y. Xu, H.J. Chao, Dynamic flow scheduling for power-efficient data center networks, in: Quality of Service (IWQoS), 2016 IEEE/ACM 24th International Symposium on, IEEE, 2016, pp. 1–10.
[23] Z. Guo, Z. Duan, Y. Xu, H.J. Chao, Jet: electricity cost-aware dynamic workload management in geographically distributed datacenters, Comput. Commun. 50 (2014) 162–174.
[24] Q. Tang, S.K. Gupta, G. Varsamopoulos, Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: a cyber-physical approach, IEEE Trans. Parallel Distrib. Syst. 19 (11) (2008) 1458–1472.
[25] R.K. Sharma, C.E. Bash, C.D. Patel, R.J. Friedrich, J.S. Chase, Balance of power: Dynamic thermal management for internet data centers, IEEE Internet Comput. 9 (1) (2005) 42–49.
[26] J.D. Moore, J.S. Chase, P. Ranganathan, R.K. Sharma, Making scheduling" cool": Temperature-Aware Workload Placement in Data Centers, in: USENIX Annual Technical Conference, General Track, 2005, pp. 61–75.
[27] Q. Tang, S. Gupta, G. Varsamopoulos, Thermal-aware task scheduling for data centers through minimizing heat recirculation, in: Cluster Computing 2007 IEEE International Conference on, sept. 2007, pp. 129–138.
[28] A. Sansottera, P. Cremonesi, Cooling-aware workload placement with performance constraints, Perform. Eval. 68 (11) (2011) 1232–1246.
[29] M.G. Noll, Hadoop benchmarking, http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasort-testdfsio-nnbench-mrbench/.

Yi Zhou (S'16) received the B.E.E., M.S.E.E. degrees in electronic engineering all from Beijing University of Technology, Beijing, in 2006 and 2010 respectively. He is currently a doctoral student in the Department of Computer Science and Software Engineering at Auburn University. Prior to joining Auburn University in 2013, he has been a software engineer in Alcatel-Lucent Technologies (China) Co.,Ltd. for four years. His research interests include energy-saving techniques, database systems, and parallel computing.

Shubbhi Taneja (S'15) received her B.E. degree from Maharishi Dayanand University, Haryana, India in 2012. She joined Samuel Ginn College of Engineering, Auburn University, to pursue a doctoral student of Computer Science in 2013. Her main area of research includes parallel and distributed systems, storage systems, energy efficient computing, and performance evaluation. Her other areas of interest include web development and STEM programs for K-12 students. She is a student member of IEEE, ACM, and IEEE Computer Society.

Xiao Qin (S'00-'04M-SM'09) received the B.S. and M.S. degrees in computer science from Huazhong University of Science and Technology in 1992 and 1999, respectively. He received the Ph.D. degree in computer science from the University of Nebraska-Lincoln in 2004. He is currently an associate professor in the Department of Computer Science and Software Engineering at Auburn University. Prior to joining Auburn University in 2007, he had been an assistant professor with New Mexico Institute of Mining and Technology (New Mexico Tech) for three years. He won an NSF CAREER award in 2009. His research is supported by the US National Science Foundation (NSF), Auburn University, and Intel Corporation. He has been on the program committees of various international conferences, including IEEE Cluster, IEEE MSST, IEEE CCGrid, IEEE IPCCC, and ICPP. His research interests include parallel and distributed systems, storage systems, fault tolerance, real-time systems, and performance evaluation. He is a member of the ACM and a senior member of the IEEE.

Jifu Zhang received the B.S. and M.S. in Computer Science and Technology from Hefei University of Technology, China, in 1983 and 1989, respectively. He received the Ph.D. degree in Pattern Recognition and Intelligence Systems from Beijing Institute of Technology in 2005. He is currently a Professor in the School of Computer Science and Technology at TYUST. His research interests include data mining and artificial intelligence, parallel and distributed systems.

Minghua Jiang received the B.S. Mathematic from Beijing Normal University in 1988. He received the M.S. and Ph.D. degree from Huazhong University of Science and Technology in 1999 and 2011, respectively. He is currently an professor in the School of Mathematic and Computer Science at Wuhan Textile University. His research interests include parallel and distributed systems, storage systems, service computing.

Mohammed I. Alghamdi (S'04-'08M-SM'13) received the B.S. degree in Computer Science from King Saud University, Riyadh, Saudi Arabia in 1999. He received the M.S. degrees in Computer Science from Colorado Technical University, Denver, Colorado in 2003. He received the Ph.D. degree in Computer Science from New Mexico Institute of Mining and Technology in 2008. Currently, he is an Assistant Professor with the Department of Computer Science, Al-Baha University, Kingdom of Saudi Arabia. His research interests include wireless networks, storage systems, parallel and distributed systems, and computer system security. He is a senior member of IEEE.