Multi-Objective Scheduling for Heterogeneous Server Systems with Machine Placement

Hongyang Sun, Patricia Stolf, Jean-Marc Pierson, Georges Da Costa
IRIT, University of Toulouse, France
{sun, stolf, pierson, dacosta}@irit.fr

***Abstract*—Heterogeneous servers are becoming prevalent in many high-performance computing environments, including clusters and datacenters. In this paper, we consider multi-objective scheduling for heterogeneous server systems to optimize simultaneously the application performance, energy consumption and thermal imbalance. First, a greedy online framework is presented to allow the scheduling decisions to be made based on any well-defined cost function. To tackle the possibly conflicting objectives, we propose a fuzzy-based priority approach for exploring the tradeoffs of two or more objectives at the same time. Moreover, we present a heuristic algorithm for the static placement of physical machines in order to reduce the maximum temperature at the server outlets. Extensive simulations based on an emerging class of high-density server system have demonstrated the effectiveness of our proposed approach and heuristics in optimizing multiple objectives while achieving better thermal balance.**

***Keywords*-Multi-objective optimization; online scheduling; machine placement; job response time; energy consumption; thermal imbalance; tradeoffs; heterogeneous server systems**

I. Introduction

Server systems that consist of heterogeneous computing nodes are becoming prevalent in clusters and datacenters. In particular, many high-performance computing systems embrace machine heterogeneity in their designs, which is believed by many to be the key for achieving energy-proportional computing [3], [5]. While application scheduling in the heterogeneous environments has been an important area of research for decades, traditional approaches have mostly focused on application performance as the sole optimization criterion. In recent years, the high energy consumption has emerged as a major issue. Studies have shown that today's datacenters are consuming nearly 2% of the global energy [13], and up to half of that is spent on cooling-related activities [18]. Hence, the need for multi-objective schedulers that also consider the energy and cooling efficiency is imminent.

In this paper, we address the Multi-Objective Optimization Problem (MOOP) for Heterogeneous Server Systems (HSS). Besides the traditional objective of application performance, we also consider the energy consumption of the servers and the thermal imbalance as additional objectives. The latter has been particularly shown to have direct impacts on the efficiency and cost of cooling in datacenters [16], [21]. As these objectives can be conflicting with each other, the aim is to efficiently explore their tradeoffs, and if possible, to optimize two or more of them simultaneously. To systematically tackle this problem, we apply two complementary approaches, which are the placements of machines and applications, respectively.

While application placement (or scheduling) has been studied in the past to optimize the performance in HSS, machine placement has received less attention. The reason is that the traditional metric of job performance or even energy consumption is independent of the positions of the physical machines, and hence are not affected by different placement configurations. In the presence of heterogeneous servers, however, the placement of the machines can influence the distribution of the generated heat, which has been shown to play an important role in the cooling efficiency of datacenters [16], [21]. While it is not feasible to dynamically reconfigure the machines' positions based on the temporal variation of the workloads, we focus on static machine placement and propose a heuristic that minimizes the maximum temperature at the server outlets according to their heat distribution characteristics in the idle state. The strategy is effective due to the correlation between the machines' static and dynamic power consumptions.

The placement of the applications is also challenging. In general, jobs arrive in an online manner, thus any future knowledge is unknown to the scheduler. Unlike many previous results (e.g., [2], [22]) that only optimize the jobs' execution times, we perform online assignment of jobs to machines using a greedy framework, which allows the scheduling decisions to be made based on any well-defined cost function. To tackle multiple objectives at the same time, we propose a fuzzy-based priority approach that optimizes two objectives in sequence. A fuzzy factor is introduced to explore any potential improvement for the second objective while relaxing the first objective up to an acceptable range. The approach is flexible enough to incorporate multiple objectives, such as those obtained by weighted sums, into the optimization, and the principle can potentially be applied to other multi-objective optimization problems.

To evaluate our proposed approach, we model and simulate Christmann's Resource Efficient Cluster Server (RECS) [4], a heterogeneous server system with high packing density and integrated cooling support. The system represents an emerging class of high-performance and energy-efficient servers for racks and clusters in a typical datacenter environment. Using average job response time, dynamic energy consumption and maximum outlet temperature as three optimization objectives, the simulation results show the effectiveness of our fuzzy-based priority approach for exploring and optimizing the tradeoffs of two or more objectives. Our static machine placement heuristic is also shown to provide significantly better thermal balance at the server outlets in terms of both maximum and average values.

The rest of this paper is organized as follows. Section II reviews some related research in the field. Section III states the models and the problems. Section IV presents our machine placement heuristic. Section V describes the job scheduling heuristics and the fuzzy-based priority approach. The simulation results are presented in Section VI. Finally, Section VII concludes the paper with some future directions.

II. Related Work

Multi-objective optimization has attracted much attention in various problem domains. In the following, we describes some state-of-the-art approaches in this area.

First, combining multiple objectives into a single one is a popular approach. The authors in [15] used Dynamic Voltage & Frequency Scaling (DVFS) to tradeoff makespan with energy consumption by considering a weighted sum of the two objectives. In [20], the same technique was applied in an online manner to minimize a combined objective of job response time and energy. A similar approach was taken in [19], which considers an additional objective of peak temperature in a multicore system, and hence optimizing the weighted sum of three objectives at the same time.

Another approach is constrained optimization for one or more objectives. In [17], DVS was used to minimize the energy consumption subject to the makespan achieved in an initial schedule. A double strategy was developed in [8] to minimize the Euclidean distance between the generated solutions to a set of user-specified constraints for a four-objective optimization problem. The authors of [12] applied ϵ-constraint method to cloud scheduling, which optimizes each objective in turn with upper bounds on others.

Some research uses priority-based approaches to optimize multiple objectives in sequence. In [1], a bi-criteria compromise function was introduced to set priorities between makespan and reliability for scheduling real-time applications. To minimize carbon emission and to maximize profit, two-step policies were proposed in [11] to map applications to heterogeneous data centers based on the relative priority of the two objectives. In [6], the authors proposed heuristics to optimize the QoS for interactive services before considering energy consumption on DVS-enabled multicore systems.

Lastly, Pareto-based approach is often used in the offline setting to generate more than one non-dominant solutions. This technique was applied in [7] to tradeoff makespan and energy consumption for heterogeneous servers. Evolutionary algorithms were employed in [10] to obtain a set of alternative solutions for scheduling scientific workloads in the Grid environment. In [23], the authors applied particle swarm optimization to approximate the Pareto frontier for the unrelated machine scheduling problem with uncertain inputs.

In this paper, we present a fuzzy-based priority approach. Although multi-objective scheduling with "fuzzy" or "good enough" solutions [25], [26] are known in the Pareto-based approach, our approach is novel in the online setting, especially when different objectives have (soft) priorities. In the thermal-aware scheduling literature, workload placement has been considered in the presence of nonuniform heat distribution in datacenter environments [16], [21], but no prior work has addressed online scheduling for multiple objectives. We also consider the placement problem for physical machines to achieve better thermal balance in heterogeneous servers, which to our best knowledge has not been studied in the past.

III. Problem Statement

A. System Model

Motivated by scheduling high-performance computing applications in *Heterogeneous Server Systems (HSS)*, we consider the following system model. There is a set $\mathcal{M} = \{M_1, M_2, \cdots, M_m\}$ of m heterogeneous machines, which need to be placed inside a server system with m positions $\{C_1, C_2, \cdots, C_m\}$ and l outlets $\{L_1, L_2, \cdots, L_l\}$. Each machine $M_j \in \mathcal{M}$ has a static power consumption U_j^{stat} when it is idle or not executing any job. A set $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$ of n jobs arrive at the system in an online manner, and each job $J_i \in \mathcal{J}$ is characterized by a release time r_i, a processing time $P_{i,j}$ and a dynamic power consumption $U_{i,j}$ if it is executed on machine M_j. We study both job scheduling and machine placement for this model, which are described in the following.

1) Job Scheduling: The jobs are to be scheduled in an online manner to the machines. That is, each job needs to be assigned irrevocably to a machine without knowledge of future job arrivals. Moreover, once a job has been assigned, no preemption or migration is allowed, which usually incurs a significant cost in terms of data reallocation. The total power consumption of machine M_j at any time t is given by

$$U_j^{tot}(t) = U_j^{stat} + \sum_{i=1}^{n} \delta_{i,j}(t) \cdot U_{i,j} \ , \tag{1}$$

where $\delta_{i,j}(t)$ is a binary variable that takes value 1 if job J_i is running on machine M_j at time t and 0 otherwise. In order to optimize performance, we restrict that each machine can host only one job at any time. Thus, we have $\sum_{i=1}^{n} \delta_{i,j}(t) \leq 1$ for all $1 \leq j \leq m$ at all time t.

2) Machine Placement: The set of machines need to be statically placed in advance to the m positions in the server. Assuming that the cooling of the system gives a steady airflow pattern inside the server, the heat generated from each position to each outlet can be described by a *heat-distribution matrix* $\mathbf{D}$, where each element $d_{x,k} \in \mathbf{D}$ denotes the fraction of the heat generated from position C_x to outlet L_k. As some heat may be dissipated through other channels (such as small holes) of the server instead of the outlets or even stay in the server enclosure, we have $\sum_{k=1}^{l} d_{x,k} \leq 1$ for all $1 \leq x \leq m$.

This model is general enough to capture the situation of many heterogeneous server systems, such as those in racks or clusters of typical datacenters.

B. Scheduling Model

There are two subproblems: First, we need to decide a *static machine placement*, that is, to find a mapping $\pi : \{1, 2, , \cdots, m\} \rightarrow \{1, 2, , \cdots, m\}$ from server positions to machines so that each position C_x is filled with a machine $M_{\pi(x)}$. Then, we need an *online job scheduling* strategy to assign each arriving job to a machine for execution. Assuming that the power consumption is completely transformed into

heat, the total amount of heat (or power) received by outlet L_k at time t can be expressed as

$$U_k^{out}(t) = \sum_{x=1}^{m} d_{x,k} \cdot U_{\pi(x)}^{tot}(t) \ . \quad (2)$$

Given a constant temperature T^{in} at all inlets of the server, the temperature at outlet L_k is given by

$$T_k^{out}(t) = T^{in} + g(U_k^{out}(t)) \ , \quad (3)$$

where function $g(U)$ converts the amount of heat U (in $Watt$) received by the outlet to the increase in its air temperature. In general, the air temperature is a function of air density ρ (in kg/m^3), airflow throughput Q (in m^3/s), and air heat capacity C (in $Joule/(^oC \cdot kg)$) The following gives an expression of $g(U)$ in terms of these parameters [21]:

$$g(U) = \frac{U}{\rho \cdot Q \cdot C} \ . \quad (4)$$

C. Optimization Objectives

We consider the following *Multi-Objective Optimization Problem (MOOP)*: optimizing the performance of the jobs, minimizing the energy consumption of the machines, and balancing the temperatures at the server outlets.

For performance, we use the average response time R_{ave} of the jobs as the metric, and it is defined as

$$R_{ave} = \frac{1}{n} \sum_{i=1}^{n} (c_i - r_i) \ , \quad (5)$$

where c_i and r_i denote the completion time and release time of job J_i, respectively.

The energy consumption can be divided into two parts: (a) E_{stat} due to the static power consumption of the machines; and (b) E_{dync} due to the dynamic execution of the jobs, which is given by

$$E_{dync} = \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{i,j} \cdot P_{i,j} \cdot U_{i,j} \ , \quad (6)$$

where $\delta_{i,j} = 1$ if job J_i is executed on machine M_j and 0 otherwise. The total energy consumption is therefore $E_{tot} = E_{stat} + E_{dync}$. In this paper, we assume that all machines are turned on at all times, so the static energy is independent of the scheduling of the jobs.

For the thermal imbalance, we use the maximum temperature T_{max}^{out} and average temperature T_{ave}^{out} at the server outlets as metrics. Apparently, larger values for T_{max}^{out} and T_{ave}^{out} indicate worse thermal imbalance. These two metrics are specified as

$$T_{max}^{out} = \max_{t_1 \le t \le t_2} \max_{1 \le k \le l} T_k^{out}(t) \ , \quad (7)$$

$$T_{ave}^{out} = \frac{1}{(t_2 - t_1) \cdot l} \int_{t_1}^{t_2} \sum_{k=1}^{l} T_k^{out}(t) dt \ , \quad (8)$$

where $[t_1, t_2]$ denotes the interval of interest, in which all jobs arrive and complete their executions.

Due to the heterogeneity of the machines, different job scheduling and machine placement strategies may result in very different job response time, dynamic energy and outlet temperatures. Moreover, these objectives can be conflicting with each other. In next section, we will propose heuristics to address each one of them as well as to deal with their tradeoffs.

D. Motivation for Machine Placement

Many server systems exhibit a non-uniform heat distribution between the positions and the outlets. Consider a simple server system with two positions and two outlets. The first position dissipates 50% of its generated heat to each outlet, whereas the second position dissipates 80% of the heat to outlet 1 and 20% to outlet 2. Given two heterogeneous machines, it is obviously more desirable to place the machine with a larger heat dissipation to the first position, in order to balance the temperatures at the two outlets and to reduce the peak temperature. The situation becomes more challenging in practical server systems with a larger number of positions and outlets, as well as a more complex spatial correlation between them. This motivates the study of machine placement in heterogeneous systems.

IV. Static Machine Placement Heuristic

In this section, we present a heuristic algorithm for static machine placement. As mentioned in Section III-D, the placement of machines can have an impact on the thermal balance at the outlets of a heterogeneous server system. While such an impact comes from both static and dynamic power consumptions of the machines, the dynamic part is not a characteristic of the machines and can be influenced by the job scheduling decisions. Hence, we will only use static power consumption to perform machine placement.[1]

To obtain the optimal placement of machines based on their static power is NP-hard, since it can be shown to contain the 3-partition problem [9] as a special case. Therefore, we will focus on heuristic solutions, and present a Greedy Machine Placement (GMP) heuristic to reduce the maximum outlet temperature. Algorithm 1 presents its pseudocode.

First, GMP sorts the machines in descending order of static power consumption, since machines that consume more power will have larger contributions to the temperatures at all outlets, so they will be placed first to avoid high peak temperature values. Let T_k^{out} denote the existing temperature at outlet L_k, and let $T_{max}^{out}(x)$ denote the maximum outlet temperature if the next machine $M_j \in \mathcal{M}$ is placed in position C_x, i.e.,

$$T_{max}^{out}(x) = \max_{k=1 \cdots l} (T_k^{out} + g(d_{x,k} \cdot U_j^{stat})) \ . \quad (9)$$

Then, machine M_j will be placed in one of the remaining positions $C_{x'} \in \mathcal{C}$ that minimizes the maximum outlet temperature, i.e., $x' = \text{argmin}_x \, T_{max}^{out}(x)$. After that, the filled position $C_{x'}$ will be removed from the available set $\mathcal{C}$, and the temperatures at all outlets will be updated. The algorithm terminates when all machines in $\mathcal{M}$ are placed in the server.

For the complexity of the GMP heuristic, the sorting and initialization takes $O(m \log m + l)$ time. In the iteration, placing each machine incurs $O(ml)$ time as each remaining

[1]There tends to be a postive correlation between the static power consumption of the machines and their dynamic power. That is, a machine with a higher static power also consumes a higher dynamic power when executing a given job. This justifies the use of static power alone for machine placement.

Algorithm 1 Greedy Machine Placement

Input: Set $\mathcal{M} = \{M_1, M_2, \cdots, M_m\}$ of m machines, set $\mathcal{C} = \{C_1, C_2, \cdots, C_m\}$ of m server positions, and heat distribution matrix D.
Output: A mapping π from server positions to machines.

```
Sort the machines in descending order of static power consumptions, i.e., U_1^stat ≥ U_2^stat ≥ ··· ≥ U_m^stat
Initialize T_k^out = 0 for all 1 ≤ k ≤ l
for j = 1 to m do
    x' = 0 and T_max^out(x') = ∞
    for each C_x ∈ C do
        T_max^out(x) = max_{k=1..l} (T_k^out + g(d_{x,k} · U_j^stat))
        if T_max^out(x) < T_max^out(x') then
            T_max^out(x') = T_max^out(x) and x' = x
        end if
    end for
    Place machine M_j to position C_{x'}, i.e., π(x') = j
    Update T_k^out = T_k^out + g(d_{x',k} · U_j^stat) for all 1 ≤ k ≤ l, and update C = C \ C_{x'}
end for
```

position is tested to determine the maximum outlet temperature. Therefore, the overall complexity is $O(m^2 l)$.

V. Online Job Scheduling Heuristics

With a fixed machine placement, we need to perform job scheduling in an online manner. This section presents heuristics for online job scheduling to optimize various objectives.

A. Greedy Online Scheduling Framework

All of our online scheduling heuristics fall into a Greedy Online Scheduling (GOS) framework, which is described in Algorithm 2.

Algorithm 2 Greedy Online Scheduling

Input: Arrival of a new job J_i, and its processing time $P_{i,j}$ and dynamic power consumption $U_{i,j}$ if it is executed on any machine $M_j \in \mathcal{M}$.
Output: Assignment of job J_i to a machine in $\mathcal{M}$.

```
j' = 0 and H_{i,j'} = ∞
for j = 1 to m do
    if H_{i,j} < H_{i,j'} then
        H_{i,j'} = H_{i,j} and j' = j
    end if
end for
Assign job J_i to machine M_{j'}
```

Under the GOS framework, any newly arrived job will be assigned greedily to a machine. The variable $H_{i,j}$ shown in the algorithm represents the cost of assigning the new job J_i to machine M_j. Depending on the target objective, $H_{i,j}$ can be a function of job response time, energy consumption, outlet temperature, or even a composite or combined function of these objectives. The job will then be assigned to a machine $M_{j'}$ with the minimum cost, i.e., $j' = \text{argmin}_j\, H_{i,j}$. The rest of this section will describe heuristics that minimize different cost functions depending on the optimization objectives.

B. Mono-Objective Scheduling

Mono-objective scheduling considers a single optimization objective when deciding where to assign each job. In this subsection, we present three mono-objective scheduling heuristics that minimize job response time, dynamic energy consumption and maximum outlet temperature, respectively. The following describes the three heuristics and their cost functions.

- *Fastest*: Assign job J_i to a machine that renders the minimum job response time. The cost function is

$$H^R_{i,j} = \max(r_i, t^{avail}_j) + P_{i,j} \ , \tag{10}$$

  where r_i is the release time of job J_i and t^{avail}_j denotes the latest time when machine M_j becomes available.
- *Greenest*: Assign job J_i to a machine that incurs the minimum dynamic energy consumption. The cost function is

$$H^E_{i,j} = P_{i,j} \cdot U_{i,j} \ . \tag{11}$$

- *Coolest*: Assign job J_i to a machine that minimizes the maximum outlet temperature. The cost function is

$$H^T_{i,j} = \max_{k=1\cdots l} (T^{out}_k + g(d_{j,k} \cdot U_{i,j})) \ , \tag{12}$$

  where T^{out}_k denotes the existing temperature at outlet L_k before assigning the job, and $d_{j,k}$ denotes the fraction of heat contributed from machine M_j to outlet L_k.

C. Multi-Objective Scheduling with a Fuzzy-Based Priority Approach

To optimize two or more objectives at the same time, we propose a novel *fuzzy-based priority* approach to perform online job scheduling.

1) Dual-Objective Scheduling: We first consider optimizing two objectives, for which we use the following composite cost function

$$H^{X,Y}_{i,j} = \langle \overline{H}^X_{i,j}(f), H^Y_{i,j} \rangle \ . \tag{13}$$

In this case, the objectives X and Y are considered one after another by first selecting all machines that offer the best performance in terms of X, and then selecting among this subset any machine that offers the best performance in terms of Y. To avoid depriving the second objective altogether, a *fuzzy factor* f is used to relax the selection criterion for the first objective up to an acceptable margin. The purpose of introducing this factor is to explore any potential improvement for Y while maintaining the performance for X within the target range. Figure 1 illustrates the basic principle of this approach using a simple example. As we can see, the machine that compromises the first objective X up to the specified fuzzy factor is selected to improve the second objective Y. The simple priority approach, on the other hand, would have scheduled for the best X with much worse Y. The value of the fuzzy factor as well as the priority should depend on the relative importance of the two objectives to optimize, which can be set by the user or the system administrator.

To implement fuzzy-based priority in the GOS framework as shown in Algorithm 2, the cost function for the first objective X is normalized between 0 and 1 in order to take the fuzzy factor into account, i.e.,

$$\overline{H}^X_{i,j} = \frac{H^X_{i,j} - H^X_{i,min}}{H^X_{i,max} - H^X_{i,min}} \ , \tag{14}$$

where $H^X_{i,min}$ and $H^X_{i,max}$ denote the minimum and maximum costs in terms of objective X among all machines to assign

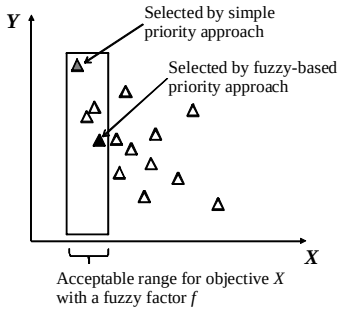


Fig. 1. The fuzzy-based priority approach in dual-objective scheduling.

job J_i. The following rule compares the relative costs of any two machines.

Fuzzy-Based Priority Rule: The costs incurred by scheduling job J_i on any two machines M_{j_1} and M_{j_2} satisfy $H^{X,Y}_{i,j_1} < H^{X,Y}_{i,j_2}$ if one of the following conditions holds:

- $\overline{H}^X_{i,j_1} \leq f < \overline{H}^X_{i,j_2}$, or
- $\overline{H}^X_{i,j_1} \leq f$ and $\overline{H}^X_{i,j_2} \leq f$ and $H^Y_{i,j_1} < H^Y_{i,j_2}$, or
- $\overline{H}^X_{i,j_1} < \overline{H}^X_{i,j_2} \leq f$ and $H^Y_{i,j_1} = H^Y_{i,j_2}$, or
- $f < \overline{H}^X_{i,j_1} < \overline{H}^X_{i,j_2}$, or
- $f < \overline{H}^X_{i,j_1} = \overline{H}^X_{i,j_2}$ and $H^Y_{i,j_1} < H^Y_{i,j_2}$.

This rule can be applied to optimize any two objectives with well-defined cost functions, such as the ones given in Equations (10)-(12) for job response time, energy consumption and maximum outlet temperature, respectively.

2) Multi-Objective Scheduling: With more than two objectives, we can take a similar approach of optimizing one objective after another, but with combined cost functions that consist of two or more objectives. For instance, the *weighted sum* method can be used to combine job response time and energy consumption to form a single objective, i.e.,

$$H^{RE}_{i,j} = \alpha \overline{H}^R_{i,j} + (1-\alpha)\overline{H}^E_{i,j} \ , \tag{15}$$

where $\alpha \in [0,1]$ denotes the relative weight assigned to job response time. Note that the cost functions for both objectives are normalized between 0 and 1 to form meaningful combination. Then, to optimize the maximum outlet temperature with the combined cost (of job response time and energy consumption), the following composite cost function can be constructed

$$H^{T,RE}_{i,j} = \langle \overline{H}^T_{i,j}(f), H^{RE}_{i,j} \rangle \ , \tag{16}$$

or conversely

$$H^{RE,T}_{i,j} = \langle \overline{H}^{RE}_{i,j}(f), H^T_{i,j} \rangle \ . \tag{17}$$

In the case where the first objective is a combination of two or more objectives as in Equation (17), the combined cost needs again to be normalized to take into account the fuzzy factor, i.e.,

$$\overline{H}^{RE}_{i,j} = \frac{H^{RE}_{i,j} - H^{RE}_{i,min}}{H^{RE}_{i,max} - H^{RE}_{i,min}} \ . \tag{18}$$

The fuzzy-based priority rule described previously can then be applied in the same way as before. The exact priorities/weights among different objectives and the fuzzy factor again depend on their relative importance.

VI. Performance Evaluations

In this section, we will evaluate the proposed machine placement and job scheduling heuristics. The evaluations are performed using the DCworms simulator [14] developed for modeling and simulating performance, power and thermal behaviors of server systems in datacenters.

A. Server Platform

Our modeled platform is based on Christmann's *Resource Efficient Cluster Server (RECS)* [4], which is a 1U multi-server system consisting of 18 heterogeneous computing nodes with integrated cooling support. The built-in power and temperature sensors allow the hardware and the application profiles to be monitored and modeled with fine granularity and high accuracy. This system represents an emerging class of high-density servers, which allows a significant number of them to be integrated in just a few rack units. While the RECS platform is chosen to conduct our experiments, our proposed models and heuristics can be generally applied to other heterogeneous servers at both cluster and datacenter environments.

Our heterogeneous RECS server consists of 8 nodes of Intel i7-2715QE, 4 nodes of Intel Atom D510 and 6 nodes of AMD G-T40N. Table I describes the detailed hardware configuration. The 18 positions of the server are laid out in two rows, as depicted in Figure 2, where two positions along the same column share a pair of inlet and outlet, with airflow drawn by fans directly attached to them.

TABLE I
Hardware configuration of the RECS server.

	Intel Core i7-2715QE	Intel Atom D510	AMD G-T40N
Frequency	2.1GHz	1.66GHz	1GHz
Static power	11.5W	9W	6.4W
#Cores (#Threads)	4(8)	2(4)	2(2)
RAM	16GB	2GB	4GB
Cache	6MB	1MB	1MB
Node count	8	4	6

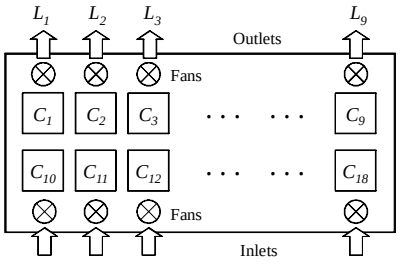


Fig. 2. The layout of the RECS server system.

For simulation purpose, the temperature at all inlets is fixed at $T^{in} = 25^oC$. The thermodynamic constants are set to be $\rho = 1.168kg/m^3$ and $C = 1004Joule/(^oC \cdot kg)$. With all fans turned on, the air throughput at each inlet/outlet is around $Q = 0.0055m^3/s$, according to the measurements

performed in [24]. Simple profiling of the hardware also gives the following heat-distribution matrix:

$$d_{x,k} = \begin{cases} 1, & \text{if } x = k \\ 0.84, & \text{if } x = k + 9 \\ 0, & \text{otherwise} \end{cases},$$

which suggests that each inlet node contributes only 84% of the generated heat to the corresponding outlet due to the relatively long heat dissipation path. The remaining heat stays in or is dissipated through other directions of the server enclosure, which is hard to measure.

B. HPC Benchmarks and Workload

For the simulation, we adopt the set of HPC benchmarks used in [14], which consist of the following applications: fft, c-ray, abinit, linpack, and tar. In particular, an application-specific approach was employed to build the performance and power profiles of these applications with different input parameters and number of threads. Table II shows the average execution time and dynamic power consumption of each application on the three types of computing nodes. The exclusive execution mode (i.e., running one job per node) was used to ensure accurate measurements of the application profiles. Lack of values in the table means that the particular application could not be executed on the corresponding node. Therefore, the node is ignored by the online scheduling heuristics for assigning that application.

TABLE II
AVERAGE EXECUTION TIME AND DYNAMIC POWER CONSUMPTION OF THE BENCHMARKS.

	Intel Core i7-2715QE	Intel Atom D510	AMD G-T40N
fft	1375s, 11.5W	6040s, 0.75W	7710s, 2.58W
c-ray	1445s, 11.79W	8445s, 0.84W	9650s, 2.16W
abinit	4388s, 22.52W	-	-
linpack	1360s, 18.30W	20130s, 2.20W	-
tar	6400s, 9.83W	23385s, 1.79W	22900s, 3.08W

The simulation workload consists of 1000 jobs, and each job is randomly selected from one of these benchmarks. The jobs arrive at the system according to the Poisson process. The load intensity ρ is proportional to the average arrival rate λ (in number of jobs per hour), and it is given by $\rho = \lambda/10$.

C. Simulation Results

We apply the greedy heuristic presented in Section IV to generate a machine placement for the RECS server system, which will be used throughout this section for evaluating different scheduling heuristics. In Section VI-C4, we will come back to machine placement and compare our heuristic with two alternative placements for evaluating the impact on outlet temperatures. All results are obtained by carrying out the experiments 10 times and taking the average.

1) Performance of Mono-Objective Heuristics: We first evaluate the three mono-objective scheduling heuristics presented in Section V-B. Their performance will be used as references for exploring the tradeoffs of two or more objectives.

Two versions of the three heuristics are implemented. One considers only the available or idle machines when assigning each job, and in case all machines are busy the assignment will be postponed until some machine becomes available. The other version considers all machines and therefore may possibly reserve a future time slot of a machine for the job in advance. We call the two versions "available" and "all", respectively. They are compared with two other scheduling policies, namely *Random* and *RoundRobin*. The former assigns each job to an available machine randomly, and the latter selects the machines in turn for assigning the jobs. Both policies are commonly used for balancing the machine loads.

Figures 3 and 4 present the simulation results of the two versions. The results show that the three heuristics (*Fastest*, *Greenest* and *Coolest*) achieve the best performance with respect to their target objective functions. For the "available" heuristics as shown in Figure 3, the performance gains under light loads are up to 40-60% for average job response time, around 15% for dynamic energy consumption, and 1.4-1.6^{o}C for maximum outlet temperature. The advantages diminish slowly as the load intensity increases, as the utilization of the computing nodes becomes higher, so jobs tend to be postponed or assigned to nodes with less energy or thermal efficiency. For the average outlet temperature, all heuristics have similar performance except *RoundRobin*, which performs better at medium to high loads at the expense of job response time and energy consumption.

For the performance of the "all" heuristics, by comparing Figures 3(a) and 4(a), we can see that the average job response time becomes worse by at least an order of magnitude. (Note the difference in scale.) Since all machines are considered in this case, a subset of them is almost always selected for their energy and thermal efficiency, resulting in highly unbalanced machine loads and hence deterioration of job response times. For the same reason, the advantages of *Greenest* and *Coolest* in terms of dynamic energy consumption and maximum outlet temperature are maintained even at high load intensities. In particular for *Greenest*, all jobs are concentrated on the most energy-efficient nodes (e.g., Intel Atom) while the other machines with low power efficiency but high performance (e.g., Intel Core i7) are left idle. Naturally, it leads to the lowest average outlet temperature for this heuristic.

For the *Fastest* heuristic, Figure 5 shows that the "available" version, which implicitly balances the utilization of the computing resources, outperforms the "all" version in terms of the jobs' average response time at high loads. The "all" version, on the other hand, performs better in terms of the maximum response time of all jobs, since it makes the best local decision for each individual job regardless of the machine availability.

In the rest of this section, we will focus on the "available" heuristics to study the tradeoffs of various objectives and the impact of different machine placements.

2) Impact of Fuzzy Factor for Dual-Objective Scheduling: We now evaluate the effectiveness of the fuzzy-based priority approach for exploring the tradeoffs between two objectives. For this purpose, we consider three cost functions defined in Equations (10)-(12), for minimizing average job response time, dynamic energy consumption and maximum outlet tempera-

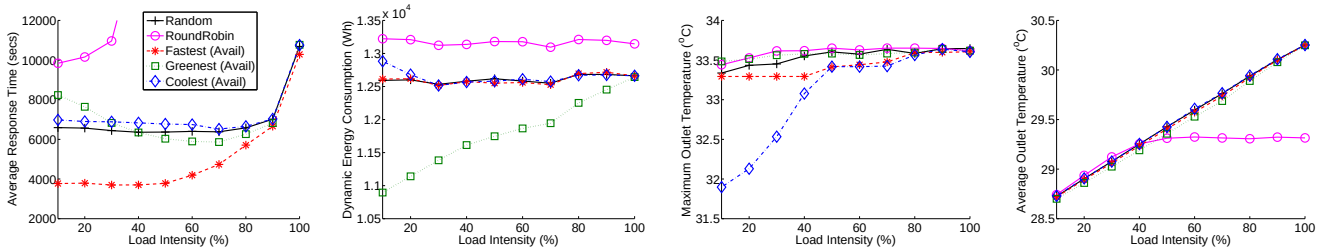


Fig. 3. Performance of the "available" version of the three mono-objective scheduling heuristics with *Random* and *RoundRobin*.

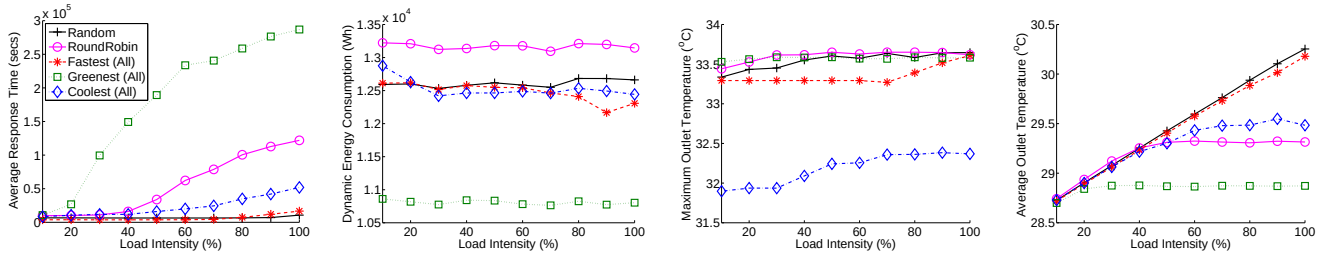


Fig. 4. Performance of the "all" version of the three mono-objective scheduling heuristics with *Random* and *RoundRobin*.

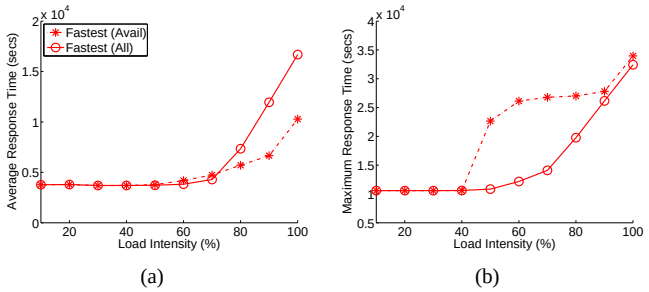


Fig. 5. Comparison of the two versions of the *Fastest* heuristic in terms of average and maximum job response time.

ture, respectively.

Figure 6 shows the impact of varying the fuzzy factor f from -1 to 1 when minimizing any two out of the three objectives at 20% load intensity. In particular, $f = -1$ means that the scheduling decision is solely based on the first objective. In this case, ties on the first cost function are broken randomly, so the second objective is completely ignored. Figures 6(a), 6(e) and 6(i) along the diagonal show the optimization of two identical objectives, which make them equivalent to the mono-objective case. The other figures plot the changes of the two objectives as a function of f, with the first objective shown on the left Y axis and the second one on the right Y axis.

First, Figures 6(b) and 6(d) indicate no change to the average response time and dynamic energy consumption unless the fuzzy factor f is set above 0.7 for $H_{i,j}^{R,E}$ and 0.5 for $H_{i,j}^{E,R}$. The results suggest that it is difficult to have good performance for one objective without significant performance degradation for the other. Depending on the relative importance of the two objectives, the fuzzy factor in this case can be set in the range of $[0.6, 0.9]$ to obtain a desirable tradeoff.

Figures 6(c) and 6(f) show that the maximum outlet temperature can be minimized simultaneously with average response time or dynamic energy consumption as soon as the second objective (temperature) is taken into account, i.e., $f \geq 0$, and before the first objective (response time or energy) is compromised, i.e., $f \leq 0.6$. Similar results can be observed in Figures 6(g) and 6(h), which optimize the temperature before response time or energy consumption. In this case, the second objective stabilizes after f reaches 0.3, and interestingly, the first objective (temperature) is also reduced slightly (by 0.1-0.2^{o}C) due to the consideration of the second objective. The improvement is probably due to the perturbation introduced in the scheduling decision that helped escape the local optimum, which was experienced by considering temperature alone. In general for optimizing response time or energy with maximum outlet temperature, the fuzzy factor can be set in the range of $[0.3, 0.6]$ for the optimal performance.

Figure 7 shows the results when the load intensity is at 40%. As can be seen, by setting appropriate values for the fuzzy factor, desirable tradeoffs between average response time and dynamic energy can again be attained. Temperature can also be optimized together with the other two objectives, but stablizes at a higher value due to the increase of load intensity. The results demonstrate the effectiveness of this fuzzy-based approach for exploring and optimizing dual-objective tradeoffs.

3) Results of Multi-objective Scheduling: We use average response time, dynamic energy consumption and maximum outlet temperature as the three objectives to evaluate the performance of multi-objective scheduling.

Previous results on dual-objective scheduling have shown the difficulty of minimizing response time and energy simultaneously. Hence, we combine the two objectives with a weighted cost function $H_{i,j}^{RE}$ as defined in Equation (15), and optimize it together with temperature using the composite

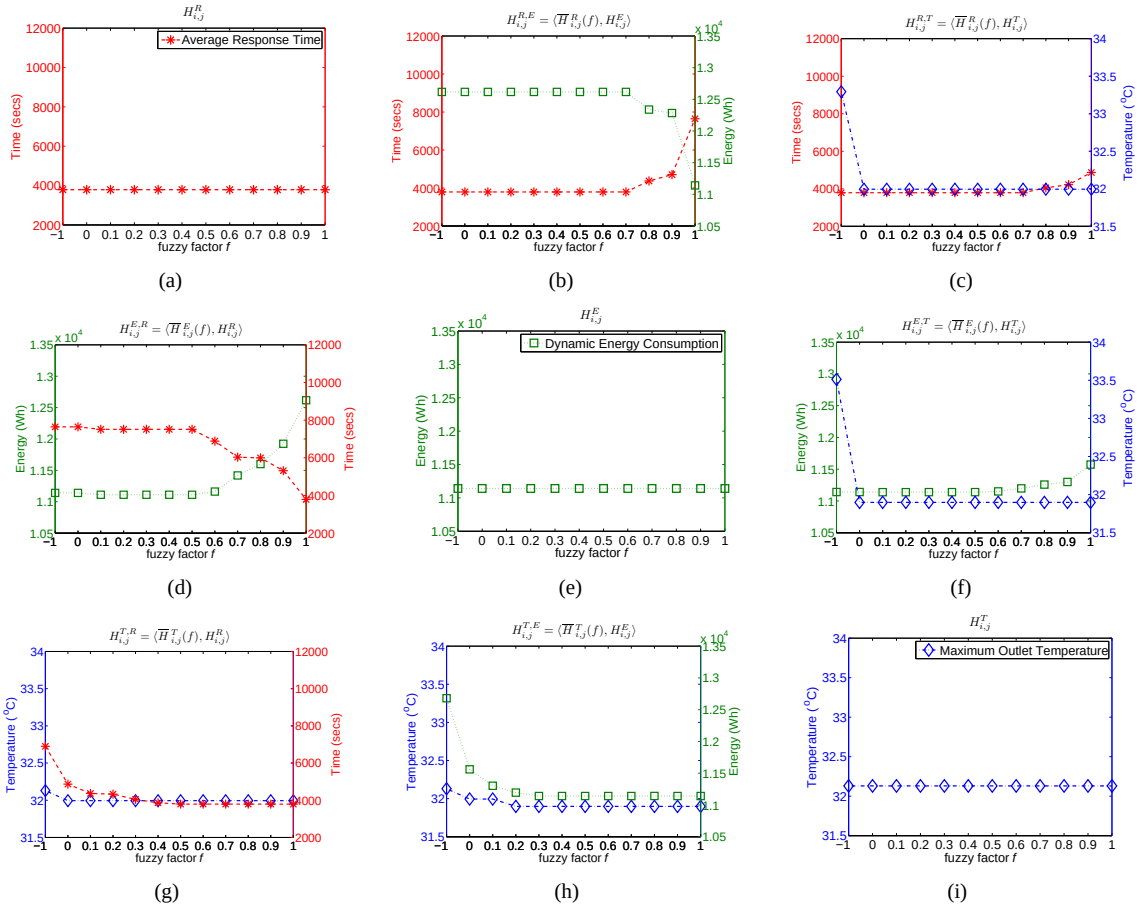


Fig. 6. The use of fuzzy-based priority approach for dual-objective scheduling at 20% load intensity. The subfigures on the diagonal show the reference values for average response time, dynamic energy consumption and maximum outlet temperature, respectively. The legends therein apply to all the subfigures.

functions $H_{i,j}^{T,RE}$ and $H_{i,j}^{RE,T}$ as defined in Equations (16) and (17). Since similar performance was observed for the two cases, we only present the results for $H_{i,j}^{T,RE}$.

Figure 8 shows the results with different values of α and f at 20% load intensity. First, as α increases from 0 to 1 in the weighted sum, we can clearly see the performance transitions for both response time and energy consumption as long as they are considered, i.e., $f \neq -1$. When the value of α is large, indicating that response time is favored more than energy, the response time is reduced as f increases. The same can be observed for energy consumption when the value of α is small. This demonstrates the effectiveness of using f in exploring the potential performance improvements for combined objectives. Figure 8(c) shows that the maximum outlet temperature can again be slightly improved by considering the response time and energy as the second objective, which also correspond to the results shown in Figures 6(g) and 6(h).

Figure 9 shows the performance of the algorithm with fixed $f = 0.4$ and $\alpha = 0.5$. We call the resulting algorithm *Combined* and compare it with the three reference heuristics. The results show a good tradeoff between average response time and dynamic energy consumption under all system loads, using *Fastest* and *Greenest* as the references. Moreover, this is achieved together with good maximum outlet temperature, which is close to the one obtained by the *Coolest* heuristic.

4) Impact of Different Machine Placements: Lastly, we study the impact of machine placement on the performance of the online scheduling heuristics. To this end, we generate three different placements of the machines, one by our GMP heuristic and two by its variants which we call MP2 and MP3, respectively. The two variants work in a similar fashion as GMP. However, MP2 sorts the machines in ascending order of static power instead of descending order, while MP3 assigns each machine to a remaining position that maximizes the maximum outlet temperature instead of minimizing it. Apparently, MP2 and MP3 are counter-intuitive, so they may not generate desirable machine configrations. They are included to simply demonstrate the impact of different machine placements on the performance, especially the outlet temperature.

Figure 10 shows the performance of the three machine placements for the same scheduling heuristic that optimizes the combined cost function $H_{i,j}^{RE,T}$ with $f = 0.1$ and $\alpha = 0.4$. The results clearly show that job response time and energy consumption are not affected by different machine placements. However, our GMP heuristic reduces the maximum outlet temperature by 1^oC compared to MP2, and depending on the load of the system the improvement is up to 3^oC compared to MP3. Moreover, GMP also improves the average outlet temperature by around 0.1-0.2^{o}C, especially at high system loads. Similar performance was also observed for other scheduling heuristics.

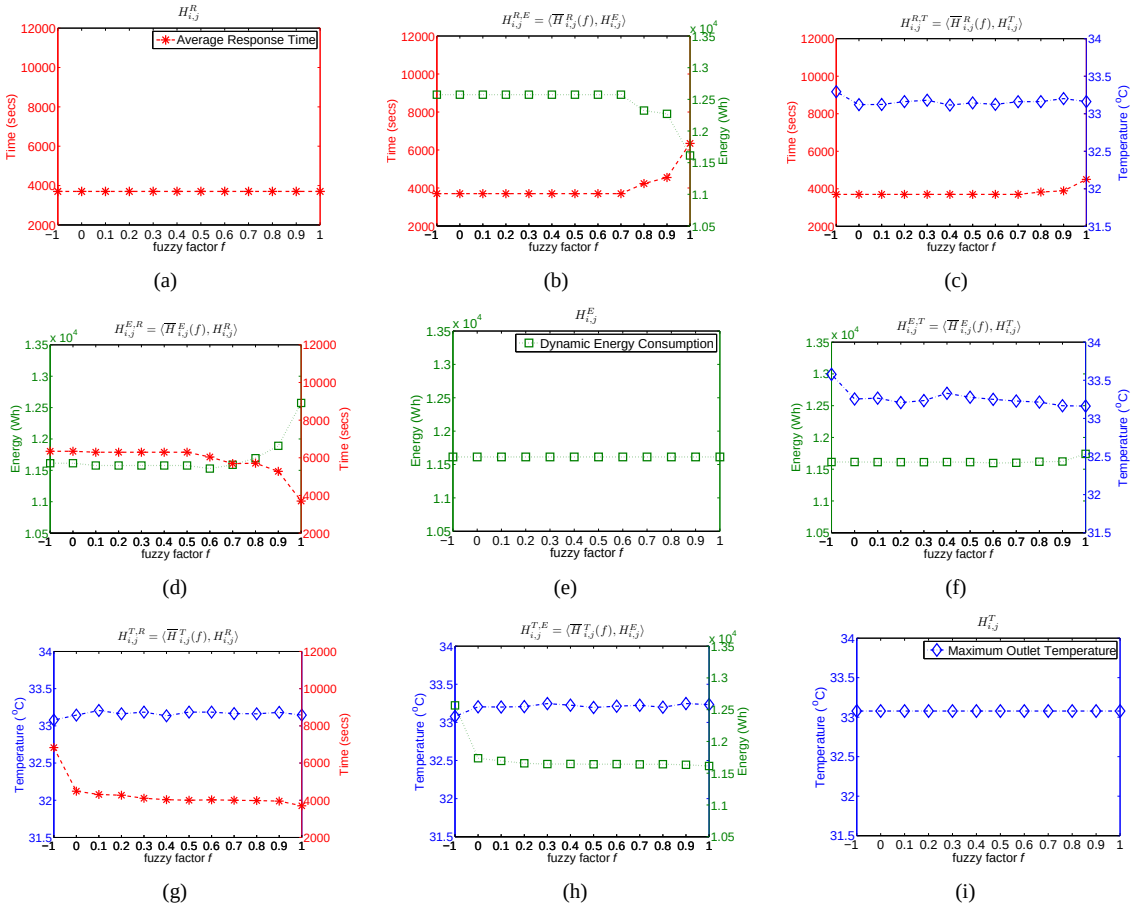


Fig. 7. The use of fuzzy-based priority approach for dual-objective scheduling at 40% load intensity. The subfigures on the diagonal show the reference values for average response time, dynamic energy consumption and maximum outlet temperature, respectively. The legends therein apply to all the subfigures.

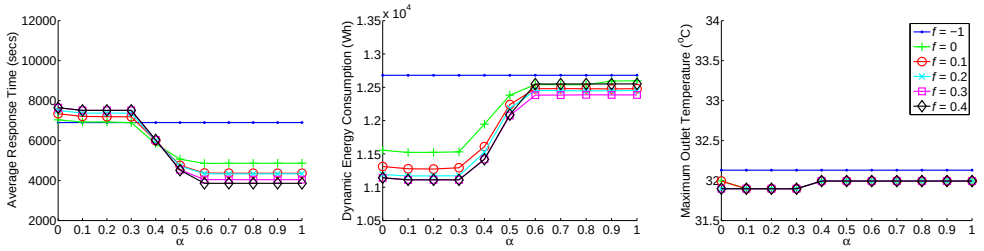


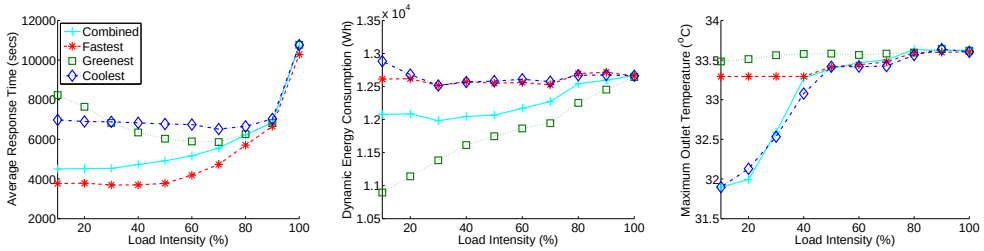
Fig. 8. Multi-objective scheduling for $H_{i,j}^{T,RE} = \langle \overline{H}_{i,j}^{T}(f), \alpha \overline{H}_{i,j}^{R} + (1-\alpha)\overline{H}_{i,j}^{E} \rangle$ with different f and α at 20% load intensity.

The results confirm that machine placement indeed affects the thermal balance at the server outlets, which directly impacts the efficiency and cost of the cooling system.

VII. Conclusion and Future Work

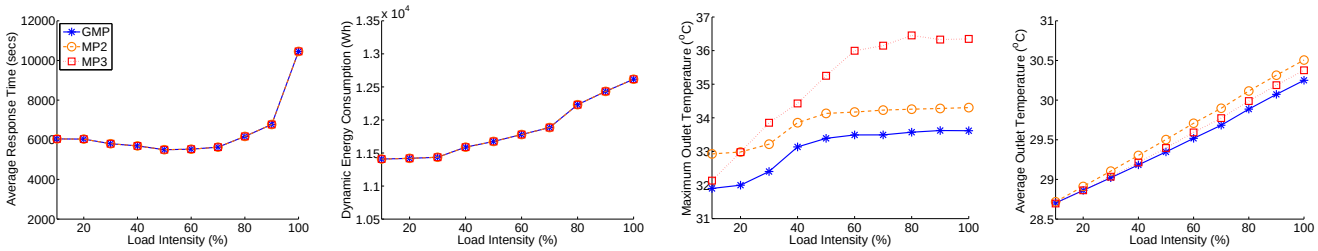
In this paper, we have considered online job scheduling and static machine placement for heterogeneous server systems. We applied a novel fuzzy-based priority approach to a greedy online scheduling framework for simultaneously optimizing multiple objectives, including average job response time, dynamic energy consumption and maximum outlet temperature. Simulations based on a high-density server system have shown that optimizing only one objective can have a negative impact on the others. The results also demonstrated the effectiveness of our approach for exploring and optimizing the tradeoffs between two or more objectives. In particular, response time and energy were shown to be orthogonal metrics, which are difficult to minimize simultaneously. However, either of them or a weighted combination can be optimized together with outlet temperature by setting appropriate fuzzy factors. Finally, different machine placements were shown to have a strong impact on the thermal balance at the server outlets, which has implications for the cost of the cooling system.

For future work, we will apply power management techniques, such as DVS or turning off idle machines, to reduce the static power consumption for better energy and thermal efficiency. For machine placement, Computational Fluid Dy-



(a) (b) (c)

Fig. 9. Multi-objective scheduling for $H_{i,j}^{T,RE} = \langle \overline{H}_{i,j}^{T}(f), \alpha \overline{H}_{i,j}^{R} + (1-\alpha)\overline{H}_{i,j}^{E} \rangle$ with $f = 0.4$ and $\alpha = 0.5$ at different load intensities.



(a) (b) (c) (d)

Fig. 10. Scheduling for $H_{i,j}^{RE,T} = \langle \overline{H}_{i,j}^{RE}(f), \alpha \overline{H}_{i,j}^{R} + (1-\alpha)\overline{H}_{i,j}^{E} \rangle$ with $f = 0.1$ and $\alpha = 0.4$ under three different machine placements.

namics (CFD) simulations can be carried out to validate the results. Finally, the approach presented in this paper can be extended to scheduling multiple servers in a datacenter by directly considering the cooling cost.

ACKNOWLEDGMENT

This research was funded by the European Commission under contract 288701 through the project CoolEmAll.

REFERENCES

[1] I. Assayad, A. Girault, and H. Kalla. A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints. In *DSN*, page 347, 2004.

[2] J. Barbosa, and B. Moreira. Dynamic job scheduling on heterogeneous clusters. In *ISPDC*, pp. 3-10, 2009.

[3] L.A. Barroso, and U. Holzle. The case for energy-proportional computing. *Computer*, 40(12):33-37, 2007.

[4] Christmann. Description for Resource Efficient Computing System (RECS). *http://shared.christmann.info/download/project-recs.pdf*, 2009.

[5] G. Da Costa. Heterogeneity: The key to achieve power-proportional computing. In *CCGrid*, pp. 656-662, 2013.

[6] Z. Du, H. Sun, Y. He, Y. He, D.A. Bader, and H. Zhang. Energy-efficient scheduling for best-effort interactive services to achieve high response quality. In *IPDPS*, pp. 637-648, 2013.

[7] J.J. Durillo, V. Nae, and R. Prodan. Multi-objective workflow scheduling: An analysis of the energy efficiency and makespan tradeoff. In *CCGrid*, pp. 203-210, 2013.

[8] H.M. Fard, R. Prodan, J. Barrionuevo, and T. Fahringer. A multi-objective approach for workflow scheduling in heterogeneous environments. In *CCGrid*, pp. 300-309, 2012.

[9] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. *First Edition, W. H. Freeman*, 1979.

[10] R. Garg and A.K. Singh. Reference Point Based Multi-Objective Optimization to Workflow Grid Scheduling. *Int. J. Appl. Evol. Comput*, 3(1):80-99, 2012.

[11] S.K. Garg, C. Yeo, A. Anandasivam, and R. Buyya. Environment-conscious scheduling of HPC applications on distributed Cloud-oriented data centers. *J. Parallel Distrib. Comput.*, 71(6):732-749, 2011.

[12] L. Grandinetti, O. Pisacane, and M. Sheikhalishahi. An approximate ϵ-constraint method for a multi-objective job scheduling in the cloud. *Future Generation Comp. Syst.*, 29(8):1901-1908, 2013.

[13] J. Koomey. Worldwide electricity used in data centers. *Environ. Res. Lett.*, Vol. 3, 034008, 2008.

[14] K. Kurowski, A. Oleksiak, W. Piatek, T. Piontek, A. Przybyszewski, and J. Weglarz. DCworms – A tool for simulation of energy efficiency in distributed computing infrastructures. *Simulation Modelling Practice and Theory*, 39:135-151, 2013.

[15] Y. Lee and A.Y. Zomaya. Energy conscious scheduling for distributed computing systems under different operating conditions. *IEEE Trans. Parallel Distrib. Syst.*, 22(8):1374-1381, 2011.

[16] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making scheduling "cool": temperature-aware workload placement in data centers. In *USENIX Conference*, pp 5-5, 2005.

[17] N.B. Rizvandi, J. Taheri, A.Y. Zomaya and Y. Lee. Linear combinations of DVFS-enabled processor frequencies to modify the energy-aware scheduling algorithms. In *CCGrid*, pp. 388-397, 2010.

[18] R. Sawyer. Calculating Total Power Requirements for Data Centers. *White Paper*, 2004.

[19] H.F. Sheikh and I. Ahmad. Simultaneous optimization of performance, energy and temperature for DAG scheduling in multi-core processors. In *IGCC*, pp. 1-6, 2012.

[20] H. Sun, Y. Cao, W.-J. Hsu. Non-clairvoyant speed scaling for batched parallel jobs on multiprocessors. In *ACM Conference on Computing Frontiers*, pp. 99-108, 2009.

[21] Q. Tang, S. Gupta, and 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):1458–1472, 2008.

[22] H. Topcuoglu, H, S. Hariri, and M.-Y. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.*, 13(3):260-274, 2002.

[23] S.A. Torabi, N. Sahebjamnia, S.A. Mansouri, M.A. Bajestani. A particle swarm optimization for a fuzzy multi-objective unrelated parallel machines scheduling problem. *Applied Soft Computing*, 13(12):4750-4762, 2013.

[24] M. Vor dem Berge, G. Da Costa, M. Jarus, A. Oleksiak, W. Piatek, and E. Volk. Modeling data center building blocks for energy-efficiency and thermal simulations. In *E2DC*, 2013.

[25] J. Xu and J. A. B. Fortes. Multi-objective virtual machine placement in virtualized data center environments. In *GREENCOM-CPSCOM*, pp. 179-188, 2010.

[26] F. Zhang, J. Cao, K. Li, S.U. Khan, and K. Hwang. Multi-objective scheduling of many tasks in cloud platforms. *Future Generation Comp. Syst.*, 2013.