Deadline-aware Energy Management in Data Centers

Cengis Hasan
School of Informatics
The University of Edinburgh, UK
email: chasan@inf.ed.ac.uk

Zygmunt J. Haas
School of Electrical and Computer Engineering
Cornell University, Ithaca, NY, USA
email: haas@ece.cornell.edu

***Abstract*—We study the dynamic energy optimization problem in data centers. We formulate and solve the following offline problem: given a set of jobs to process, where the jobs are characterized by arrival instances, required processing time, and completion deadlines, and given the energy requirements of switching servers ON or OFF, in which time-slot which server has to be assigned to which job; and in which time-slot which server has to be switched ON or OFF, so that the total energy is optimal for some time horizon. We formulate the offline problem as a binary integer program that can be considered as a new version of generalized assignment problem which includes new constraints stemming from deadline characteristics of jobs and the activation energy of servers. We propose an online algorithm that solves the problem heuristically, and we compare it to random assignment solution.**

***Index Terms*—job scheduling, data centers, deadline-aware energy management, generalized assignment problem, online algorithm.**

I. Introduction

Energy consumption is one of the most important practical and timely problem associated with data centers for cloud computing. The urgency of this problem has been identified by both, the governmental agencies and the industry. While policies that consider the physical design of the data centers have been studied in the technical literature rather thoroughly, the operational characteristics of the data centers (e.g., the required performance of the executed applications, such as the type of services and the applications being supported, their QoS requirements, associated background server maintenance processes, etc.) have rarely been accounted for. In particular, there is a need to investigate the implications of exploiting the operational characteristics of the data centers in the context of energy management policies, as those could lead to potentially significant energy savings and operational cost reduction of a computer cloud.

In this work, we study the effect of the deadline characteristics of the jobs (applications) and their requirements on the design of the data centers' energy management policies. A particular interest of our study is to solve the dynamic energy optimization problem. Assuming a time interval, we seek to minimize the total energy consumed during this time interval. We solve the offline problem, in which we know the arrival times of the jobs, their service demands and deadlines. In particular, the offline problem is defined as follows:

- consider a discrete time horizon: $T = \{1,\ldots,t_{max}\}$ where an element in T is a time-slot,
- a server can be either in the ON or the OFF states,
- in any time-slot, a server can serve at most one job,
- in any time-slot, a job can be served by at most a single server,
- a job has to be served within its deadline constraint, i.e., the job's service demand has to be completed before the job's deadline,
- a server needs n_{ON} consecutive slots (i.e., the setup time) and E_{ON} energy per time-slot in order to transition from the OFF state to the ON state,
- a server consumes E_{slot} energy per time-slot when it serves a job.

Based on these constraints, we calculate the minimal total energy in the following way:

> ***find****: in which time-slot which server has to be assigned to which job; and in which time-slot which server has to be switched ON or OFF.*

We refer to the above problem as *offline problem*, to indicate that all the jobs' characteristics (arrival instance, processing duration, and completion deadline) are known before the problem is solved. The corresponding online problem is one in which some of the jobs arrive after the execution of other jobs have already started and, thus, the characteristics (and even the existence) of all the jobs is not known at the time that the initial assignment is made. Therefore, the offline solution becomes a lower bound for evaluating of an online algorithm. Note that an online algorithm can find a solution for present time only. We propose an online algorithm which aims to minimize the total cost **per time-slot**. The cost is composed of the energy cost per time-slot and a weighted sum of deadline and service demand of the jobs. When a server is not assigned to a job, we keep this server ON during a portion of time. This is a similar approach studied in [1]. We calculate every possible assignment of servers to jobs present in the system based on the above cost definition. Then, we find which assignment of servers to jobs minimizes the total cost per time-slot. We show that this problem corresponds to the assignment problem, which is solvable in polynomial time using the well-known Hungarian algorithm.

The work of Z. J. Haas was supported in part by the NSF Grant number: ECCS-1308208. C. Hasan was a postdoctoral research associate at Cornell University when this work was carried out.

A. Related Work

The methods aiming to reduce data center power consumption could be classified into four approaches [1]: *power-proportionality*, which attempt to guarantee that servers consuming power in proportion to their utilization [2]–[4]; *energy-efficient server design*, which attempt to determine the proper server architecture for a given workload [5]–[7]; *dynamic server provisioning*, which attempts to determine the times when the servers should be kept on or off [8], [9]; *consolidation and virtualization*, which attempts to reduce the power consumption by resource sharing [10]–[12]. We refer the reader to [13]–[17] for further literature related to our work.

The dynamic version of the generalized assignment problem is directly related to the offline problem that we study in this paper. Actually, our problem could be considered as a version of the dynamic generalized assignment problem which includes new additional constraints. In [18], the authors firstly put forward the dynamic generalized assignment problem, and they formulate the continuous-time optimal control model in order to develop an efficient time-decomposition procedure for solving the problem. The discrete time version is studied in [19] by associating a starting time and a finishing time with each task. The authors also use column generation algorithm to compute lower bounds.

II. The Offline Problem

We represent by $N = \{1,\dots,|N|\}$ *the set of servers*, and $J = \{1,\dots,|J|\}$ *the set of jobs*. A job j is characterized by its arrival time-slot t_j and the completion deadline as $t_j + \Delta_j$, where Δ_j denotes *the execution duration* of job j. We denote by $T_j = \{t_j,\dots,t_j+\Delta_j\}$ the maximal lifetime of a job within the system. Consider $t_{max} = \max_{j\in J}(t_j+\Delta_j)$, then the total time horizon is $T = \{1,\dots,t_{max}\}$. We would like to calculate *the optimal assignments that minimize total energy consumption respecting the deadlines of the jobs, starting from time-slot 1 until t_{max}*. The duration of a time-slot is τ. Thus, the deadline of a job is given by $\Delta_j = n_j\tau$, where n_j is an integer. The *initial service demand* of any job j is denoted by w_j in terms of processor cycles. Each server is characterized by its *speed* $s_i =$ [processor cycles/second], $\forall i \in N$. So, the total processing demand that a server can handle during a time-slot is $s_i\tau$. If the service demand of job j is w_j and server i is assigned to job j, then at the end of the current time-slot the service demand of job j is reduced to $w_j - s_i\tau$. We assume that a server needs n_{ON} time-slots in order to switch from the OFF state to the ON state. A server in the ON state consumes E_{slot} energy per time-slot. During transition from the OFF state to the ON state, a server expends E_{ON} energy per time-slot. Let us define the following binary integer variable,

$$x_{ijt} = \begin{cases} 1, & \text{server } i \text{ is assigned to job } j \text{ in time-slot } t \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Note that if $x_{\emptyset jt} = 1$, it means that no server is assigned to job j in time-slot t (it can be assumed that the job is in *hold*). Furthermore, it is also possible for a server to be ON, yet not to be assigned to any job in a time-slot, i.e. $x_{i\emptyset t} = 1$. Based on these definitions, we assume that any server i in time-slot t is in the OFF state if the following holds true:

$$\sum_{j\in J\cup\emptyset} x_{ijt} = 0. \tag{2}$$

We also define a new variable y_{it} with the following meaning:

$$y_{it} = \begin{cases} 1, & \text{server } i \text{ is in transition from OFF to ON in time-slot } t \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

For example, let $n_{\mathrm{ON}} = 3$, and server i is switched ON in time-slot 5. Then, $y_{it} = 1$, only for $t = \{5,6,7\}$. Note also that if $y_{it} = 1$ then, as long as $n_{\mathrm{ON}} > 0$, it is always true that $x_{i\emptyset t} = 1$, otherwise it is not true, i.e. if $y_{it} = 0$ then, $x_{i\emptyset t} = 0$ or 1. This constraint can be given by

$$y_{it} \le x_{i\emptyset t}, \quad \forall i \in N, \forall t \in T. \tag{4}$$

Thus, the minimal total energy can be calculated by

$$\min_{x,y}\left(\sum_{i\in N}\sum_{j\in J\cup\emptyset}\sum_{t\in T_j} E_{\mathrm{slot}} x_{ijt} + \sum_{i\in N}\sum_{t\in T}(E_{\mathrm{ON}} - E_{\mathrm{slot}})y_{it} \,\middle|\, \text{Constraints}\right), \tag{5}$$

where note that if any server i is switched ON in time-slot t, i.e. $y_{it} = 1$ then, $x_{i\emptyset t} = 1$ (due to eq. (4)), thus it corresponds to only E_{ON} energy expenditure in objective part of eq. (5), i.e. $E_{\mathrm{slot}}x_{i\emptyset t} + (E_{\mathrm{ON}} - E_{\mathrm{slot}})y_{it} = E_{\mathrm{ON}}$. The “Constraints” in eq. (5) are elaborated below. As we mentioned above, if a server is in the OFF state in time-slot $t-1$, and it is switched ON in time-slot t, then it will spend n_{ON} consecutive time-slots to transition to the ON state. Using the above definitions, in order to satisfy that all jobs are fully served, we need the following constraints:

$$\sum_{i\in N\cup\emptyset}\sum_{t\in T_j} s_i\tau x_{ijt} \ge w_j, \quad \forall j \in J \tag{6}$$

which means that until the Δ_j/τ time-slot, job j has to be handled. We define $s_\emptyset = 0$, meaning that if a job is not assigned to a particular server in a time-slot, its service demand will not be processed during that time-slot. The constraint allowing at most one job to be assigned to a server in a time-slot can be given by

$$\sum_{j\in J\cup\emptyset} x_{ijt} \le 1, \quad \forall i \in N, \forall t \in T_j. \tag{7}$$

We also need to guarantee that only one server can be assigned to a job or we can choose not to assign any server during a time-slot (i.e. it is not possible to assign more than one server to a job):

$$\sum_{i\in N\cup\emptyset} x_{ijt} = 1, \quad \forall j \in J, \forall t \in T_j. \tag{8}$$

We also set $x_{\emptyset\emptyset t} = 0$, $\forall t \in T$. Moreover, whenever an OFF server is switched ON in time-slot t, then the server cannot be assigned to any job during consecutive n_{ON} time-slots. If a server is OFF in time-slot $t-1$ and switched ON in time-slot t, then the following constraints must hold:

$$\text{if } \sum_{j'\in J\cup\emptyset} x_{ij'(t-1)} = 0 \text{ and } x_{i\emptyset t} = 1, \quad \sum_{t'=t}^{t+n_{\mathrm{ON}}-1} y_{it'} \geq n_{\mathrm{ON}}$$

This if-then constraint can be replaced with the following:

$$\sum_{t'=t}^{t+n_{\mathrm{ON}}-1} y_{it'} \geq n_{\mathrm{ON}} x_{i\emptyset t}\left(1-\sum_{j'\in J\cup\emptyset} x_{ij'(t-1)}\right), \forall i \in N, \forall t \in \tilde{T}. \quad (9)$$

where $\tilde{T} = \{1,\ldots,t_{max} - n_{\mathrm{ON}} + 1\}$. We denote as $z_{it} = x_{i\emptyset t}\left(1-\sum_{j'\in J\cup\emptyset} x_{ij'(t-1)}\right)$ the variable which takes value 1 if server i is switched OFF in time-slot $t-1$ and it is switched ON in time-slot t, otherwise the variable is equal to 0. We are able to transform the multiplication in variable z_{it} as following:

$$\begin{aligned} z_{it} &\leq x_{i\emptyset t}, \quad \forall i \in N, \forall t \in \tilde{T} \\ z_{it} &\leq 1 - \sum_{j'\in J\cup\emptyset} x_{ij'(t-1)}, \quad \forall i \in N, \forall t \in \tilde{T} \\ z_{it} &\geq x_{i\emptyset t} - \sum_{j'\in J\cup\emptyset} x_{ij'(t-1)}, \quad \forall i \in N, \forall t \in \tilde{T}. \end{aligned} \quad (10)$$

Note that the following must always hold:

$$n_{\mathrm{ON}} \sum_{t\in T} z_{it} = \sum_{t\in T} y_{it}, \quad \forall i \in N \quad (11)$$

$$z_{it} \leq y_{it}, \quad \forall i \in N, \forall t \in T. \quad (12)$$

If a server is OFF, it is forbidden to switch it ON last n_{ON} consecutive time-slots of last arriving job:

$$\text{if } \sum_{j'\in J\cup\emptyset} x_{ij'(t_{max}-n_{\mathrm{ON}}-1)} = 0, \quad \sum_{t'=t_{max}-n_{\mathrm{ON}}}^{t_{max}-1} y_{it'} \leq 0$$

which can be transformed to the following form:

$$\sum_{t'=t_{max}-n_{\mathrm{ON}}}^{t_{max}-1} y_{it'} \leq n_{\mathrm{ON}} \sum_{j'\in J\cup\emptyset} x_{ij'(t_{max}-n_{\mathrm{ON}}-1)}, \quad \forall i \in N. \quad (13)$$

On the other hand, no job can be assigned to an OFF server:

$$\text{if } \sum_{j'\in J\cup\emptyset} x_{ij'(t-1)} = 0, \quad \sum_{j'\in J\cup\emptyset} x_{ij't} \leq 0, \forall i \in N, \forall t \in T.$$

which can be simplified to

$$\sum_{j'\in J\cup\emptyset} x_{ij't} \leq \sum_{j'\in J\cup\emptyset} x_{ij'(t-1)}, \quad \forall i \in N, \forall t \in T. \quad (14)$$

The optimization that we formulated above is a binary integer problem. Therefore, as it is known, it is NP-hard to calculate the optimal value in such a case.

III. Online Heuristics

An online algorithm is one that can process its input piece-by-piece in a serial fashion, i.e., in the order that the input is fed to the algorithm, without having the entire input available from the start. In contrast, an offline algorithm is given the whole problem data from the beginning and is required to output an answer which solves the problem at hand. Bearing this in mind, we propose an online heuristic algorithm for solving the offline problem.

A. Assumptions and Cost Function

We consider that the algorithm needs only the information of the current time-slot t and the previous time-slot $t-1$. We denote by $\mathcal{J}(t)$ the jobs in the system in time-slot t, the available servers $\mathcal{S}_{\mathrm{ON}}(t)$, the unavailable servers $\mathcal{S}_{\mathrm{OFF}}(t)$, the servers being in the activation process $\mathcal{S}_A(t)$ in time-slot t. Let us also define the *jobs-to-server ratio* per time-slot as following: $|\mathcal{J}(t)|/(|\mathcal{S}_{\mathrm{ON}}(t)| + |\mathcal{S}_A(t)|)$. Moreover, n_J is the number of jobs to be accumulated in order that we can activate an OFF server. We assume that

1) if server i is not assigned to a job in time-slot t, then the algorithm starts the waiting time $\tau_W(i,t)$ for server i;
2) if $\tau_W(i,t) \geq t_{\mathrm{wait}}$, then the algorithm switches server i from ON to OFF where t_{wait} is the maximal wait time;
3) if $|\mathcal{J}(t)|/(|\mathcal{S}_{\mathrm{ON}}(t)| + |\mathcal{S}_A(t)|) \geq n_J$, then activate an OFF server;

Let $\delta_j(t)$ represent *total delay* of job j until time-slot t since its first occurrence in the system. Let us consider the following cost of assigning server i to job j in time-slot t:

$$c_{ijt} = E_{\mathrm{slot}} + \exp\left(\frac{\delta_j(t-1)-\Delta_j}{\tau}\right)\left(w_j(t-1) - s_i\tau\right)^+ \quad (15)$$

where by $(\delta_j(t-1)-\Delta_j)/\tau$, we normalize the difference by time-slot length τ letting to characterize the cost of delay. On the other hand, $(w_j(t-1)-s_i\tau)^+$ is a monotonically decreasing function with respect to the increasing values of time-slots. The cost of not assigning a server to job j in time-slot t can thus be given by

$$c_{\emptyset jt} = \exp\left(\frac{\delta_j(t-1)-\Delta_j}{\tau}\right) w_j(t-1). \quad (16)$$

If no job is assigned to server i in time-slot t, the cost is only the energy consumed during a time-slot $c_{i\emptyset t} = E_{\mathrm{slot}}$. We set $c_{\emptyset\emptyset t} = 0$ if no server is assigned to no job.

B. Total Cost Minimization per Slot

Consider that we would like to calculate the total minimal cost per time-slot based on the above cost definitions. We below show that this problem is exactly the *classical assignment problem* being solvable in polynomial time. Denote as the number of ON servers and the jobs in time-slot t as $|\mathcal{S}_{\mathrm{ON}}(t)|$ and $|\mathcal{J}(t)|$, respectively. Recall that a server may not be assigned to a job as well as no server can be assigned to a job. Consider some "virtual servers" of which number is equal to $|\mathcal{J}(t)|$ in addition to ON servers in time-slot t. We represent by $\emptyset_i$, the ith virtual server. Besides, consider some $|\mathcal{S}_{\mathrm{ON}}(t)|$ "virtual jobs" in addition to the jobs in time-slot t. We show by $\emptyset^j$, the jth virtual job. We define those virtual elements that will substitute the empty set in formulations. Thus, we have totally $|\mathcal{S}_{\mathrm{ON}}(t)| + |\mathcal{J}(t)|$ both ON servers and jobs in time-slot t. Below, an example is given. We define $c_{i\emptyset^j t} = c_{i\emptyset t}$, $\forall j \in \{1,2,\ldots,|\mathcal{S}_{\mathrm{ON}}(t)|\}$ as well as $c_{\emptyset_i jt} = c_{\emptyset jt}$, $\forall i \in \{1,2,\ldots,|\mathcal{J}(t)|\}$. It is obvious that $c_{\emptyset_i\emptyset^j t} = c_{\emptyset\emptyset t} = 0$, $\forall i \in \{1,2,\ldots,|\mathcal{J}(t)|\}$ and $\forall j \in \{1,2,\ldots,|\mathcal{S}_{\mathrm{ON}}(t)|\}$. The total minimal cost in time-slot t can be calculated by following binary integer program:

$$\min_{x} \sum_{\tilde{\mathcal{S}}_{\mathrm{ON}}(t)} \sum_{\tilde{\mathcal{J}}(t)} c_{ijt} x_{ijt} \text{ subject to}$$
$$\sum_{j \in \tilde{\mathcal{J}}(t)} c_{ijt} x_{ijt} = 1, \quad \forall i \in \tilde{\mathcal{S}}_{\mathrm{ON}}(t),$$
$$\sum_{i \in \tilde{\mathcal{S}}_{\mathrm{ON}}(t)} c_{ijt} x_{ijt} = 1, \quad \forall j \in \tilde{\mathcal{J}}(t),$$
$$x_{ijt} \in \{0,1\}, \quad \forall i \in \tilde{\mathcal{S}}_{\mathrm{ON}}(t), \forall j \in \tilde{\mathcal{J}}(t) \tag{17}$$

where $\tilde{\mathcal{S}}_{\mathrm{ON}}(t) = \mathcal{S}_{\mathrm{ON}}(t) \cup \{\emptyset_1, \ldots, \emptyset_{|\mathcal{J}(t)|}\}$ and $\tilde{\mathcal{J}}(t) = \mathcal{J}(t) \cup \left\{\emptyset^1, \ldots, \emptyset^{|\mathcal{S}_{\mathrm{ON}}(t)|}\right\}$. The formulation corresponds exactly to the *classical assignment problem*. The optimal solution of any assignment problem can be found using well-known *Hungarian algorithm*.

IV. Online Algorithm

Let us introduce the variables and sets used in the algorithm:

- $\mathcal{S}_{\mathrm{ON}}(t)$: set of available servers in time-slot t
- $\mathcal{S}_{\mathrm{OFF}}(t)$: set of unavailable servers in time-slot t
- $\mathcal{S}_A(t)$: set of servers in activation process in time-slot t
- $\tau_A(i,t)$: time spent until time-slot t since activation of server i
- $\tau_W(i,t)$: waiting time until time-slot t of server i
- $\tilde{\tau}_A(t)$: vector of time spent until time-slot t since activation
- $\tilde{\tau}_W(t)$: vector waiting time until time-slot t
- $b_W(i,t)$: the flag bit showing if server i should be in waiting state in time-slot t
- $\tilde{b}_W(t)$: vector of the flag bits
- $\delta_j(t)$: total delay of job j until time-slot t since its first occurrence in the system
- $w_j(t)$: remained demand of job j in time-slot t
- $A(t)$: assignment matrix in time-slot t

We show by $f(\cdot|step\ k)$ a set or variable evolved within the algorithm in step k. The algorithm works step by step as described in Algorithm 1.

Algorithm 1: *Online Algorithm*

```
1.  forall $s \in \mathcal{S}_A(t)$,
      if $\tau_A(s,t) \geq t_{\mathrm{setup}}$
        a) Add server $s$ to $\mathcal{S}_{\mathrm{ON}}(t)$, i.e. $\mathcal{S}_{\mathrm{ON}}(t|\text{step } 4) = \mathcal{S}_{\mathrm{ON}}(t) \cup s$
        b) Remove server $s$ from $\mathcal{S}_A(t)$, i.e. $\mathcal{S}_A(t|\text{step } 4) = \mathcal{S}_A(t) \setminus s$
        c) Remove $\tau_A(s,t)$ from $\tilde{\tau}_A(t)$: $\tilde{\tau}_A(t|\text{step } 4) = \tilde{\tau}_A(t) \setminus \tau_A(s,t)$
        d) set $b_W(s,t|\text{step } 4) = 0$
        e) set $\tau_W(s,t|\text{step } 4) = 0$
      endif
    endfor
2.  forall $s \in \mathcal{S}_{\mathrm{ON}}(t|\text{step } 4)$,
      if $b_W(s,t|\text{step } 4) = 1$ and $\tau_W(s,t|\text{step } 4) \geq t_{\mathrm{wait}}$
        a) set $\mathcal{S}_{\mathrm{OFF}}(t|\text{step } 5) = \mathcal{S}_{\mathrm{OFF}}(t) \cup s$
        b) set $\mathcal{S}_{\mathrm{ON}}(t|\text{step } 5) = \mathcal{S}_{\mathrm{ON}}(t|\text{step } 4) \setminus s$
        c) set $\tilde{b}_W(t|\text{step } 5) = \tilde{b}_W(t|\text{step } 4) \setminus b(s,t|\text{step } 4)$
        d) set $\tilde{\tau}_W(t|\text{step } 5) = \tilde{\tau}_W(t|\text{step } 4) \setminus \tau(s,t|\text{step } 4)$
      endif
    endfor
3.  if $\frac{|\mathcal{J}(t)|}{|\mathcal{S}_{\mathrm{ON}}(t)|+|\mathcal{S}_A(t)|} \geq n_J$,
      a) set $\mathcal{S}_{\mathrm{OFF}}(t|\text{step } 6)$ = Remove $|\mathcal{J}(t)| - |\mathcal{S}_{\mathrm{ON}}(t|\text{step } 5)| - |\mathcal{S}_A(t|\text{step } 4)|$ servers from $\mathcal{S}_{\mathrm{OFF}}(t|\text{step } 5)$
      b) set $\mathcal{S}_A(t|\text{step } 6) = \mathcal{S}_A(t|\text{step } 4) \cup \{\mathcal{S}_{\mathrm{OFF}}(t|\text{step } 4) \setminus \mathcal{S}_{\mathrm{OFF}}(t|\text{step } 5)\}$
      forall $s \in \mathcal{S}_A(t|\text{step } 6)$
        a) set $\tilde{\tau}_A(t|\text{step } 6) = \tilde{\tau}_A(t|\text{step } 4) \cup \tau_A(s,t)$
      endfor
      c) set $\tilde{\tau}_A(t|\text{step } 6) = \tilde{\tau}_A(t|\text{step } 6) + \tau$
    else
      a) set $\tilde{\tau}_A(t|\text{step } 6) = \tilde{\tau}_A(t|\text{step } 4) + \tau$
    endif
4.  set cost matrix $C(t)$
5.  Calculate optimal assignments for time-slot $t$:
    $A(t) = \text{Hungarian}[C(t)]$
6.  set $\mathcal{S}_{\mathrm{TEMP}} = \emptyset$.
    forall $s \in \mathcal{S}_A(t|\text{step } 6)$,
      if server $s$ is not assigned to a job in time-slot $t$,
        a) add server $s$ to $\mathcal{S}_{\mathrm{TEMP}}$, i.e. $\mathcal{S}_{\mathrm{TEMP}} = \mathcal{S}_{\mathrm{TEMP}} \cup s$
      else
        if $b_W(s,t|step\ 5) = 1$,
          a) set $b_W(s,t|step\ 8) = 0$
          b) set $\tau_W(s,t|step\ 8) = 0$
        endif
      endif
    endfor
7.  Choose randomly a server $i \in \mathcal{S}_{\mathrm{TEMP}}$
    if $b_W(i,t|\text{step } 5) = 0$
      a) set $b_W(i,t|\text{step } 9) = 1$
      b) set $\tau_W(i,t|\text{step } 9) = \tau$
    else
8.    a) set $\tau_W(i,t|\text{step } 9) = \tau_W(i,t|\text{step } 5) + \tau$
    endif
9.  if server $i \in \mathcal{S}_A(t|\text{step } 6)$ is assigned to job $j \in \mathcal{J}(t)$ in time-slot $t$
      a) $w_j(t+1) = w_j(t) - s_i\tau$
    else
      a) $w_j(t+1) = w_j(t)$
10. forall $j \in \mathcal{J}(t)$
      a) set $\delta_j(t+1) = \delta_j(t) + \tau$
    endfor
```

V. Possible Enhancements in Online Algorithm

In the current version of online algorithm, the decision of optimal assignments is performed only taking into account the available information preceding two subsequent slots; that can be improved by using some statistical inference techniques where the assignment cost's exponential part given in equation 15 will be updated dynamically. On the other hand, jobs-to-servers ratio can be transformed to total service demand per total server speed in the following way: $\sum_{j \in \mathcal{J}(t)} w_j(t) / \sum_{i \in \mathcal{S}_{\mathrm{ON}}(t)} s_i$. So, the activation of a new server could be decided whenever $\sum_{j \in \mathcal{J}(t)} w_j(t) / \sum_{i \in \mathcal{S}_{\mathrm{ON}}(t)} s_i \geq \min_{j \in \mathcal{J}(t)} \left(t_j + \Delta_j - t\right)$. Another enhancement may be to define a new criterion of activation or switching OFF a server which takes into consideration periodically the average number of jobs, service demand, and deadline of jobs. For example, the distribution of hourly number of jobs, service demand, and deadline of jobs is found and applied in the criterion.

TABLE I
COMPARISON OF ONLINE TOTAL ENERGY COST WITH RANDOM, OPTIMAL AND RELAXED OPTIMAL TOTAL ENERGY COST. $|N| = 3$, $|J| = 8$.

Example	$[s_i]_{i\in N}$	$\left[w_j\right]_{j\in J}$	$\left[t_j\right]_{j\in J}$	$\left[\Delta_j\right]_{j\in J}$	**Online**	**Random Assignment**	**Optimal**	**Relaxed**
1	[4 2 2]	[4 1 2 5 5 5 1 3]	[2 2 3 3 3 5 5 5]	[3 4 2 2 4 4 4 3]	3500	543960	2200	1300
2	[2 2 4]	[1 1 5 3 2 1 4 1]	[2 2 2 3 4 4 5 6]	[2 4 2 2 4 3 2 3]	2940	93380	1800	900
3	[4 3 2]	[3 1 2 1 3 2 3 2]	[2 2 3 4 4 6 6 6]	[3 2 3 3 4 4 2 4]	3440	137780	1600	850
4	[4 4 4]	[4 4 2 3 3 3 5 2]	[2 2 2 2 4 6 6 6]	[2 3 3 2 2 4 2 4]	137400	138880	1800	1300
5	[4 3 4]	[4 3 1 4 3 1 2 4]	[2 2 3 4 4 4 6 6]	[4 3 4 4 2 2 4 4]	3800	273400	1600	1100

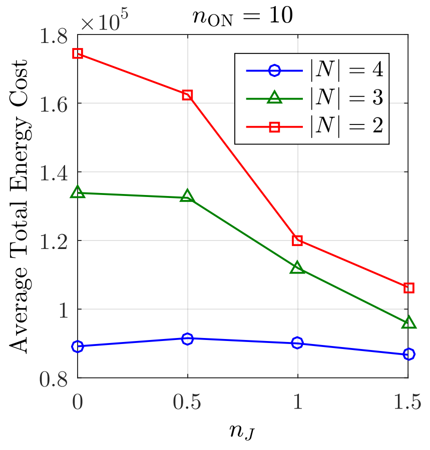

Fig. 1. Average total energy cost with respect to n_J.

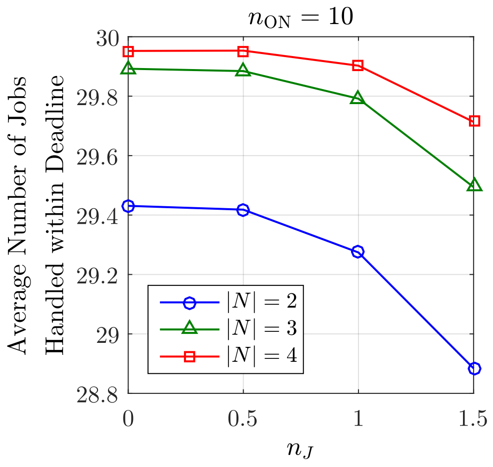

Fig. 2. Average number of jobs handled within deadline with respect to n_J.

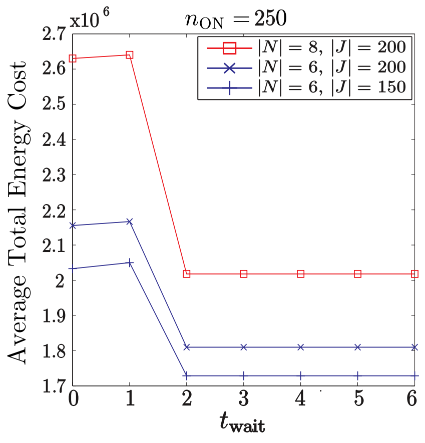

Fig. 3. Average total energy cost with respect to t_{wait}.

TABLE II
COMPARISON OF ONLINE TOTAL ENERGY COST WITH RANDOM ONLINE TOTAL ENERGY COST FOR DIFFERENT VALUES OF PARAMETERS. EXAMPLE 4 IN TABLE 1 IS STUDIED. (OA = ONLINE ALGORITHM, RA = RANDOM ASSIGNMENT)

				$t_{\text{wait}} = 1$				
n_J	1	2	3	4	5	6	7	8
OA	137400	92600	92600	47400	47400	47400	47400	47400
RA	138880	138160	228200	92600	137800	183000	47400	183000
				$t_{\text{wait}} = 2$				
n_J	1	2	3	4	5	6	7	8
OA	3600	3600	3600	3600	3600	3600	3600	3600
RA	5400	4200	4200	3600	4200	6600	4800	4200

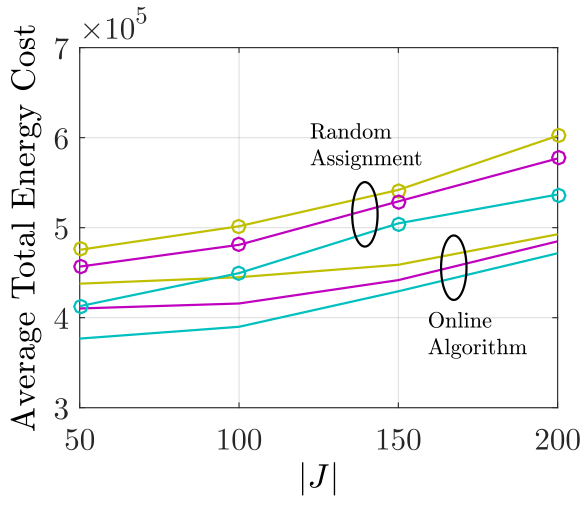

Fig. 4. Average total energy cost with respect to $|J|$.

VI. NUMERICAL RESULTS

We introduce *random assignment algorithm* which is similar to online algorithm but having a random assignment matrix in step 8. Random assignment matrix is obtained by random permutations. In each numerical result, we assume that $\tau = 1$ second, $E_{\text{slot}} = 200$ Jouls, $E_{\text{ON}} = 160$ Jouls.

Small Size Examples: In Table 1 and Table 2, the speed of servers, service demands of jobs, arrival slots of jobs, deadlines of jobs follows a uniform distribution taking integer values from $[1,4]$, $[1,5]$, $[2,6]$, $[1,4]\times\tau$, respectively. We assume that all servers are ON, initially.

In Table 1, we assume that $t_{\text{wait}} = 1$ second, $n_J = 1$, $n_{\text{ON}} = 250$. The comparison in Table 2 is based on $|N| = 3$ servers and $|J| = 8$ jobs. Note that even in case of a small size example, the random assignment algorithm is far from optimal cost. In Example 4, an extreme case occurs where none server is ON when 4th job arrives to the system in time-slot 5. This happens because of our choice of $t_{\text{wait}} = 1$ second which causes to switch OFF all servers. Since $n_{\text{ON}} = 250$, the activation of a server increases the total energy cost. In Table 2, for different values of t_{wait} and n_J, we compare online total energy cost with random online total energy cost. Note that when t_{wait} increases, both algorithms do better. Increasing n_J could also improve the performance of online algorithm; but, it is not always guaranteed for random assignment algorithm.

Critical Values of n_J and t_{wait}: In Figure 1 and Figure 2, the

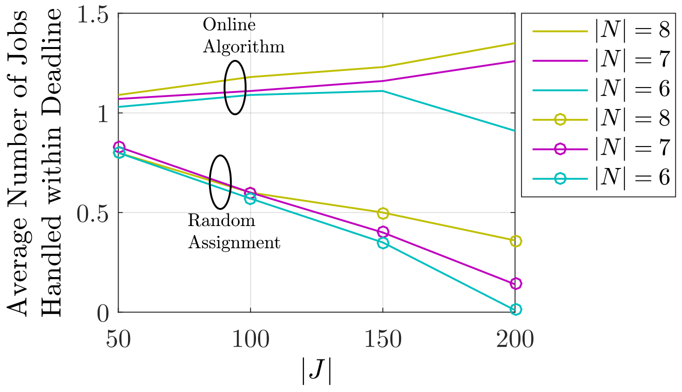

Fig. 5. Average number of jobs handle within deadline with respect to $|J|$.

following parameters are assumed: $t_{\text{wait}} = 2$ seconds, $n_J = 5$, $n_{\text{ON}} = 10$, and $|J| = 30$. The speed of servers, service demands of jobs, arrival slots of jobs, deadlines of jobs, initial states of servers follow a uniform distribution taking integer values from $[2,4]$, $[10,20]$, $[2,200]$, $[10,20] \times \tau$, $[0,1]$, respectively.

In Figure 1, average total energy cost with respect to n_J is depicted for $|N| = 1$, 2 and 3. The figure implies that increasing values of n_J tends to result in a positive effect on average total energy cost; but, it also shows that for $|N| = 4$, the curve is not monotonically decreasing with respect to n_J.

In Figure 2, average number of jobs handled within deadline with respect to n_J is depicted for $|N| = 1$, 2 and 3. It is inevitable that average number of jobs handled within deadline decreases with increasing values of n_J.

In Figure 3, we plot how average total energy cost changes with respect to t_{wait}. We again assume that $n_{\text{ON}} = 250$. Figure 3 implies that the average total energy could be very dependent on t_{wait}. We see that in the considered default values, for $t_{\text{wait}} \geq 2$, the average total energy cost decreases dramatically. Changing the number of servers and jobs do not affect the critical value of t_{wait}.

Comparison of Online Algorithm with Random Assignment: In Figure 4 and Figure 5, the following parameters are assumed: $t_{\text{wait}} = 2$ seconds, $n_J = 5$, $n_{\text{ON}} = 250$. Figure 4 plots the change of average total energy cost as well as Figure 5 depicts the change of average number of jobs handled within deadline with respect to number of jobs for $|N| = 6$, 7, and 8. The advantage of online algorithm is inevitable in the figures.

VII. Conclusions and Future Work

We studied the dynamic energy optimization problem in data centers. The corresponding offline problem has been formulated and solved using binary integer programming. We showed that the offline problem is a new version of dynamic generalized assignment problem including new constraints issuing from deadline characteristics of jobs and difference of activation energy of servers. Furthermore, we proposed an online algorithm that solves the problem heuristically and compared it to random assignment.

As an immediate extension in the offline problem, we can take into account more than two states of the servers, i.e. instead of only ON and OFF modes, we can also consider IDLE, HIBERNATE modes, etc. We can introduce a new time cost in case of a preemption which corresponds to migration a job from a server to another one. Moreover, another constraint can be geographically separated servers as well as we can define hierarchy among jobs and forbid a particular job to be assigned to a particular server. A possible work could also discuss to do flexibility by some factor in deadline constraints of jobs.

References

[1] A. Gandhi, "Dynamic server provisioning for data center power management," *PhD Thesis*, Carnegie Mallon University, Pittsburgh, 2013.

[2] L. A. Barroso and U. Hlzle, "The case for energy-proportional computing," *IEEE Computer*, vol. 40, no. 12, pp. 33-37, 2007.

[3] K. Choi, R. Soma and M. Pedram, "Fine-grained dynamic voltage and frequency scaling for precise energy and performance trade-off on the ratio of off-chip access to on-chip computation times," *in Proceedings of the Conference on Design, Automation and Test in Europe, DATE'04*, Paris, France, 2004.

[4] C.-H. Hsu and W.-C. Feng, "A power-aware run-time system for high-performance computing," *in Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, SC'05*, Seattle, WA, USA, 2005.

[5] H. Amur and K. Schwan, "Achieving power-efficiency in clusters without distributed," *in Workshop on Energy Efficient Design*, Saint-Maolo, France, 2010.

[6] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl and R. Gupta, "Augmenting network interfaces to reduce PC energy usage," *in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, NSDI'09*, Boston, MA, USA, 2009.

[7] Q. Deng, D. Meisner, A. Bhattacharjee, T. F. Wenisch and R. Bianchini, "CoScale: Coordinating CPU and memory system DVFS in server systems," *in Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO'12*, British Columbia, Canada, 2012.

[8] A. Gandhi, M. Harchol-Balter, R. Raghunathan and M. A. Kozuch, "Distributed, robust auto-scaling policies for power management in compute intensive server farms," *in IEEE Sixth Open Cirrus Summit (OCS)*, 2011, Atlanta, GA, USA, Oct. 2011.

[9] M. Lin, A. Wierman, L. Andrew and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *in Proceedings of 2011 IEEE INFOCOM, INFOCOM'11*, Shanghai, China, 2011.

[10] B. Urgaonkar, P. Shenoy and T. Roscoe, "Resource overbooking and application profiling in shared hosting platforms," *SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 239-254, December 2002.

[11] A. Verma, P. Ahuja and A. Neogi, "pMapper: power and migration cost aware application placement in virtualized systems," *in Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, Middleware'08*, Leuven, Belgium, 2008.

[12] R. Nathuji, A. Kansal and A. Gha, "Q-clouds: Managing performance interference effects for QoS-aware clouds," *in Proceedings of the 5th European Conference on Computer Systems, EuroSys'10*, Paris, France, 2010.

[13] N. Bansal, T. Kimbrel, K. Pruhs, "Speed scaling to manage energy and temperature," *Journal of ACM*, vol. 54 no. 1, 2007.

[14] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, M. Sviridenko, "Buffer overflow management in QoS switches," *SIAM J. Comput.*, vol. 33 no. 3, pp. 563-583, 2004.

[15] P. Chuprikov, S. I. Nikolenko, K. Kogan, "On demand elastic capacity planning for service auto-scaling," *IEEE INFOCOM*, 2016.

[16] P. Chuprikov, S. I. Nikolenko, K. Kogan, "Priority queueing with multiple packet characteristics," *IEEE INFOCOM*, 2015.

[17] S. I. Nikolenko, K. Kogan, "Single and multiple buffer processing," *Encyclopedia of Algorithms* , pp. 1988-1994, 2016.

[18] K. Kogan and A. Shtub, "DGAP - The dynamic generalized assignment problem," *Annals of Operations Research*, vol. 69, no. 0, pp. 227-239, 1997.

[19] L. Moccia, J.-F. Cordeau, M. F. Monaco and M. Sammarrab, "A column generation heuristic for a dynamic generalized assignment problem," *Computers & Operations Research*, vol. 36, no. 9, pp. 2670-2681, 2009.