2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)

Towards Thermal-Efficient Hadoop Clusters through Scheduling

Yi Zhou*, Mohammed Alghamdi†, Gautam Dudeja, Shubbhi Taneja‡*, Xiao Qin*
*Department of Computer Science and Software Engineering
Auburn University, Auburn, Alabama 36849-5347
Email: {yzz0074, gzd0004, shubbhi, xqin}@auburn.edu ‡corresponding author:shubbhi@auburn.edu
†Computer Science Department
Al-Baha University, Al-Baha City, Kingdom of Saudi Arabia

***Abstract*—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 2.5C over the default FIFO scheduler; our scheduling solution saves approximately 15% of cooling cost with marginal performance degradation.**

***Keywords*-Data Centers, Thermal Energy, Hadoop, MapReduce**

I. Introduction

Energy cost spent on cooling systems turns into a huge fraction of a data center's maintenance cost. The cooling cost nowadays is much higher than the cost of building data centers housing IT equipments [1]. A handful of studies were conducted to reduce the energy cost in light of improving the server operational efficiency. 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% [2] of energy consumption without dramatically affecting system performance.

Three reasons motivate us to 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 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. 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 [3]. 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 [4]. For example, due to the heat re-circulation, a rack server's heat dissipation requirements is roughly 30 times larger than those of servers in 1990 [5].

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 [6]. Most existing energy consumption models ignore the correlations between heat re-circulation and hardware workload (e.g., CPU utilization and disk utilization). 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 the cooling systems. The heat re-circulation effect exists, because a portion of the heat emitted from a server's outlet is recirculated into other servers' inlets. Such an effect further drives up the temperatures, which in turn lead to an extra cost of cooling down the overall temperature in the data centers. Although static and dynamic approaches were proposed to alleviate the heat re-circulation effect, it remains an open issue of how to systematically minimize heat re-circulation in Hadoop clusters. Due to the lack of a heat flow model characterizing the heat re-circulation in terms of the sever placements, thermal-aware scheduling in Hadoop becomes a challenging task. We are motivated by this challenge to develop the scheduler to thermal-aware Hadoop scheduler by incorporating an effective heat model.

978-1-5386-3066-2/17 $31.00 © 2017 IEEE
DOI 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.84

IEEE computer society

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:05:57 UTC from IEEE Xplore. Restrictions apply.

II. Related Work

A. Thermal-Efficient Computing

Thermal-efficient computing has increasingly become prevalent in data centers. Abbasi *et al.* developed the *TACOMA* system, which decreases heat distribution by server provisioning [7]. *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 [8][9].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.

B. 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 [10]. 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.* also implemented a resource scheduling algorithm minimizing a subset of resources to service computing workload [11]. Unlike the aforementioned thermal management approaches that are designed for general data center, our solution is focused on thermal management in Hadoop clusters.

C. 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 [12]. *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 [13], which heuristically solves temperature imbalances by redistributing workload inversely proportional to chassis outlet temperatures.Unfortunately, the existing thermal-efficient solutions are inadequate to 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.

III. 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. 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 Section III-A, we present the thermal model, a centerpiece in *TAH*. Last, we detail the implementation of *TAH* in Section III-B.

A. TAH 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.

1) The Thermal Model: 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.

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.

Variables	Description
T_{sup}	Air temperature as supplied from cooling unit
T_{out}	Outlet Temperature (c)
T_{in}	Inlet Temperature (c)
$\triangle T$	Change/Rise in temperature (c)
T_{red}	Red Line/Threshold temperature
$\alpha_{i,j}$	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
$T_{in}^{possilble}$	Possible Inlet temperature if job assigned
C_{util}	CPU utilization
D_{util}	Disk Utilization

Table I: Model Notation

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 2 where K_{disk} is a constant contribution of an active disk to outlet temperature T_{out}).

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. 1 shows the actual representation of Inlet Temperature of a server at

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:05:57 UTC from IEEE Xplore. Restrictions apply.

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 [14].

$$R = 1 - \frac{d^i}{d}$$
$$T_{in}{}^i = T^i_{initial} - RT_{sup}{}^i + AT_{out} \quad (1)$$

1 can be reorganized to obtain outlet temperatures.

R in 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 1 A is a heat re-circulation or cross-interference vector, which sums up the temperature rise of a server node. Each element $\alpha_{i,j}$ from the matrix A represents the fraction contribution of heat from server $node_i$'s outlet to $node_j$'s inlet temperature. Such a summation is reasonable, because the node's neighbors sitting at placement outlet temperature.

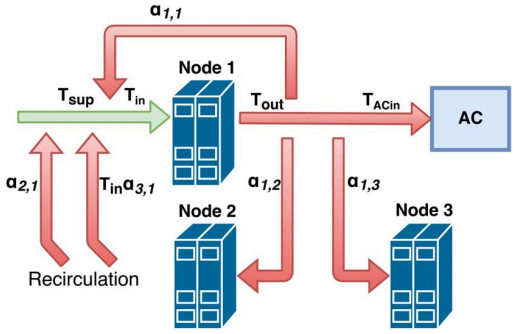


Figure 1: Cross-Interference Between Neighbor Servers

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 exampe, [15]). Fig. 1 demonstrates the cross-interference between neighbor servers.

$$T_{rise}{}^i = K_{disk}{}^i + aC_{util}{}^i + b$$
$$T_{out}{}^i = T^i_{initial} + T_{rise} \quad (2)$$

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.

2) HeartBeat Mechanism Modification: In order to obtain a task tracker's thermal information feeding into the thermal model, we have to build a channel in which Task Trackers and a Job Tracker communicate in a timely manner. A few available mechanisms exist in Hadoop include *Metrics*, *Counters*, and *HeartBeat*. Metrics are non-core part of Hadoop's internal MapReduce interface; rather, Metrics are envisioned as an external process. We choose to adopt *Heartbeat* thanks to its 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.

B. Architectural Framework and Implementation

To incorporate thermal awareness into schedulers, we implement our *TAH* in the Hadoop system. In this section, we discuss the design and implementation issues of *TAH*. Fig. 2 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.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:05:57 UTC from IEEE Xplore. Restrictions apply.

- Blocks processed by all the map tasks share the same size.
- Each task is assigned to utmost one Task Tracker.
- Heat re-circulation effect is more profound to servers mounted on the same rack.

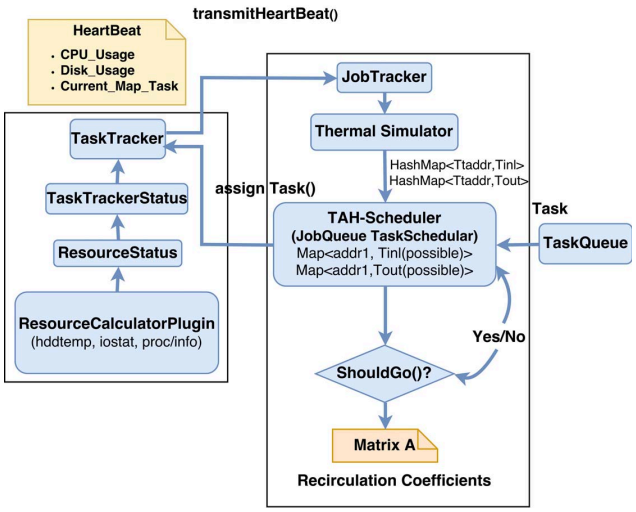


Figure 2: TAH Design Implementation

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.

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) 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 onward each time task tracker i 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 2 in Section III-A) 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 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 effect 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 3).

$$\triangle T^i = {T_{out}}^i - {T_{in}}^i \tag{3}$$

6) Now a method calculates the temperature rise per map task by using 4. 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.

$$\text{Temp rise per map} = \frac{\triangle T}{map_i} \tag{4}$$

7) We employ a method to obtain and maintain estimated outlet temperatures if a job is assumed to be assigned to that tracker $T_{out}^{possilble}$.
8) Given server i, we introduce a method calculating the average of $\alpha_{i,j}$ where j ε [1...n]. Here n is number of Task Trackers in cluster as in 5.

$$\text{Average Factor for server i} = \frac{\sum_1^n}{n} \tag{5}$$

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

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:05:57 UTC from IEEE Xplore. Restrictions apply.

(see 5). Once known how the neighbor servers could be critically affected by the re-circulation effect of heat, the scheduler then applies $T_{out}^{possilble}$ to calculate the inlet temperatures of all its neighbor nodes; the temperatures are kept as $T_{in}^{possilble}$.

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 6 is for server i and all its neighbors, estimated temperature $T_{in}^{possilble}$ is lower than redline temperature T_{red}.

$$\text{criteria} = T_{in}^{possilble} < T_{red} \tag{6}$$

11) If server i doesn't 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.

IV. Experimental Results

We evaluate the energy efficiency of a ten-node Hadoop cluster, where the prototype of *TAH* is implemented. Table II shows the hardware configuration of the computing nodes (i.e., NameNode and DataNode) in the Hadoop cluster.

Table II: Server Specifications

Node	Number of cores	RAM	Storage
HP Xeon	4 cores(2.8GHz)	2GB	143GB
Dell	4 cores(2.8GHz)	2GB	143GB

To evaluate our schedulers, popular Hadoop Benchmarks are chosen to stress test a cluster with a wide range of CPU and disk workload. The Benchmarks considered include *WordCount*, *Distributed Grep*, *PI estimation*, and *Tera Sort*.

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. Figs. 3(a) and (b) show the number of *WordCount* task distribution among the five nodes processing data of 10GB and 15GB, 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 assigns 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.

Comparing Fig. 3(a) and (b), we observe that regardless of the applied scheduler, increasing the data size from 10G to 15G 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.

Figs. 4(a) and (b) depict the inlet temperature distributions when the Hadoop and *TAH* scheduler is employed, respectively. 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.

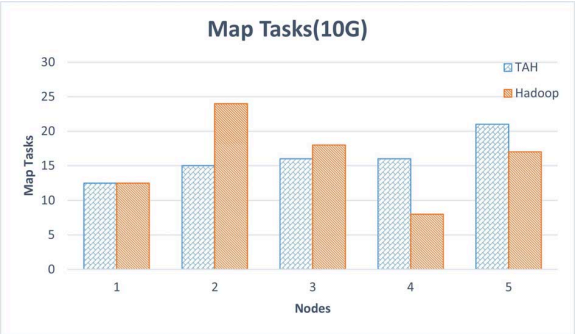
After comparing Fig. 4(a) with Fig. 4(b), we conclude that the amount of processed data affects peak inlet temperatures. This trend is applicable to the two evaluated schedulers. An intriguing observation is that a large amount input data allows the *WordCount* benchmark take further thermal benefit from *TAH*. The experimental results suggest that data-intensive applications like *WordCount* can take full thermal advantage from our *TAH*.

Now we evaluate *TAH* using the *Grep* benchmark. Figs. 5(a) and 5(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. Figs. 6(a) and 6(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 reduced.

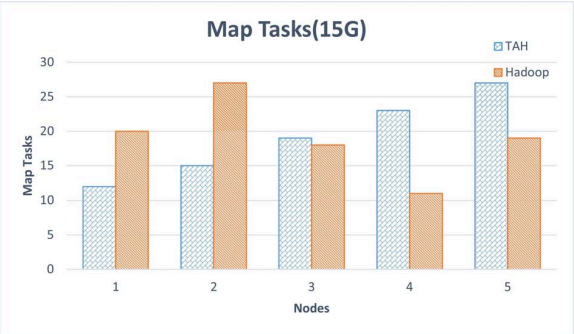
The *TeraSort* applicationis capable of sorting a massive amount of data (e.g., 1TB) in a fast manner. The *TeraSort* benchmark facilitate an integrated testing of the HDFS and MapReduce layers in Hadoop clusters. The TeraSort benchmark suite offers added benefits allowing us to measure thermal behavior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

Figs. 7(a) and (b) illustrate that *TAH* lowers the peak inlet 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

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:05:57 UTC from IEEE Xplore. Restrictions apply.

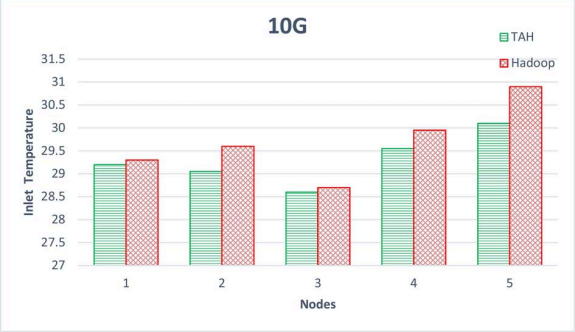


(a) Number of Map Tasks vs Node(10GB)

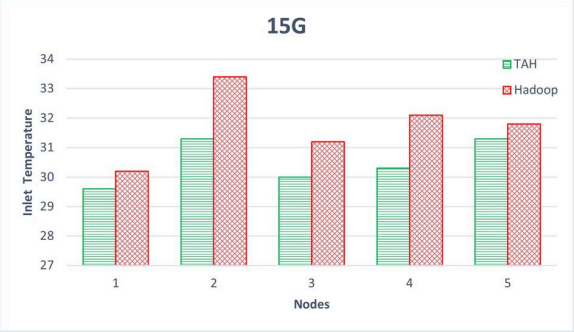


(b) Number of Map Tasks vs Node(15GB)

Figure 3: Map Task Distribution for WordCount

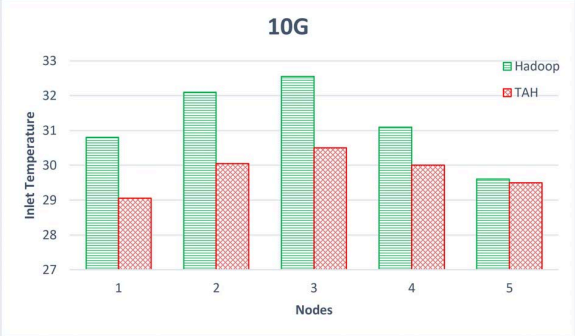


(a) Peak Inlet Temperature vs Node (10GB)

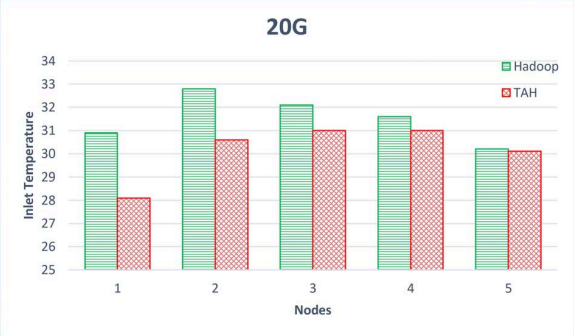


(b) Peak Inlet Temperature vs Node (15GB)

Figure 4: Peak Inlet Temperature for WordCount



(a) Peak Inlet Temperature vs Node (10GB)



(b) Peak Inlet Temperature vs Node (20GB)

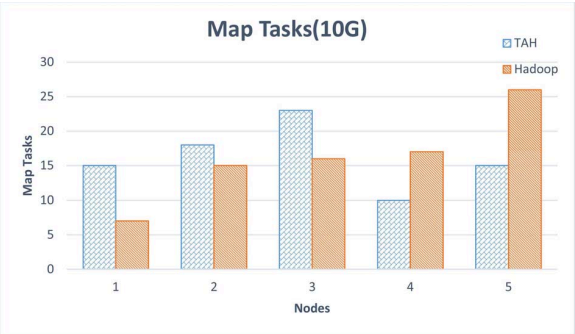
Figure 5: Peak Inlet Temperature for Grep

average of 2.3 °C and 2.2 °C for the input data size of 10GB and 20GB, respectively. In the 10GB 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 20GB 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.

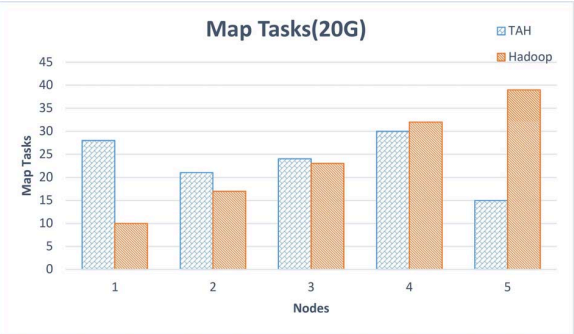
Again, we attribute the peak-temperature reductions offered by *TAH* to task load distributions, which are plotted in Figs. 8(a) and (b). The results show that *TAH* suppresses peak inlet temperatures of hot spots by minimizing the number of running tasks. The task distribution patterns observed in Fig. 8 agree with those of Figs. 6 and 3.

Let us exam the last benchmark - *PI*, which is a Hadoop program computing the PI value. Unlike WordCount and

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:05:57 UTC from IEEE Xplore. Restrictions apply.

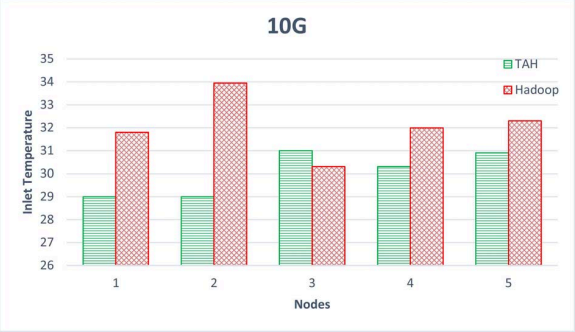


(a) Number of Map Tasks vs Node(10GB)

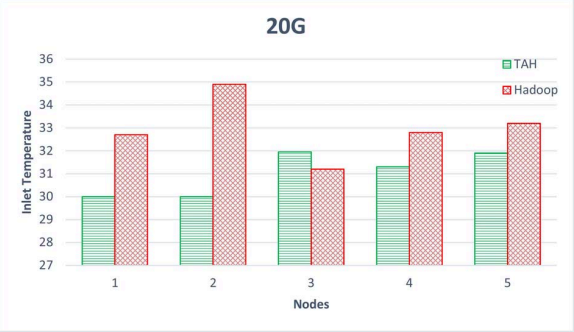


(b) Number of Map Tasks vs Node(20GB)

Figure 6: Map Task distribution for Grep

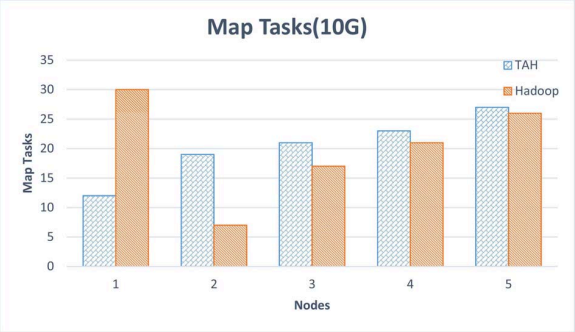


(a) Peak Inlet Temperature vs Node (10GB)

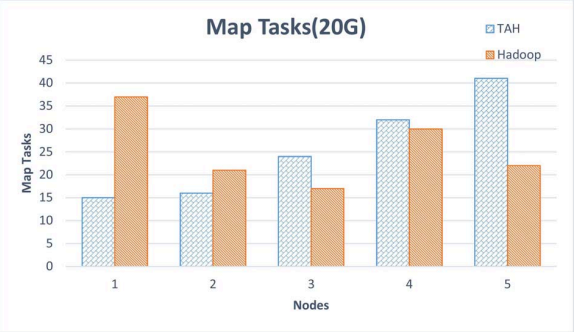


(b) Peak Inlet Temperature vs Node (20GB)

Figure 7: Peak Inlet Temperature for Tera Sort



(a) Number of Map Tasks vs Node(10GB)



(b) Number of Map Tasks vs Node(20GB)

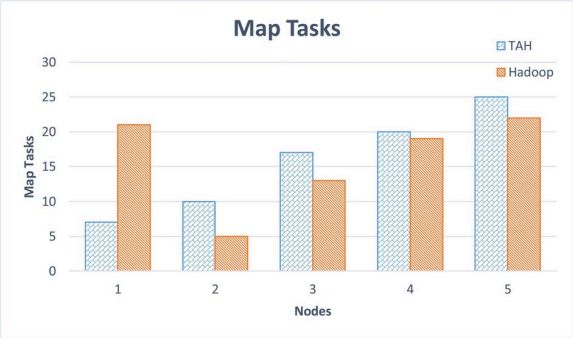
Figure 8: Map Task distribution for Tera Sort

Grep, the *PI* benchmark is a CPU-intensive job. *PI* loads no data from HDFS on the cluster. Fig. 9(a) plots the *PI* task distributions among the data nodes, whereas Fig. 9(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. 9(a)). Consequently, *TAH* lowers the peak inlet temperatures by up to 4.5 °C, with an average of 2.2 °C (see Fig. 9(b)). The task distribution trend and the peak-temperature trends are similar to those of the previous three benchmarks.

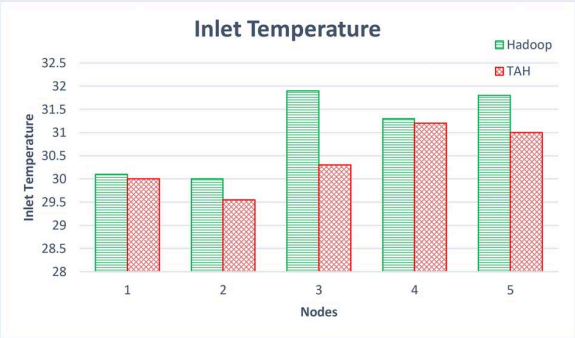
V. Conclusion

In this study, we developed *TAH* - a thermal-aware scheduler 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

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:05:57 UTC from IEEE Xplore. Restrictions apply.



(a) Map Task Distribution vs Node



(b) Peak Inlet Temperature vs Node

Figure 9: Pi Estimation Map Distribution and Peak Inlet Temperature Reduction

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 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%.

REFERENCES

[1] E. Naserian, S. M. Ghoreyshi, H. Shafiei, P. Mousavi, and A. Khonsari, "Cooling aware job migration for reducing cost in cloud environment," *The Journal of Supercomputing*, vol. 71, no. 3, pp. 1018–1037, 2015.

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

[3] D. Kliazovich, P. Bouvry, and S. U. Khan, "Dens: data center energy-efficient network-aware scheduling," *Cluster computing*, vol. 16, no. 1, pp. 65–75, 2013.

[4] K. Ebrahimi, G. F. Jones, and A. S. Fleischer, "A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities," *Renewable and Sustainable Energy Reviews*, vol. 31, pp. 622–638, 2014.

[5] C. D. Patel, C. E. Bash, C. Belady, L. Stahl, and 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.

[6] Y. Li, H. Zhang, and 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.

[7] Z. Abbasi, G. Varsamopoulos, and S. K. Gupta, "Tacoma: Server and workload management in internet data centers considering cooling-computing power trade-off and energy proportionality," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 9, no. 2, p. 11, 2012.

[8] C. D. Patel, R. Sharma, C. E. Bash, and 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.

[9] A. H. Beitelmal and C. D. Patel, "Thermo-fluids provisioning of a high performance high density data center," *Distributed and Parallel Databases*, vol. 21, no. 2-3, pp. 227–238, 2007.

[10] Y. Guo, Y. Gong, Y. Fang, P. P. Khargonekar, and X. Geng, "Energy and network aware workload management for sustainable data centers with thermal storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 8, pp. 2030–2042, 2014.

[11] J. Shuja, K. Bilal, S. A. Madani, and S. U. Khan, "Data center energy efficient resource scheduling," *Cluster Computing*, vol. 17, no. 4, pp. 1265–1277, 2014.

[12] Q. Tang, S. K. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 11, pp. 1458–1472, 2008.

[13] R. K. Sharma, C. E. Bash, C. D. Patel, R. J. Friedrich, and J. S. Chase, "Balance of power: Dynamic thermal management for internet data centers," *Internet Computing, IEEE*, vol. 9, no. 1, pp. 42–49, 2005.

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

[15] A. Sansottera and P. Cremonesi, "Cooling-aware workload placement with performance constraints," *Performance Evaluation*, vol. 68, no. 11, pp. 1232–1246, 2011.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 23:05:57 UTC from IEEE Xplore. Restrictions apply.