Thermal Aware Workload Scheduling with Backfilling for Green Data Centers

Lizhe Wang†, Gregor von Laszewski‡, Jai Dayal† and Thomas R. Furlani†

† Service Oriented Cyberinfrastructure Lab, Rochester Institute of Technology, Rochester, NY 14623
‡ Pervasive Technology Institute, Indiana University at Bloomington, Bloomington, IN 47408
† Center for Computational Research, State University of New York at Buffalo, Buffalo, NY 14203

Abstract—Data centers now play an important role in modern IT infrastructures. Related research has shown that the energy consumption for data center cooling systems has recently increased significantly. There is also strong evidence to show that high temperatures with in a data center will lead to higher hardware failure rates and thus an increase in maintenance costs. This paper devotes itself in the field of thermal aware resource management for data centers. This paper proposes an analytical model, which describes data center resources with heat transfer properties and workloads with thermal features. Then a thermal aware task scheduling algorithm with backfilling is presented which aims to reduce power consumption and temperatures in a data center. A simulation study is carried out to evaluate the performance of the algorithm. Simulation results show that our algorithm can significantly reduce temperatures in data centers by introducing endurance decline in performance.

Keywords - Thermal aware, data center, task scheduling

I. INTRODUCTION

Electricity usage is the most expensive portion of a data center’s operational costs. In fact, the U.S. Environmental Protection Agency (EPA) reported that 61 billion KWh, 1.5% of US electricity consumption, is used for data center computing [1]. Additionally, the energy consumption in data centers doubled between 2000 and 2006. Continuing this trend, the EPA estimates that the energy usage will double again by 2011. It is reported in [2] that power and cooling cost is the most dominant cost in data centers [2]. It is reported that cooling costs can be up to 50% of the total energy cost [2]. Even with more efficient cooling technologies in IBM’s BlueGene/L and TACC’s Ranger, cooling cost still remains a significant portion of the total energy cost for these data centers. It is also noted that the life of a computer system is directly related to its operating temperature. Based on Arrhenius time-to-fail model [2], every 10°C increase of temperature leads to a doubling of the system failure rate. Hence, it is important to level down data center operational temperatures for reducing energy cost and increasing reliability of compute resources.

Consequently, resource management with thermal considerations is important for data center operations. Some research has developed elaborate models and algorithms for data center resource management with computational fluid dynamics (CFD) techniques [3], [4]. Other work [4], [5], however, argues that the CFD based model is too complex and is not suitable for online scheduling. This has lead to the development of some less complex online scheduling algorithms. Sensor-based fast thermal evaluation model [6], [7], Generic Algorithm & Quadratic Programming [8], [9], and the Weatherman – an automated online predictive thermal mapping [10] are a few examples.

In this paper, we propose a Thermal Aware Scheduling Algorithm with Backfilling (TASA-B) for data centers. In our previous work [11], a Thermal Aware Scheduling Algorithm (TASA) is developed for data center task scheduling. The TASA-B extends the TASA by allowing the backfilling of tasks under thermal constrains. Our work differs from the above systems in that we’ve developed some elaborate heat transfer models for data centers. Our model, therefore, is less complex than the CFD models and can be used for on-line scheduling in data centers. It however can provide enough accurate description of data center thermal maps than [11], [12]. In detail, we study a temperature based workload model and a thermal based data center model. This paper then defines the thermal aware workload scheduling problem for data centers and presents the TASA-B for data center workloads. We use simulations to evaluate thermal aware workload scheduling algorithms and discuss the trade-off between throughput, cooling cost, and other performance metrics. Our unique contribution is shown as follows. We propose a general framework for thermal aware resource management for data centers. Our framework is not bound to any specific model, such as the RC-thermal model, the CFD model, or a task-temperature profile. A new heuristic of TASA-B is developed and evaluated, in terms of performance loss, cooling cost, and reliability.

The rest of this paper is organized as follows. Section II introduces the related work and background of thermal aware workload scheduling in data centers. Section III presents mathematical models for data center resources and workloads. We present our thermal aware scheduling algorithm with backfilling for data centers in Section IV. We evaluate the algorithm with a simulation in Section V. The paper is finally concluded in Section VI.

II. RELATED WORK AND BACKGROUND

A. Data center operation

The racks in a typical data center, with a standard cooling layout based on under-floor cold air distribution, are back-to-
back and laid out in rows on a raised floor over a shared plenum. Modular computer room air conditioning (CRAC) units along the walls circulate warm air from the machine room over cooling coils, and direct the cooled air into the shared plenum. The cooled air enters the machine room through floor vent tiles in alternating aisles between the rows of racks. Aisles containing vent tiles are cool aisles; equipment in the racks is oriented so their intake draws inlet air from cool aisles. Aisles without vent tiles are hot aisles providing access to the exhaust air, and typically, rear panels of the equipment [2].

Thermal imbalances interfere with efficient cooling operation. Hot spots create a risk of redlining servers by exceeding the specified maximum inlet air temperature, damaging electronic components and causing them to fail prematurely. Non-uniform equipment loads in the data center cause some areas to heat more than others, while irregular air flows cause some areas to cool less than others. The mixing of hot and cold air in high heat density data centers leads to complex airflow patterns that create these troublesome hot spots. Therefore, objectives of thermal aware workload scheduling are to reduce both the maximum temperature for all compute nodes and the imbalance of the thermal distribution in a data center. In a data center, the thermal distribution and computer node temperatures can be obtained by deploying ambient temperature sensors, on-board sensors [2], [2], and with software management architectures like Data Center Observatory [2], Mercury & Freon [2], LiquidN2 & C-Oracle[2].

### B. Task-temperature profiling

Given a certain compute processor and a steady ambient temperature, a task-temperature profile is the temperature increase along with the task’s execution. It has been observed that different types of computing tasks generate different amounts of heat, therefore resulting with distinct task-temperature profiles [2].

![Temperature profile](image1)

**Fig. 1.** Task-temperature profile of a compute node in buffalo data center

Task-temperature profiles can be obtained by using some profiling tools. Figure 1 shows the task-temperature profile of a compute node in the Center for Computational Research (CCR) of State University of New York at Buffalo. The X-axis is the time and the Y-axis gives two values: the average workload (task execution time in the passed hour) and the average temperature for a compute node. Figure 1 clearly indicates the task and temperature correlation: normally as computation loads in term of task CPU time augment, compute node temperatures increases incidentally. Figure 2 shows a task-temperature profile, which is obtained by running SPEC 2000 benchmark (swim) on a IBM BladeCenter with 2 GB memory and Red Hat Enterprise Linux AS 3 [2]. It is both constructive and realistic to assume that the knowledge of task-temperature profile is available based on the discussion [2], [2] that task-temperature can be well approximated using appropriate prediction tools and methods.

### III. SYSTEM MODELS AND PROBLEM DEFINITION

#### A. Compute resource model

This section presents formal models of data centers and workloads and a thermal aware scheduling algorithm, which allocates compute resources in a data center for incoming workloads with the objective of reducing temperature in the data center. A data center $DataCenter$ is modeled as:

$$DataCenter = \{Node, TherMap\} \quad (1)$$

where, $Node$ is a set of compute nodes, $TherMap$ is the thermal map of a data center.

A thermal map of a data center describes the ambient temperature field in a 3-dimensional space. The temperature field in a data center can be defined as follows:

$$TherMap = Temp( <x, y, z >, t) \quad (2)$$

It means that the ambient temperature in a data center is a variable with its space location $(x, y, z)$ and time $t$.

We consider a homogeneous compute center: all compute nodes have identical hardware and software configurations. Suppose that a data center contains $I$ compute nodes as shown in Figure 13:

$$Node = \{node_i | 1 \leq i \leq I\} \quad (3)$$

![swim](image2)

**Fig. 2.** Task-temperature profile of SPEC2000 (swim) [2]
The $i^{th}$ compute node is described as follows:

$$node_i = (x, y, z, t, Temp(t))$$

(4)

where, $< x, y, z >$ is node's location in a 3-dimensional space. $t$ is the time when $node_i$ is available for job execution. $Temp(t)$ is the temperature of $node_i$ at time $t$.

Figure 3 shows the layout compute nodes and their ambient environment. To be simplicity, we assume the compute nodes and their ambient environment shares the same 3-dimensional position $< x, y, z >$.

Fig. 3. Layout of compute nodes and their ambient environment.

The process of heat transfer of $node_i$ is described with a RC-thermal model [2], [2], [2]. As shown in Figure 4, $P$ denotes the power consumption of compute node at current time $t$, $C$ is the thermal capacitance of the compute node, $R$ denotes the thermal resistance, and $Temp(node_i, < x, y, z >, t)$ represents the ambient temperature of $node_i$ in the thermal map. Therefore the heat transfer between a compute node and its ambient environment is described in the Eq 5 (also shown in Figure 5).

It is supposed that an initial die temperature of a compute node is constant during the period $[0, t]$ (this can be true when calculating for a short time period). Then the compute node temperature $node_i.Temp(t)$ is calculated as Eq 6.

![RC-thermal model](image)

Fig. 4. RC-thermal model.

**D. Research issue definition**

Based on the above discussion, a job schedule is a map from a job $job_j$ to certain work node $node_i$ with starting time $job_j.start$:

$$schedule_j : job_j \rightarrow (node_i, job_j.start)$$

(10)

A workload schedule $Schedule$ is a set of job schedules $\{schedule_j|job_j \in Job\}$ for all jobs in the workload:

$$Schedule = \{schedule_j|job_j \in Job\}$$

(11)

We define the workload starting time $T_0$ and finished time $T_\infty$ as follows:

$$T_\infty = \max_{1 \le j \le J} \{job_j.t_{arrival} + job_j.t_{req}\}$$

(12)

$$T_0 = \min_{1 \le j \le J} \{job_j.t_{arrival}\}$$

(13)

Then the workload response time $T_{response}$ is calculated as follows:

$$T_{response} = T_\infty - T_0$$

(14)

Assuming that the specified maximum inlet air temperature in a data center is $TEMP_{max}$, thermal aware workload scheduling in a data center could be defined as follows: given a workload set $Job$ and a data center $DataCenter$, find an optimal workload schedule, $Schedule$, which minimizes $T_{response}$ of the workload $Job$:

$$\min T_{response}$$

subject to:

$$\max_{1 \le i \le t} \{node_i.Temp(t)|T_0 \le t \le T_\infty\} \le Temp_{max}$$

(15)
node_i.Temp(t) = RC \times \frac{d \text{node}_i.Temp(t)}{dt} + \text{Temp}(\text{node}_i, <x,y,z>, t) - RP

(5)

\begin{align*}
\text{node}_i.Temp(t) &= PR + \text{Temp}(\text{node}_i, <x,y,z>, 0) + (\text{node}_i.Temp(0) - PR - \text{Temp}(\text{node}_i, <x,y,z>, 0)) \times e^{-\frac{t}{\text{therm}}}
\end{align*}

(6)

\begin{align*}
\text{node}_i.Temp(t) &= \text{node}_i.Temp(0) + \text{job}_j.\Delta\text{Temp}(t) -
\{ PR + \text{Temp}(\text{node}_i, <x,y,z>, 0) + (\text{node}_i.Temp(0) - PR - \text{Temp}(\text{node}_i, <x,y,z>, 0)) \times e^{-\frac{t}{\text{therm}}} \}
\end{align*}

(9)

It has been discussed in our previous work [?] that research issue of Eq. 15 is an NP-hard problem. Section IV then presents a scheduling heuristic, which is thermal aware scheduling algorithm with backfilling to solve this research issue.

IV. THERMAL AWARE WORKLOAD SCHEDULING ALGORITHM WITH BACKFILLING

This section discusses our Thermal Aware Scheduling Algorithm with Backfilling (TASA-B). The TASA-B is implemented in a data center scheduler and is executed periodically. Normally a scheduling algorithm contains two stages: (1) job sorting and (2) resource allocation. The TASA-B assumes incoming jobs have been sorted with certain system predefined priorities. Then the TASA-B only focuses the resource allocation stage. The TASA-B is an online scheduling algorithm, which periodically updates resource information as well as the data center thermal map from via temperature sensors.

The key idea of TASA-B is to schedule “hot” jobs on “cold” compute nodes and tries to backfill jobs on available nodes without affecting previously scheduled jobs. Based on (1) temperatures of ambient environment and compute nodes which can be obtained from temperature sensors, and (2) online job-temperature profiles, the compute node temperature after job execution can be predicated with Eq. 5. Online TASA-B algorithm schedules jobs based on the temperature prediction.

Algorithm 1 presents the TASA-B. Lines 1 – 4 initialize variables. Line 1 sets the initial time stamp to 0. Lines 2 – 4 set compute nodes available time to 0, which means all nodes are available from the beginning. extranodeset is the nodes that can be backfilled. Line 5 initializes it as empty set.

Lines 6 – 27, of Algorithm 1 schedule jobs periodically with an interval of Tinterval. Lines 6 and 7 update thermal map TherMap and current temperatures of all nodes from the input ambient sensors and on-board sensors. Then, line 8 sorts all jobs with decreased job_j.\Delta\text{Temp}(job_j, t^{req}) jobs are sorted from “hottest” to “coolest”. Line 9 sorts all nodes with increasing node temperature at the next available time, node_i.Temp(node_i, t^a); nodes are sorted from “coolest” to “hottest” when nodes are available.

Lines 10 – 16 cool down the over-heated compute nodes. If a node’s temperature is higher than a pre-defined temperature TEMP\text{max}, then the node is cooled for a period of Tcool. During the period of Tcool, there is no job scheduled on this node. This node is then inserted into the sorted node list, which keeps the increased node temperature at next available time.

Lines 17 – 24 allocate jobs to all compute nodes. Related research [?] indicates that, based on the standard model for the microprocessor thermal behavior, for any two tasks, scheduling the “hotter” job before the “cooler” one, results in a lower final temperature. Therefore, Line 17 gets a job from sorted jobs list to schedule. This job is the “hottest” job meaning the job has the largest task-temperature profile. Lines 18 – 20 try to schedule the job on some backfilled nodes with the backfilling algorithm 2 if the job only requires 1 processor. If the job

<table>
<thead>
<tr>
<th>Algorithm 1 Thermal Aware Scheduling Algorithm with Backfilling (TASA-B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
</tr>
<tr>
<td>02</td>
</tr>
<tr>
<td>03</td>
</tr>
<tr>
<td>04</td>
</tr>
<tr>
<td>05</td>
</tr>
<tr>
<td>06</td>
</tr>
<tr>
<td>07</td>
</tr>
<tr>
<td>08</td>
</tr>
<tr>
<td>09</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
<tr>
<td>17</td>
</tr>
<tr>
<td>18</td>
</tr>
<tr>
<td>19</td>
</tr>
<tr>
<td>20</td>
</tr>
<tr>
<td>21</td>
</tr>
<tr>
<td>22</td>
</tr>
<tr>
<td>23</td>
</tr>
<tr>
<td>24</td>
</tr>
<tr>
<td>25</td>
</tr>
<tr>
<td>26</td>
</tr>
<tr>
<td>27</td>
</tr>
</tbody>
</table>
requires more than 1 processor or the job cannot be scheduled on backfilled nodes. Lines 21 – 23 schedule the jobs on some nodes with thermal aware scheduling algorithm 3.

Algorithm 1 waits for a period of $T_{\text{interval}}$ and accepts incoming jobs. It then proceeds to the next scheduling round.

Algorithm 2 Thermal Aware Scheduling Algorithm for job $j$

```
1 get job, $j$ nodes, nodedset, from sorted Node set:
   nodedsetj ← {node1, node2, ..., nodep}
2 $t_0 ← \max\{node_k.t^a | node_k ∈ nodedsetj\}$
   node$\text{max}_1$ is the node which gives $\max\{node_k.t^a\}$
3 schedule job $j$ on nodedsetj
4 job.$j.t_{\text{start}} ← t_0$
5 $Temp^{bf}_{\text{max}} ← \max\{node_k.Temp(t^a) | node_k ∈ nodedsetj\}$
   node$\text{max}_2$ is the node which gives $\max\{node_k.Temp(t^a)\}$
6 FOR node $k ∈ nodedsetj \setminus \{node_{\text{max}_1}, node_{\text{max}_2}\}$
7 node.$k.bf ← \text{yes}$
8 node.$k.t^{bf\text{sta}} ← node_k.t^a$
9 node.$k.t^{bf\text{end}} ← t_0$
10 node.$k.Temp^{bf\text{sta}} ← node_k.Temp^{node_k.t^a}$
11 node.$k.Temp^{bf\text{end}} ← Temp^{bf}_{\text{max}}$
12 insert node$\text{k}$ into an extra node set, extranodeset
   keep the increased order of node$\text{k}.t^{bf\text{sta}}$.
13 ENDFOR
14 FOR node $k ∈ nodedsetj$
15 node.$k.t^a ← t_0 + job.j.t_{\text{req}}$
16 calculate node.$k.Temp(node_k.t^a)$ with Eq. 6 & 9
17 ENDFOR
18 insert every node in nodedsetj into Node
   keep the increased order of node.$i.Temp(node_i.t^a)$
   node$\text{i} ∈$ Node
```

Figure 6 shows the available backfilling nodeset on backfilled nodes, Lines 21 – 23 schedule the jobs on some nodes with thermal aware scheduling algorithm 3.

Algorithm 3 schedules job $j$ on some compute nodes with thermal considerations. Firstly, Algorithm 3 gets a set of nodes nodedset for job $j$, which are the coolest nodes. Line 2 locates the earliest available time $t_0$ of all nodes in nodedsetj.

Then, job $j$ is scheduled on the node set nodedsetj and the job starting time is set as the latest available time of nodes in nodedsetj, which is $t_0$ of node$\text{max}$ (Line 3 – 4).

Line 5 calculates the maximum temperature at time $t_0$ for all nodes in nodedsetj, which is used for later backfilling algorithms. For example, if some jobs want to be backfilled on these nodes, these jobs should not delay the execution of any previously scheduled jobs, and should not affect the maximum node thermal profiles, which were used to schedule previous jobs.

Line 6 – 13 update compute node information for backfilling. We use a conservative backfilling algorithm, which means that all previous scheduled jobs should not be affected. Therefore, the temperature profiles and available profiles that determine the schedule of job $j$ should not be affected. In specific, $t_0$ and Temp$_{\text{bf}}t_{\text{max}}$ should be changed by the backfilled jobs. Figure 6 and Figure 7 show the available backfilling periods for nodedsetj. Then node$\text{max}_1$ and node$\text{max}_2$ cannot be backfilled as they have no available time or temperature slot to accommodate any backfilled jobs.

Line 7 sets nodes in nodedsetj except node$\text{max}_1$ & node$\text{max}_2$ can be backfilled. Line 8 sets the starting time for backfilling of these nodes, which is the current available time for the nodes. Line 9 sets the end time for backfilling of these nodes, which is the earliest starting time of the job, job$\text{j}$, determined by the node$\text{max}$. Line 10 sets the start temperature for backfilling of these nodes, which are the temperatures of nodes at time of current available time. Line 11 sets the upper temperature limit for backfilling. If some jobs are backfilled on these nodes, the node temperature should not exceed the node temperature limit $Temp^{bf}_{\text{max}}$, which was used to determine the which nodes are allocated for the job job$\text{j}$. Finally, line 12 organizes all compute nodes for backfilling into a node set, which is sorted by the available backfilling starting time.

```
Fig. 6. Available time slots of thermal aware backfilling algorithm
```

Algorithm 3 schedules the job$\text{j}$ with backfilling. To decrease the algorithm complexity, we only consider the jobs that require one processor. Algorithm 3 checks all nodes in extranodeset, which are the nodes that are sorted by increased available time for backfilling. If the end time for backfilling of a node has passed, this node cannot be backfilled and should be removed from extranodeset (Line 2 – 6). Line 7 adjusts the start time for backfilling if it has passed. Line 8
Algorithm 3 Thermal Aware Backfilling Algorithm for job\(_j\)

1. FOR node\(_k\) ∈ extranodeset DO
2. IF node\(_k\).t\(_{bfend}\) < walltime THEN
3. node\(_k\).bf ← no
4. remove node\(_k\) from extranodeset
5. go to node\(_k\) + 1
6. ENDIF
7. node\(_k\).t\(_{bfsta}\) ← max{walltime, node\(_k\).t\(_{bfsta}\)}
8. IF node\(_k\).t\(_{bfend}\) − node\(_k\).t\(_{bfsta}\) ≥ job\(_j\).t\(_req\) and node\(_k\).Temp(node\(_k\).t\(_bfsta\) + job\(_j\).t\(_req\)) ≤ node\(_k\).Temp\(_{bfend}\) THEN
9. schedule job\(_j\) on node\(_k\)
10. job\(_j\).t\(_start\) ← node\(_k\).t\(_bfsta\)
11. node\(_k\).t\(_bfsta\) ← job\(_j\).t\(_start\) + job\(_j\).t\(_req\)
12. node\(_k\).Temp\(_{bfsta}\) ← node\(_k\).Temp(job\(_j\).t\(_start\) + job\(_j\).t\(_req\)), which is calculated with Eq. 6 & 9
13. ENDIF
14. IF node\(_k\).t\(_bfsta\) ≥ node\(_k\).t\(_bfend\) THEN
15. node\(_k\).bf ← no
16. remove node\(_k\) from extranodeset
17. ENDFOR

checks whether the the node’s time slot and temperature slot can accommodate the job job\(_j\). If job\(_j\) can be backfilled on node\(_k\) (Line 10), then the backfilling information of node\(_k\) are updated (Line 11, 12). Line 14 – 15 check whether node\(_k\) can be further backfilled.

V. SIMULATION AND PERFORMANCE EVALUATION

A. Simulation environment

We simulate a real data center environment based on the Center for Computational Research (CCR) of State University of New York at Buffalo. All jobs submitted to CCR are logged during a 30-day period, from 20 Feb. 2009 to 22 Mar. 2009. CCR’s resources and job logs are used as input for our simulation of the Thermal Aware Scheduling Algorithm with Backfilling (TASA-B).

CCR’s computing facilities include a Dell x86 64 Linux Cluster consisting of 1056 Dell PowerEdge SC1425 nodes, each of which has two Irwindale processors (2MB of L2 cache, either 3.0GHz or 3.2GHz) and varying amounts of main memory. The peak performance of this cluster is over 13 TFlop/s.

The CCR cluster has a single job queue for incoming jobs. All jobs are scheduled with a First Come First Serve (FCFS) policy, which means incoming jobs are allocated to the first available resources. There were 22385 jobs submitted to CCR during the period from 20 Feb. 2009 to 22 Mar. 2009. Figure 8, Figure 9 and Figure 10 show the distribution of job execution time, job size (required processor number) and job arrival rate in the log. We can see that 79% jobs are executed on one processor and job execution time ranges from several minutes to several hours.

Figure 1 shows the task-temperature profiles in CCR. In the simulation, we take the all 22385 jobs in the log as input for the workload module of TASA-B. We also measure the temperatures of all computer nodes and ambient environment with off-board sensors. Therefore the thermal map of data centers and job-temperature profiles are available. Online temperatures of all computer nodes can also be accessed from CCR web portal.

In the following section, we simulate the Thermal Aware Scheduling Algorithm (TASA-B) based on the job-temperature profile, job information, thermal maps, and resource information obtained in CCR log files. We evaluate the thermal aware scheduling algorithm by comparing it with the original job execution information logged in the CCR, which is scheduled by FCFS. In the simulation of TASA-B, we set the maximum temperature threshold to 125 °F.

B. Experiment results and discussion

1) Data center temperature: Firstly, we consider the maximum temperature in a data center as it correlates with the cooling system operating level. We use ∇Temp\(_{max}\) to show the maximum temperature reduced by TASA-B.

∇Temp\(_{max}\) = Temp\(_{max}\)\(_{fcfs}\) − Temp\(_{max}\)\(_{tasa-b}\) (17)

where, Temp\(_{max}\)\(_{fcfs}\) is the maximum temperature in a data center where FCFS is employed, and Temp\(_{max}\)\(_{tasa-b}\) is the maximum temperature in a data center where TASA-B is employed.

In the simulation we got ∇Temp\(_{max}\) = 4.9 °F. Therefore, TASA-B reduces 4.9 °F of the average maximum temperatures of the 30-day period in CCR.

We also consider the average temperatures in a data center, which relates the system reliability. In Figure 13 the red line shows the average ambient temperatures of all compute nodes, which is scheduled by TASA-B and blue line shows the average temperatures of all nodes in the log files, which were scheduled by FCFS. Compared with FCFS, the average temperature reduced by TASA-B is 14.6 °F.

2) Job response time: We have reduced power consumption and have increase the system reliability, both by decreasing the data center temperatures. However, we must consider that there may be trade off by an increased response time.

The response time of a job job\(_j\).t\(_{res}\) is defined as job execution time (job\(_j\).t\(_{req}\)) plus job queueing time (job\(_j\).t\(_{start}\) − job\(_j\).t\(_{arrive}\)), as shown below:

job\(_j\).t\(_{res}\) = job\(_j\).t\(_{req}\) + job\(_j\).t\(_{start}\) − job\(_j\).t\(_{arrive}\) (18)

To evaluate the algorithm from the view point of users, job response time indicates how long it takes for job results to return to the users.

As the thermal aware scheduling algorithm intends to delay scheduling jobs to some over-hot compute nodes, it may increase the job response time. Figure 11 shows the response time of the FCFS and Figure 12 shows the response time of the TASA-B.

In the simulation we calculate the overall job response time overhead as follows:

overhead = \sum_{1 \leq j \leq J} \frac{job\(_j\).t\(_{res}\)_{tasa-b} − job\(_j\).t\(_{res}\)_{fcfs}}{job\(_j\).t\(_{res}\)_{fcfs}} (19)
In the simulation, we obtained the overhead = 11%, which means that we reduced the maximum temperature by 4.9 °F and the average temperature by 14.6 °F in CCR’s data center by paying cost of increasing 11% job response time.

VI. CONCLUSION

With the advent of Cloud computing, data centers are becoming more important in modern cyberinfrastructures for high performance computing. However current data centers can consume a cities worth of power during operation, due to factors such as resource power supply, cooling system supply, and other maintenance. This produces CO$_2$ emissions and significantly contributes to the growing environmental issue of Global Warming. Green computing, a new trend for high-end computing, attempts to alleviate this problem by delivering both high performance and reduced power consumption, effectively maximizing total system efficiency.

Currently power supply for cooling systems can occupy up to 40% – 50% of total power consumption in a data center. This paper presents a thermal aware scheduling algorithm for data centers to reduce the temperatures in data center. Specifically, we present the design and implementation of an efficient scheduling algorithm to allocate workloads based on their task-termpatue profiles and find suitable resources for their execution. The algorithm is studied through a simulation based on real operation information from CCR’s data center. Test results and a performance discussion justify the design and implementation of the thermal aware scheduling algorithm.

REFERENCES

Fig. 13. Average temperature of all compute nodes


