2011 IEEE/ACM International Conference on Green Computing and Communications

Temperature, Power, and Makespan Aware Dependent Task Scheduling for Data Centers

Zheng Li, Li Wang, and Shangping Ren
Department of Computer Science
Illinois Institute of Technology
Chicago, Illinois 60616
Email: {zli80, lwang64, ren}@iit.edu

Gang Quan
Electrical and Computer Engineering Department
Florida International University
Miami, FL, 33174
Email: gang.quan@fiu.edu

***Abstract*—High performance computing data centers are playing increasingly important roles in our daily life. However, as data centers increase in size and number, the power consumption at the data centers has also increased dramatically. We are facing the challenge of reducing energy consumption, lowering down the peak inlet temperature and at the same time meeting short makespan requirements. In this paper, we present two dependent task scheduling algorithms to balance the trade-offs among data center's power consumption, peak inlet temperature, and application's makespan. We compare them with two existing algorithms, i.e., the *List* algorithm and the *CoolestInlets* algorithms. Our extensive simulations show clear advantages of the proposed approaches over the *List* and the *CoolestInlets* algorithms for both homogeneous and heterogeneous data centers.**

***Keywords*-Energy Consumption, Peak Inlet Temperature, Task Dependency, Application Makespan, Scheduling Algorithm**

I. Introduction

High Performance Computing (HPC) data centers are playing more and more important roles in our daily life, ranging from modeling financial markets to forecasting weather and nature disasters. In recent years, their sizes have grown dramatically, however, the power consumption has also increased severely. And in large data centers, over 50% of the power consumed is *not* used for executing tasks, rather, it is used for cooling the centers [1], [2].

Bash [3] pointed out that heat recirculation often increases the inlet temperature and causes hot spots in data centers. In order to prevent data servers from being overheated and ensure that the temperature at each data server is below the *red line* temperature, the computer room air conditioners are used to supply cold air and remove heat from the center. The supplied air must be cold enough so that the hottest server can be quickly cooled down. Clearly, the higher the temperature of a server, the lower the temperature of cooling air provided by the air conditioner has to be. Unfortunately, as pointed out by Moore [4], lowing the room air conditioner's working temperature degrades its capability of removing heat from the room. Therefore, if we can reduce the peak inlet temperature, the air conditioner can work at a higher temperature with better performance on heat removal, and hence lower down the cooling cost.

Researchers have been exploring different approaches to reduce the cooling cost. Moore et al. [4] has defined heat-recirculation-factor and used it to guide the power distribution on each server so that minimal peak inlet temperature can be achieved, however, which is not a task scheduling algorithm. Tang et al. [5] extended the work and provided modify-MinHR and XInt-GA algorithms to minimize the peak inlet temperature, but the assumption is that servers in the data centers are homogeneous, tasks are independent and application's makespan is not a concern.

In this paper, we investigate how to schedule dependent tasks in both homogeneous and heterogeneous data centers to achieve the following objectives:

1) lower the peak inlet temperature,
2) reduce energy consumption for executing tasks, and
3) shorten application's makespan.

It is not difficult to see that these objectives conflict with each other. For instance, a powerful server may finish a task in a very short time period, but has high energy cost. Therefore, frequently using this server reduces applications' markspan, but may cause the server to be a hot spot. This paper presents two multi-phased heuristic task scheduling algorithms that take an application's makespan and its execution energy consumption into consideration while trying to minimize a data center's peak inlet temperature.

The rest of the paper is organized as follows: Section II introduces the system model. Section III presents two task scheduling algorithms that balance the trade-offs among data center's peak inlet temperature, power consumption, and application's makespan. The theoretical analysis about the proposed algorithms is discussed in Section IV. Section V discusses simulation results. We conclude the paper and point out future work in Section VI.

II. Models and Assumptions

Task Model

We assume that an application consists a set of dependent tasks. The dependency relationship among tasks is modeled by a directed acyclic graph [6] $G = (V, E)$, where each vertex in the graph represents a task, and dependency among tasks are represented by an edge between corresponding task nodes. If $(V_i, V_k) \in E$, i.e., task V_k depends on task

978-0-7695-4466-3/11 $26.00 © 2011 IEEE
DOI 10.1109/GreenCom.2011.12

V_i. When task k is assigned to server u, then the task k's finishing time can be obtained as:

$$F_{k,u} = S_{k,u} + e_{k,u} \tag{1}$$

where, $e_{k,u}$ is execution time of task k on server u and $S_{k,u}$ is the starting time of task k on server u. Further more, we assume that task executions are nonpreemptive and each task is large and complex enough that it fully utilizes the server it is assigned to and the execution time is long enough cause the server to reach certain thermal state[4].

Inlet Temperature Model

Based on [7], the inlet temperature is the summation of supplied temperature provided by air conditioners and caused by executing tasks, which is given as:

$$\mathbb{T}_{inlet} = \mathbb{T}_{sup} + \mathbb{D} \times \mathbb{P}_{exe} \tag{2}$$

where

$$\mathbb{D} = [(\mathbb{K} - \mathbb{A}^T k)^{-1} - \mathbb{K}^{-1}] \tag{3}$$

and $\mathbb{T}_{sup} = [T^u_{sup} | u = 1, 2, ..., S]^T$, T^u_{sup} is the supplied air temperature for server u by air conditioners; $\mathbb{P}_{exe} = [P^u_{exe} | u = 1, 2, ..., S]^T$, P^u_{exe} is server u's power consumption for executing the assigned tasks; $\mathbb{A} = [a_{u,l}]_{S \times S}$, $a_{u,l}$ is the heat cross interference coefficient between server u and l, $\mathbb{K} = diag(k_1, \ldots, k_s)$, $k_i = \rho f_i C_p$ is server i's thermodynamic factor, f_i is the air flow rate around the server, C_p is the heat carried by unit of air, and ρ is the air density.

Power Consumption Model

For data centers, tasks execution and heat cooling are the two power consumption entities. Given server u's energy consumption rate R_u and the execution time of task k on server u, $e_{k,u}$ ($e_{k,u} = 0$, if task k is not assigned to server u), the total power consumption P_{exe} used for task execution is:

$$P_{exe} = \sum_{k=1}^{N} \sum_{u=1}^{S} e_{k,u} R_u \tag{4}$$

Where N is the number of tasks, and S is the number of servers in the data center. We assume the energy cost for communication between different servers is negligible.

The power consumption caused by room air conditioners for cooling purpose depends on the air conditioners' supplied temperature. Based on Moore [4], we have:

$$P_{cool} = \frac{P_{exe}}{CoP(T_{sup})} \tag{5}$$

Where CoP is the air conditioner's performance coefficient defined as:

$$CoP(T_{sup}) = 0.0068 T^2_{sup} + 0.0008 T_{sup} + 0.458 \tag{6}$$

From (5) and (6), it is clear that the cooling cost can be reduced by increasing the air temperature supplied by the air conditioners. Furthermore, if the peak inlet temperature T_{peak} is below *red line* temperature T_{red}, we can increase T_{sup} to T'_{sup} [5]:

$$T'_{sup} = T_{sup} + T_{red} - T_{peak} \tag{7}$$

From formula (5) and (6), we can conclude the lower the peak inlet temperature, the less power cost for cooling.

III. Balance Power Consumption, Temperature and Application's Makespan

Before presenting our approaches, we first introduce two algorithms that are used for task schedulings, i.e., the *List* [8], [9], [10], and the *CoolestInlets* [4] algorithms.

The optimization goal for the *List* algorithm is to minimize the makespan of a given application. Hence, at every scheduling point, *List* algorithm always chooses the server that offers the earliest finishing time. The *CoolestInlets* algorithm, on the other hand, ignores the makespan constraint, and tries to minimize the peak inlet temperature in the data center. Therefore, its scheduling policy is always allocating the incoming task to the currently coldest server. Clearly, when inlet temperature, computing energy consumption, and application's makespan are all taken into consideration, neither of these approaches works well.

A. Power, Temperature, and Makespan Aware Dependant Task Scheduling Algorithm

In order to minimize the peak inlet temperature, intuitively, we need to avoid using the severs that have large heat cross interference coefficients. Moore [4] defines a heat-recirculation-factor (HRF) rule to guide power distribution within the data center to minimize the peak inlet temperature. More specially, if the total computing power consumed by all the severs is P_{exe}, then the minimum peak inlet temperature is achieved when server u's computing energy $p(u)$, follows the following formula:

$$p(u) = \frac{HRF(u)}{\sum_{l=1}^{S} HRF(l)} \times P_{exe} \tag{8}$$

Where S is the number of servers, and $HRF(u)$ is the heat recirculation factor of server u, which is the physical property of the server describing its contribution for heat recirculation within the data center. The HRF measurements for each server are given in Tang [5] and Moore [4].

As we know, makespan can be shortened by reducing each task's earliest finishing time (i.e., latency), computing energy cost can be reduced by choosing the server with less computing power consumption, and peak inlet temperature can be minimized when the power distribution meets the HRF rule. Unfortunately, these constraints conflict with each other and can not be met simultaneously. Therefore, we need to find a better compromise.

For our concerns, lowering the peak inlet temperature is the first priory. At the same time, we also make effort to save the computing energy cost and reduce the application's makespan. By placing the later two concerns in different priorities, we propose two algorithms, i.e., temperature-computation-energy-latency (TEL) algorithm

and temperature-latency-computation-energy (TLE) algorithm. For the TEL algorithm, we set saving the P_{exe} as the second priority, and then reducing the makespan; While, which in TEL algorithm have the reversed orders.

B. The TEL and TLE Algorithm

The basic strategy behind the TEL algorithm is to let the servers consume the power based on formula (8) to guarantee the minimum peak inlet temperature. So we first estimate the total computing energy consumption, then use HRF rule to distribute the power to each server, we call it the reserved power, The reserved power is used to guide the tasks assignment. The general rule is trying to assign each incoming task to a server that has sufficient reserved power. If no such server exists, select one that needs the minimal extra power. Under this rule, we adjust the scheduling policy to save the computing energy cost and shorten the makespan. The algorithm is implemented in the following steps:

Step 1: Estimate P_{exe}: Using the $List$ algorithm to do the tasks allocation, then add all the computing energy cost to obtain P_{exe}.

Step 2: Compute power distribution: Compute and reserve $p(u)$ for server u using formula (8).

Step 3: Tasks assignment: Using $List$ algorithm to re-schedule each incoming task, if the selected server u has enough reserved power, assign the task to it. Otherwise, choose another one according to the following policy:

1) Choose the server with enough reserved power, breaking the ties by selecting the one that has the minimum computing power cost and breaking further ties by earliest finishing time.
2) If no server has enough reserved power, select the one that needs minimal extra power, breaking the ties by minimal computing energy cost and breaking further ties by earliest finishing time. In addition, in order to let the power distribution within the data center match the HRF rule, we need to add extra reserved power to other servers according to their HRF factors.

Algorithm 1 gives the details for the TEL algorithm.

Where $M = \{u|u = 1, 2, ..., S\}$, and $p_{i,u} = e_{i,u} \times R_u$, that is, the computing energy cost for task i on server u. $ts = \{ts(i)|i = 1, 2, ...N\}$, and $ts(i)$ stores the server that the task i will be assigned to. To be more specific, in algorithm 1, line 1 corresponds to step 1 and line $2-4$ corresponds to step 2, line $5-26$ corresponds to step 3, where line $6-8$ is re-scheduling using List, line $9-13$ is the policy 1 and line $14-26$ is the policy 2.

When makespan has higher priority than computing energy cost, the computing orders of line 12 in algorithm 1 should be reversed, that is TLE algorithm. And algorithm 2 gives the details.

Algorithm 1 TEL (S, N, $HRF(u)$, $e_{i,u}$, R_u, M, ts)

```
1: task scheduling with List algorithm to get P_exe
2: for i = 1 to S do
3:    compute p(i) using formula (8)
4: end for
5: for i = 1 to N do
6:    schedule the task i using P_exe and List to server u
7:    if p(u) ≥ p_{i,u} then
8:       p(u) = p(u) − p_{i,u}; ts(i) = u
9:    else
10:      M_T = {k|p(k) ≥ p_{i,k}, k ∈ M}
11:      if |M_T| > 0 then
12:         Compute M_ET and M_LET using the formulas:
    1)      M_ET = {k|p_{i,k} = min{p_{i,k}, k ∈ M_T}}
    2)      M_LET = {k|F_{i,k} = min{F_{i,k}, k ∈ M_ET}}
13:         ts(i) = M_LET(1); p(ts(i)) = p(ts(i)) − p_{i,ts(i)}
14:      else
15: M_T = {k|(p_{i,k} − p(k)) = min{(p_{i,k} − p(k)), k ∈ M}}
16:         Compute M_ET and M_LET as step 12.
17:         ts(i) = M_LET(1)
18:         for v = 1 to S do
19:            if v ≠ ts(i) then
20:               p(v) = p(v) + |p_{i,ts(i)} − p(ts(i))| ×
                  HRF(v)/HRF(ts(i))
21:            end if
22:         end for
23:         p(ts(i)) = 0
24:      end if
25:   end if
26: end for
27: return ts
```

Algorithm 2 TLE (S, N, $HRF(u)$, $e_{i,u}$, R_u, M, ts)

```
.....
12: Compute M_LT and M_ELT by using the formulas:
  1)  M_LT = {k|F_{i,k} = min{F_{i,k}, k ∈ M_T}}
  2)  M_ELT = {k|p_{i,k} = min{p_{i,k}, k ∈ M_LT}}
13: ts(i) = M_ELT(1)
.....
16: Compute M_LT and M_ELT as step 12.
17: ts(i) = M_ELT(1)
.....
```

C. Algorithm Complexity

For step 1, we use the $List$ algorithm to obtain the reserved power of each server, the time cost is $O(NS)$, where N and S are the number of tasks and servers, respectively. For step 2, computing reserved power for S servers, so time cost is $O(S)$. In step 3, for each task, if it can be scheduled by $List$ algorithm(line 6-8), the time cost is $O(S)$, otherwise, it will be assigned to a server according to policy 1)(line 9 - 13) or policy 2) (line 14 - 26), both of which cost $O(S)$ time. So for N tasks, the time complexity is $O(NS)$, which is the same for TLE and TEL algorithms.

IV. Theoretical Analysis

Now, we formally analyze our proposed algorithms. A few definitions are introduced firstly to simplify the discussion.

Definition 1: Energy Consumption Heterogeneity Factor (ECHF) $ECHF$ represents energy consumption rate variation within the data center, which is given below:

$$ECHF = \sqrt{\frac{max\{R_u | u \in M\}}{min\{R_u | u \in M\}}} \tag{9}$$

Definition 2: Task Execution Heterogeneous Factor (TEHF) $TEHF$ represents tasks execution time variation and is given as:

$$TEHF = \frac{1}{N} \sum_{j=1}^{N} \sqrt{\frac{max\{e_{j,u} | u \in M\}}{min\{e_{j,u} | u \in M\}}} \tag{10}$$

Definition 3: Temperature cumulative distribution function (TCDF) $TCDF(temp)$ is a function of $temp$ that, among all the servers, what percentage of which have the temperature not higher than the value of $temp$.

$$TCDF(temp) = \frac{|\{u | T^u_{inlet} \leq temp, u \in M\}|}{S} \tag{11}$$

Where T^u_{inlet} is the inlet temperature of server u.

A. Homogeneous Data Center

In homogeneous data center, all the servers have the same energy consumption rate and task execution rate.

Lemma 4.1: In homogeneous data center, energy consumed by executing the task is independent of the server which is assigned to.

Proof: This lemma is derived directly from the homogeneity property. ■

Lemma 4.2: In homogeneous data center, for a given task graph, the total computing energy consumption is a constant value in spite of different task allocation algorithms.

Proof: According to Lemma 4.1, the computing energy cost is constant for a specific task, and also, the number of tasks is fixed. So the total computing power cost is a constant value and independent of tasks allocation algorithms. ■

Lemma 4.3: In homogeneous data center, TLE and TEL algorithms generate the same task schedule.

Proof: According to Lemma 4.1, we cannot break the ties by minimum computing power cost, so the computing steps of both algorithms are the same. ■

According to Lemma 4.2, the total computing energy consumption is independent of tasks scheduling algorithm. As the power distributions for TEL/TLE algorithm are set by HRF rule, so the peak inlet temperature can be minimized.

B. Heterogeneous Data Center

In heterogeneous data centers, neither the task execution rate nor energy consumption rate are unified.

As HRF rule is independent of system architecture, both TEL and TLE algorithms can minimize the peak inlet temperature. By setting different priorities for saving computing energy cost and minimizing the makespan, the TEL saves more computing energy cost, while TLE offers shorter makespan. If in a deep heterogeneous system, i.e., both $ECHF$ and $TEHF$ are very large, this phenomenon becomes more obvious. The intuitive explanation is that short makespan requires short execution time, which results in less computing energy cost. However, when the variances are larger, the above relationship becomes weaker, hence the difference between TEL and TLE becomes more obvious.

V. Simulation Results and Discussions

In our experiments, we compare the proposed two algorithms: TEL and TLE with $List$ and $CoolestInlets$ in the aforementioned three facets: peak inlet temperature, computing energy cost and makespan.

A. Simulation Setup

We adopt the value of Cross Interference Coefficient ($a_{u,l}$) from Tang's experiment [5]. The data center has 10 racks, each rack is equipped with 10 servers. The $HRF(u)$ [5] is the heat recirculation factor for server u, in our simulation, we define it as:

$$HRF(u) = \frac{1}{\sum_{l=1}^{S} a_{u,l}} \tag{12}$$

We also use TGFF [11] to generate the dependent task graph including 356 tasks.

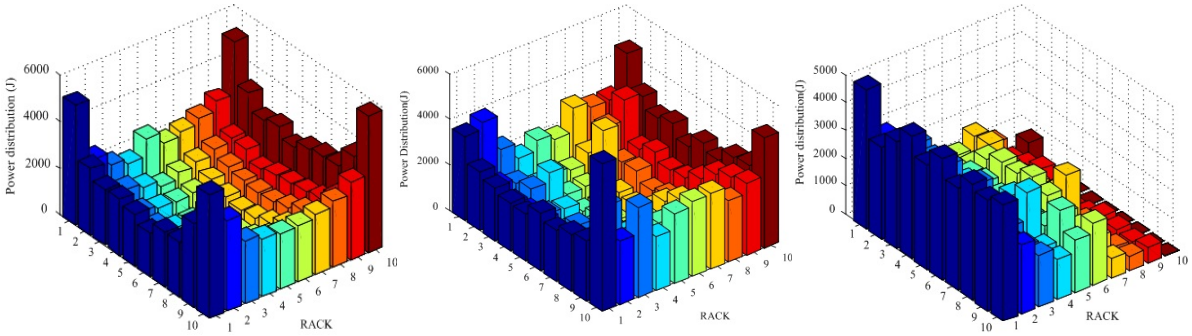
B. Homogeneous Data Center

For a given task graph, the computing energy cost P_{exe} is a constant value (Lemma 4.2). So we only need to compare the peak inlet temperature and makespan. Further more, based on Lemma 4.3, TEL and TLE converge to the same result, so in this section, we use TEL to represent the two.

Figure 1 shows the power distribution for all the aforementioned algorithms. That of TEL algorithm has the maximum similarity as the ideal one(Figure 2b). From the $TCDF$ comparison(Figure 2a), we can see that TEL has the minimum peak inlet temperature, which is about 1.5^oC and over 5^oC lower than *CoolestInlets* and *List*, respectively.

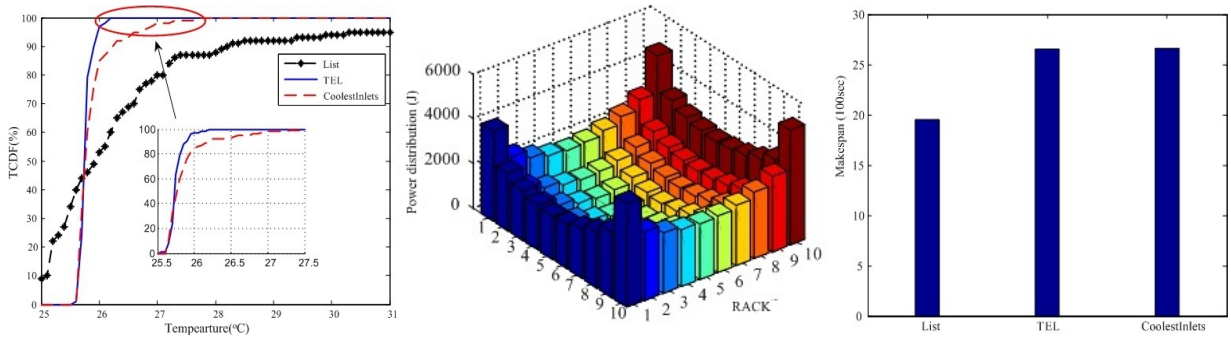
The reason is, *List* may cause hot spots as some servers are frequently used, so their inlet temperatures will be much higher than that of others. For *CoolestInlets*, generally, it can balance the heat within the data center, while, it can still happen to the scenario that the current coldest server is inefficient for the incoming task and result in the inlet temperature increasing rapidly.

By using the HRF rule to guide the task scheduling, TEL algorithm avoids stepping into the above mentioned snares, so the inlet temperatures can be evenly distributed among all the servers.



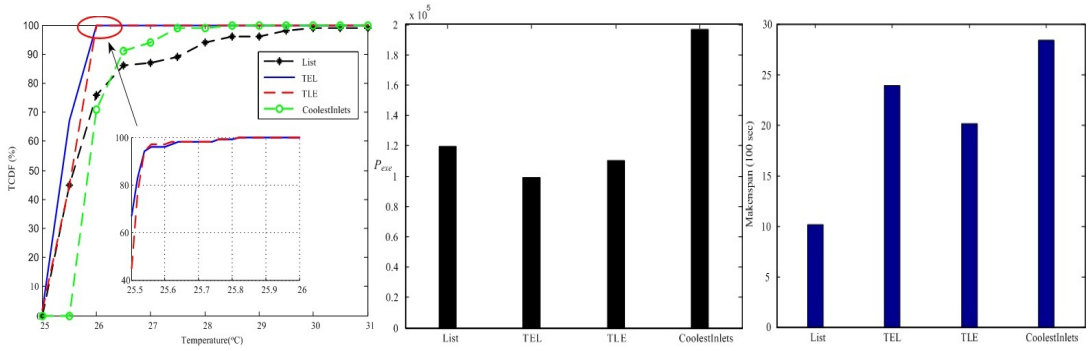
(a) Power distribution for TEL/TLE (b) Power distribution for CoolestInlets (c) Power distribution for List

Figure 1: Power distribution comparison in homogeneous system



(a) TCDF comparison (b) Ideal power distribution (c) Makespan comparison

Figure 2: Simulation result for homogeneous system



(a) $TCDF$ comparison (b) Total computing power cost (c) Makespans comparison

Figure 3: Simulation result for $TEHF$ = 2.0, $ECHF$ = 2.0

Figure 2c gives the makespan comparison. In order to achieve the lowest peak inlet temperature, *TEL* sacrifices some performance of makespan, which is about 600sec longer than that of *List* and nearly the same as that of *CoolestInlets*.

C. Heterogeneous Data Center

The advantages of TEL and TLE are more obvious in the heterogeneous data center.

Figure 3a gives the $TCDF$ comparison in homogeneous data center, as reducing peak inlet temperature is the first priority for TEL and TLE, they have nearly the same peak inlet temperature, which is about 2^oC and 4.5^oC lower than that of *CoolestInlets* and *List*, respectively. Figure 3b shows the comparison for computing energy cost. TEL needs the least computing power, which of *List* is about $10000J$ more than that of TLE, and *CoolestInlets* consumes the most. Figure 3c gives the makespan comparison, although which of TLE is about two times as long as that of *List*, it is still 300sec and 600sec shorter than TEL and *CoolestInlets*.

When the heterogeneous factors become larger, both *List* and *CoolestInlets* behave even worse. However, the variation of heterogeneous factors have little impact for TEL and TLE, their peak inlet temperatures grow very slowly, both of which are about over 5^oC lower than that of *List* and *CoolestInlets* (Figure 4).

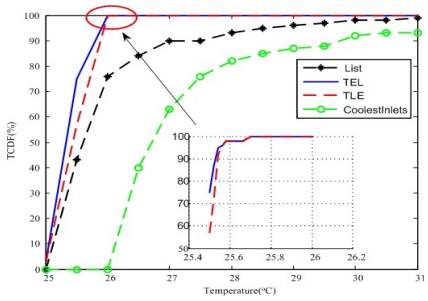


Figure 4: TCDF comparison ($TEHF$ = 4.0, $ECHF$ = 4.0)

VI. Conclusion

The paper addresses the problem of scheduling dependent tasks in data centers with multiples objectives, i.e., minimizing data center's peak inlet temperature, computing energy consumption and application's makespan. We present two task scheduling algorithms, i.e., the TEL and the TLE algorithms, to make appropriate trade offs based on the priorities of these three objectives. In particular, minimizing data center's peak inlet temperature is the first priority for both of the algorithms. The TEL algorithm takes the computing energy consumption reduction as the second priority and application's makespan as the third; while the TLE algorithm takes reversed orders. The TEL algorithm has good performance in lowering the peak inlet temperature and saving computation energy cost; while The TLE algorithm results in the same peak inlet temperature as the TEL algorithm, but with shorter makespan than the TLE algorithm by sacrificing certain computation energy.

Our future work will investigate how to schedule tasks with real-time constraint to servers in data center that minimizes peak inlet temperature and total energy consumption.

Acknowledgment

This research is supported in part by NSF under grants CNS 0746643, CNS 1018731, CNS 1035894, CNS 0969013, CNS 0917021 and CNS 1018108.

References

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

[2] R. F. Sullivan, "Alternating cold and hot aisles provides more reliable cooling for server farms," *White Paper, Uptime Institute*, 2000.

[3] C. Bash and G. Forman, "Cool job allocation: measuring the power savings of placing jobs at cooling-efficient locations in the data center," in *USENIX Annual Technical Conference on Proceedings of the USENIX Annual Technical Conference*, 2007, pp. 29:1–29:6.

[4] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making scheduling "cool": temperature-aware workload placement in data centers," in *Proceedings of the annual conference on USENIX Annual Technical Conference*, 2005, pp. 5–5.

[5] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, "Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, pp. 1458–1472, 2008.

[6] T. Xie and X. Qin, "A new allocation scheme for parallel applications with deadline and security constraints on clusters," in *Cluster Computing, IEEE International*, 2005, pp. 1 –10.

[7] Q. Tang, T. Mukherjee, S. K. S. Gupta, and P. Cayton, "Sensor-based fast thermal evaluation model for energy efficient high-performance datacenters," in *Proc. Fourth Int'l Conf. Intelligent Sensing and Information Processing*, 2006.

[8] T. Xie and X. Qin, "An energy-delay tunable task allocation strategy for collaborative applications in networked embedded systems," *IEEE Transactions on Computers*, vol. 57, pp. 329–343, 2008.

[9] Y.-K. Kwok and I. Ahmad, "Benchmarking the task graph scheduling algorithms," in *Proceedings of the 12th. International Parallel Processing Symposium on International Parallel Processing Symposium*. IEEE Computer Society, 1998, pp. 531–537.

[10] A. Rădulescu and A. J. C. van Gemund, "On the complexity of list scheduling algorithms for distributed-memory systems," in *Proceedings of the 13th international conference on Supercomputing*, 1999, pp. 68–75.

[11] R. Dick, D. Rhodes, and W. Wolf, "Tgff: task graphs for free," in *Hardware/Software Codesign, Proceedings of the Sixth International Workshop on*, Mar. 1998, pp. 97 –101.