Thermal-Aware Scheduling of Batch Jobs in Geographically Distributed Data Centers

Marco Polverini, Antonio Cianfrani, Shaolei Ren, *Member, IEEE*, and Athanasios V. Vasilakos

Abstract—Decreasing the soaring energy cost is imperative in large data centers. Meanwhile, limited computational resources need to be fairly allocated among different organizations. Latency is another major concern for resource management. Nevertheless, energy cost, resource allocation fairness, and latency are important but often contradicting metrics on scheduling data center workloads. Moreover, with the ever-increasing power density, data center operation must be judiciously optimized to prevent server overheating. In this paper, we explore the benefit of electricity price variations across time and locations. We study the problem of scheduling batch jobs to multiple geographically-distributed data centers. We propose a provably-efficient online scheduling algorithm—GreFar—which optimizes the energy cost and fairness among different organizations subject to queueing delay constraints, while satisfying the maximum server inlet temperature constraints. GreFar does not require any statistical information of workload arrivals or electricity prices. We prove that it can minimize the cost arbitrarily close to that of the optimal offline algorithm with future information. Moreover, we compare the performance of GreFar with ones of a similar algorithm, referred to as T-unaware, that is not able to consider the server inlet temperature in the scheduling process. We prove that GreFar is able to save up to 16 percent of energy-fairness cost with respect to T-unaware.

Index Terms—Data center, scheduling, resource management, thermal Aware, energy

1 Introduction

With the emergence of cloud computing services, there has been a growing trend toward large-scale and geographically distributed data centers. Many megawatts of electricity are required to power such data centers, and companies like Google and Microsoft spend a large portion of their overall operational costs (e.g., tens of millions of dollars) on electricity bills [1].

Better energy efficiency of servers and lower electricity prices are both important in reducing the energy cost. Given the heterogeneity of servers in terms of energy efficiency and the diversity of electricity prices over geographically distributed data centers and over time, the key idea is to preferentially shift power draw to energy-efficient servers and to places and times offering cheaper electricity prices. Moreover, with the ever-increasing power density generating an excessive amount of heat, thermal management in data centers is becoming imperatively important for preventing server overheating that could potentially induce server damages and huge economic losses [2]. As a consequence, optimally distributing the workloads to facilitate heat recirculation and avoid overheating has to be incorporated in data center operation. Nonetheless, reducing energy cost by means of overloading servers in data centers where the electricity price is low may not be a viable solution, since this may result in a higher server temperature that imposes serious concerns for system reliability.

In addition to reducing energy cost, satisfying fairness and delay constraints is also important. For example, when multiple organizations or groups of users share the resources, allowing jobs from a few users to run first while deferring other jobs for lower electricity prices may result in reduced energy cost, but may sacrifice the response time of other users and adversely affect fairness among users. Since the performance metrics of energy, fairness and delay are often contradicting, it is desirable to have a tunable system with the flexibility to meet different business requirements.

In this paper, we consider the problem of developing an online scheduler that distributes batch workloads geographically across multiple data centers and to heterogeneous servers for minimizing energy cost taking into account energy prices, fairness consideration, delay requirements and the maximum server inlet temperature constraints. This is a challenging problem because data centers experience time-varying workloads, server availability and electricity prices with possibly unknown statistics and/or non-stationary distributions. Even though some statistics may be estimated or predicted, applying traditional optimization techniques such as dynamic programming to compute the globally optimal solution can be time consuming and hence, it is not applicable for online schedulers in practice [6].

We propose a practical yet provably-efficient online scheduling algorithm "GreFar" to solve this problem. Our algorithm does not require any prior knowledge of the system statistics (which can even be non-stationary) or any

- *M. Polverini and A. Cianfrani are with the DIET Department, University of Roma—La Sapienza, Roma, Italy. E-mail: {polverini, cianfrani}@diet.uniroma1.it.*
- *S. Ren is with the SCIS, Florida International University, Miami, FL 33199. E-mail: sren@fiu.edu.*
- *A.V. Vasilakos is with the National Technical University of Athens, Athens 10680, Greece. E-mail: vasilako@ath.forthnet.gr.*

Manuscript received 24 May 2013; revised 19 Oct. 2013; accepted 7 Dec. 2013; date of publication 19 Dec. 2013; date of current version 30 Apr. 2014. Recommended for acceptance by C.A. Varela.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCC.2013.2295823

prediction on future job arrivals and server availability. Moreover, it is computationally efficient and easy to implement in large practical systems. GreFar constructs and solves an online optimal problem based on the current job queue lengths, server availability and temperature, and electricity prices; the solution is proven to offer close to the offline optimal performance with future information. More precisely, given a cost-delay parameter $V \geq 0$, GreFar is $O(1/V)$-optimal with respect to the average (energy-fairness) cost against the offline optimal algorithm while bounding the queue length by $O(V)$. Without considering fairness, the energy-fairness cost solely represents the energy cost, while with fairness taken into account, it is an affine combination of energy cost and fairness score (which is obtained through a fairness function). Furthermore, our algorithm is associated with two control parameters, i.e., cost-delay parameter and energy-fairness parameter, which can be appropriately tuned to provide a desired performance tradeoff among energy, fairness and queueing delay.

To complement the analysis, we conduct a simulation study to evaluate our algorithm. Our results show that: (1) GreFar effectively reduces the energy cost (at the expense of increased delay) by opportunistically processing batch jobs using energy-efficient servers and when electricity prices are sufficiently low; (2) With an appropriate energy-fairness parameter, GreFar achieves much higher fairness while only incurring a marginal increase in energy cost; (3) GreFar is flexible in achieving a desirable tradeoff between energy cost, fairness and queueing delay; (4) Thanks to its ability to schedule jobs while satisfying the maximum server inlet temperature constraints, GreFar is able to improve performance with respect to its primitive version presented in [3].

With respect to the algorithm proposed in [3], we make several improvements in this paper. The most significant one is that we incorporate a thermal-aware model, so that our new algorithm is able to take its scheduling decisions keeping the maximum temperature reached by each server under control. As the results show, this ability leads to an improvement in cost saving. Moreover, we significantly extend the performance evaluation, considering the new analysis as well as the impact of workloads on the average cost.

The remainder of the paper is organized as follows. Section 2 discusses related work. Section 3 describes the system, job and scheduling models. Sections 4 and 5 presents the online algorithm and its analysis. In Section 6 the scenario used for simulation is introduced, while Section 7 shows the performance evaluation results. Finally, Section 8 concludes the paper.

2 Related Work

We provide a snapshot of the related work from the following aspects:

Electricity cost variations. There has been a growing interest in cutting electricity bills for large data centers. Several prior studies explore the opportunity of energy saving by executing jobs when and/or where the electricity prices are low (e.g., [7], [8], [9], [10], [11]). Among them, some perform local optimization at each time period without considering the electricity variations across time periods; thus, they do not offer performance guarantees for the average energy cost or queueing delay over a large time horizon (e.g., [7], [8]). Some prior studies assume that the electricity price variations and/or job arrivals follow certain stationary (although possibly unknown) distributions ([9], [10], [11]). In practice, the job arrivals may not follow any stationary distributions, especially in an enterprise computing environment where different organizations only submit job requests sporadically.

There are other related studies on scheduling workloads across multiple data centers. Lin et al. [5] propose an online right-sizing algorithm which dynamically turns on/off servers to minimize the delay plus energy cost, under the assumption that the electricity price is fixed over time. Guenter et al. [6] consider a similar problem but propose to predict the future service demand using a Markov chain to determine the number of active servers. Later, Buchbinder et al. [4] study the problem by taking the bandwidth cost into consideration; the proposed solution does not address the queueing delay requirement. Qureshi et al. [1] quantify the economic gains by scheduling workloads across multiple data centers, which is an empirical study without providing analytical performance bounds on the proposed scheduling algorithm.

Temperature-aware resource management. Temperature-aware (or thermal-aware) resource management in data centers has recently attracted much research interest. In general, temperature-aware resource management can be classified as temperature-reactive and temperature-proactive approaches. In temperature-reactive approaches, decisions are made reactively to avoid server overheating based on the observed real-time temperature. For example, [20] dynamically schedules workloads to cool servers based on the instantaneously observed temperature to avoid server overheating; [19] optimally allocates power in a data center for MapReduce workloads by considering the impacts of real-time temperature on the server performance; [2] presents a cyber-physical approach for monitoring the real-time temperature distribution in data centers, facilitating temperature-reactive decisions. Unlike temperature reactive approaches, temperature-proactive approaches explicitly take into account the impact of resource management decisions on the temperature increase. Nonetheless, directly using computational fluid dynamics (CFD) models for temperature prediction incurs a prohibitive complexity [17], leading to the development of a less complex yet sufficiently accurate heat transfer model, as shown in [18]. For example, without considering delays, [13] maximizes the total capacity of a data center subject to temperature constraint based on this model; [12], [14] develop software/hardware architectures for various types of thermal management in high-performance computing environments, e.g., minimizing the peak inlet temperature through task assignment to alleviate the cooling requirement.

Compared with the existing research, GreFar does not require any prior knowledge or assume any (stationary) distributions of the system dynamics. To the best of our knowledge, GreFar is the first provably-efficient solution that minimizes the energy-fairness cost with temperature constraint and queueing delay guarantees under arbitrarily time-varying system dynamics (e.g., non-stationary random server availability, electricity prices, job arrivals).

3 Problem Formulation

This section describes the data center system model, job model and our scheduling model. We consider a system with time indexes $t = \{0, 1, 2, \ldots\}$, where each t represents a scheduling instant.

3.1 Data Center System Model

There are N geographically distributed data centers, each of which houses thousands of servers. Servers may be homogeneous or heterogeneous in hardware and performance characteristics. One major source of heterogeneity is that data centers operate several generations of servers from multiple vendors. Application needs, hardware innovations and prices jointly determine which type of servers to purchase. Our model accommodates both homogeneous and heterogeneous servers. Next, we specify the *state* of each data center, which is time-varying and captures the randomness in the environment.

3.1.1 Server Availability

Server availability may change over time due to different reasons such as server failures, software upgrades, influence of other workloads, etc. For example, the increase of interactive workloads may reduce the number of servers available to process batch jobs. In data center i at time t, we consider without loss of generality, that there are $K_i(t)$ (possibly heterogeneous) available servers. In data center i, each server k, for $k = 1, 2, \ldots, K_i(t)$ is characterized by three parameters: processing speed $s_{i,k}(t)$, idle power $\tilde{p}_{i,k}(t)$ and active power $p_{i,k}(t)$, where $p_{i,k}(t) > \tilde{p}_{i,k}(t)$.[1] To make the notation more concise, we use the vectorial expression $\mathbf{s}_i(t) = [s_{i,1}(t), s_{i,2}(t), \ldots, s_{i,K_i(t)}(t)]$, $\tilde{\mathbf{p}}_i(t) = [\tilde{p}_{i,1}(t), \tilde{p}_{i,2}(t), \ldots, \tilde{p}_{i,K_i(t)}(t)]$, and $\mathbf{p}_i(t) = [p_{i,1}(t), p_{i,2}(t), \ldots, p_{i,K_i(t)}(t)]$.

3.1.2 Electricity Price

Due to the deregulation of electricity markets, electricity prices stochastically vary over time (e.g., every hour or 15 minutes) and across different locations [9], [26], [22]. We use $\varphi_i(t)$ to denote the electricity price in data center i during the time t. The function $\varphi_i(t)$ maps the total energy usage in data center i during time t to the electricity price. For the ease of presentation, we consider a particular example of $\varphi_i(t)$ which is a constant electricity price regardless of the actual energy usage during time t. The constant electricity price during a time period has been widely studied in prior works (e.g., [5]). However, our model and analysis are still applicable when the total electricity cost is not a linear function of the energy consumption. For example, the electricity cost can be an increasing and convex (or other) function of the energy consumption [4]. In such scenarios, the amount of other workloads such as interactive workloads also affects the energy price, and hence, we need to add another component, i.e., energy consumed by other workloads during each time t, into the data center state.

1. Other server states, such as "low performance", can also be captured if we include the server state selection into the scheduling decisions (detailed in Section 3.3).

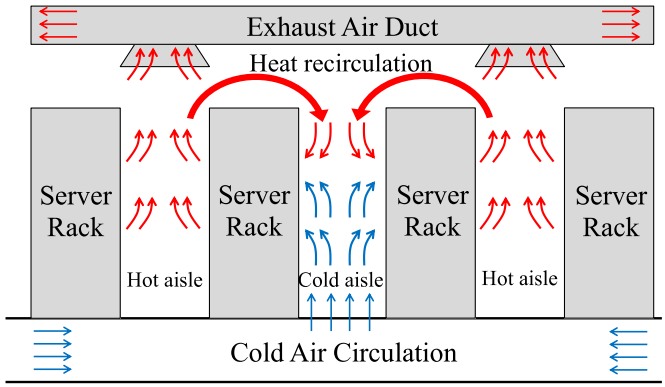

Fig. 1. System diagram.

3.1.3 Inlet Temperature

In a data center, a significant portion of the server power is eventually dissipated as heat, which in turn increases the server inlet temperature and consequently affects the server performance as well as reliability [13]. Typically, as illustrated in Fig. 1, a computer room air conditioning (CRAC) unit blows cold air into the data center and exhaust air ducts are placed above the servers to remove the hot air.

Due to the fast growing power density, cooling system incurs a significant power consumption in data centers (e.g., over 30 percent of the total power consumption) [14]. While it depends on several factors (e.g., air conditioner efficiency), we model the cooling power consumption (P_i^C) of the ith data center as a function of the servers power consumption (P_i, to be specified later) and of the supply temperature (L_i^{sup}), i.e. the temperature of the air generated by the cooling system. In particular we use the model provided in [12]:

$$P_i^C = \frac{P_i}{CoP\left(L_i^{sup}\right)}, \tag{1}$$

where $CoP(L_i^{sup})$ is referred to as *coefficient of performance* of the cooling system given a temperature of L_i^{sup} for the supplied cold air. The coefficient of performance $CoP(L_i^{sup})$ characterizes the efficiency of the cooling system and, as corroborated by prior work that performs extensive simulation and modeling based on CFD [12], an example of $CoP(L_i^{sup})$ is given as follows:

$$CoP\left(L_i^{sup}\right) = 0.0068\left(L_i^{sup}\right)^2 + 0.0008L_i^{sup} + 0.458. \tag{2}$$

To better understand Eqs. (1) and (2), we note that a lower supply temperature requires more work from the cooling system and hence increase the cooling power consumption. Combining the cooling power and server power, the total power consumption is given by

$$P_i^{EQ} = P_i + P_i^C = P_i\left(1 + \frac{1}{CoP(L_i^{sup})}\right). \tag{3}$$

Intuitively, the inlet temperature of a server depends on the power consumption of the server itself and on the power

consumption of others servers in the data center due to heat recirculation. To take into account this cross-contribution, the Heat Transfer Matrix D is defined. The generic element d_{ij} represents the impact of the j^{th} server on the i^{th} server. In this work we consider the 1D model proposed in [13]: only the servers placed in the same rack affect the inlet temperature of a server.

Based on the abstract heat model [12], the $K_i(t)-$dimension vector of server inlet temperatures in data center i at time t can be expressed as

$$\mathbf{L}_i^{in}(t) = \mathbf{L}_i^{sup}(t) + \mathbf{D}_i(t) \cdot \mathbf{b}_i(t), \tag{4}$$

where $\mathbf{L}_i^{sup}(t)$ is the supply temperature for each server, $\mathbf{D}_i(t)$ is heat transfer matrix, and $\mathbf{b}_i(t)$ is the (average) server power consumption expressed in a $K_i(t)-$dimension vector form where all the servers that are available for processing batch jobs in data center i are represented as elements in the vector. Thus, at any time t, the server inlet temperature constraint can be expressed as

$$\max \mathbf{L}_i^{in}(t) = \mathbf{L}_i^{sup}(t) + \max\left[\mathbf{D}_i(t) \cdot \mathbf{b}_i(t)\right] \leq L_i^{\max}. \tag{5}$$

Based on the above discussion, we can mathematically represent the state of data center i during time t using a tuple $\mathbf{x}_i(t) = \{\mathbf{s}_i(t), \tilde{\mathbf{p}}_i(t), \mathbf{p}_i(t), \varphi_i(t), \mathbf{L}_i^{sup}(t), \mathbf{D}_i(t)\}$, for $i = 1, 2, \ldots, N$. Throughout the paper, we also use the vectorial expression $\mathbf{x}(t) = [\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_N(t)]$ wherever applicable. As we have noted, $\mathbf{x}_i(t)$ is stochastically changing over time and the actual value of $\mathbf{x}_i(t)$ is not revealed until the beginning of time t. Unlike in prior works [9], [10], [11], we do not impose any restriction on the distribution of $\mathbf{x}_i(t)$, such as independent and identically distributed (i.i.d.).

3.2 Job Model

Each job, or service request, is characterized by a tuple $\{d, \mathcal{D}, \rho\}$. We use $d > 0$ to represent the service demand (i.e., job length) in terms of processor cycles;[2] $\mathcal{D}$ represents a subset of data centers $\{1, 2, \ldots, N\}$ that this job can be scheduled to, which often relates to where the job's data is stored; $\rho \in \{1, 2, \ldots, M\}$ (where M is the total number of accounts) is the account from which the job originates, where an account may represent a user, a group of users or an organization. During its execution, a job can be suspended and resumed later.

Depending on the job characteristics, we classify the jobs into $J \geq 1$ types and refer to the jobs belonging to type j as type-j jobs, for $j = 1, 2, \ldots, J$. A type-j job can be fully specified using $\mathbf{y}_j = \{d_j, \mathcal{D}_j, \rho_j\}$. In practice, we can group jobs having approximately the same characteristics into the same type. We denote by $a_j(t)$ the number of arrival jobs of type j during time slot t. The (time-varying) arrival process $\{a_j(t) \in \mathbb{Z}^+,\ j = 1, 2, \ldots, J,\ \text{and}\ t = 0, 1, 2, \ldots\}$ does not necessarily follow any stationary distributions and can be arbitrary, while the only assumption, which is not a limiting factor in practice, is boundedness:

$$0 \leq a_j(t) \leq a_j^{\max}, \tag{6}$$

for a finite positive integer $a_j^{\max}$, $j = 1, 2, \ldots, J$ and $t = 0, 1, 2, \ldots$.

In the model, we assume that jobs of the same type can be processed by any number of servers simultaneously. In practice, however, it may be possible that only a certain number of servers can process jobs of the same type in parallel. Our model can be adapted easily to capture this fact by adding a parallelism constraint for each job type. Specifically, we need to add a constraint on the scheduling decisions such that the maximum number of servers that can be used to process jobs of the same type simultaneously is upper bounded.

3.3 Scheduling Model

3.3.1 Performance Metrics

We consider three important but often contradicting performance metrics in practice: energy cost, fairness and queueing delay.

Energy cost: Energy consumption is one of the largest contributing factor to data center cost. Our scheduler considers time-varying electricity prices across data centers to decide where and when to run jobs to reduce the energy cost. With an average utilization $u_{i,k}(t) \in [0, 1]$ (as determined by the amount of workloads processed by the server), the average power consumption incurred by server k in data center i at time t is given by

$$b_{i,k}(t) = \left[p_{i,k}(t) - \tilde{p}_{i,k}(t)\right] \cdot u_{i,k}(t) + \tilde{p}_{i,k}(t). \tag{7}$$

As a consequence the power consumption of data center i during time t can be mathematically represented by vector $\mathbf{b}_i(t) = [\mathbf{p}_i(t) - \tilde{\mathbf{p}}_i(t)] \otimes \mathbf{u}_i(t) + \tilde{\mathbf{p}}_i(t)$. With a slight abuse of notation, we let $b_{i,k}(t) = 0$ if $u_{i,k}(t) = 0$, or equivalently server k incurs a zero power when it is not processing any workloads (e.g., server k can be turned off, while we neglect the server turning on/off costs because the control decisions are made infrequently). Thus, the total server power consumption in data center i can be expressed as

$$P_i(t) = \sum_{k=1}^{K_i(t)} b_{i,k}(t). \tag{8}$$

Correspondingly, considering a linear energy cost function, the energy cost for cooling systems and processing the scheduled batch jobs in data center i during time t is[3]

$$e_i(t) = \varphi_i(t) \cdot P_i^{EQ}(t) = \varphi_i(t)\left[P_i(t) + P_i^C(t)\right], \tag{9}$$

where $P_i^C(t)$ is given by (1). Thus, the total energy cost during time t can be expressed as $e(t) = \sum_{i=1}^{N} e_i(t)$.

Fairness: Fairness in resource allocation among different accounts is an important concern for data centers. We define a *fairness* function to mathematically characterize the

2. In this study, we use the CPU demand or processor cycles as a consolidated proxy of the job's service demand, while the jobs' demands in terms of memory, storage, etc., can be explicitly considered if we extend the service demand $d > 0$ from a scalar to a vector in which each element corresponds to one type of demand.

3. Each time t has the same duration and hence, we normalize the duration to one and the duration does not appear in the energy cost expression (9).

fairness of resource allocation. There are various fairness functions such as α-fair index [21]. In this paper, we use the following fairness function[4]:

$$f(t) = -\sum_{m=1}^{M}\left[\frac{r_m(t)}{R(t)} - \gamma_m\right]^2, \tag{10}$$

where $r_m(t)$ is the total amount of computing resource allocated to all the jobs from account m during time t, $R(t) = \sum_{i=1}^{N}\sum_{k=1}^{K_i(t)} s_{i,k}(t)$ is the total available computing resource during time t, and $\gamma_m \geq 0$ is the weighting parameter indicating the desire amount of resource allocation to account m. The fairness score is maximized when $r_m(t) = \gamma_m R(t)$ for $m = 1, 2, \ldots, M$.

Queueing delay: Given a job arrival process, queueing delay is closely related to the average number of jobs in the queue. In this paper, we bound the queue length, which in turn determines the average delay performance. For each type of jobs, there are separate queues maintained in each data center. For type-j jobs, we denote the queue lengths at time t in data center i by $q_{i,j}(t)$.

3.3.2 Scheduler Decisions

Control decisions, including job scheduling and server allocation, are made at the beginning of each time t. In practice, the duration of each time t depends on the scheduling quantum. For example, we may consider 1 hour as the duration of a time slot, which correspond to the price updating frequencies in electricity markets [26], [22]. At the beginning of time t, the following decisions are made:

(1) $r_{i,j}(t) \in \mathbb{Z}^+$: the number of type-$j$ jobs scheduled to data center i during time t, for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, j$. A job cannot be split into multiple parts and hence the scheduling decision $r_{i,j}(t)$ takes integer values.

(2) $h_{i,j}(t) \in \mathbb{R}^+$: the number of type-$j$ jobs processed in data center i during time t, for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, J$. Since we allow a job to be paused during its execution, $h_{i,j}(t)$ is not necessarily an integer.

(3) $u_{i,k}(t) \in [0, 1]$: the utilization of server k in data center i during time t, (equivalently, $u_{i,k}(t)s_k(t)$ computing resource is provided by server k). Note that $\sum_{j=1}^{J} h_{i,j}(t)d_j \leq \sum_{k=1}^{K_i(t)} u_{i,k}(t)s_k(t)$ is satisfied, i.e., the processed work cannot exceed the provided computing resource.

For notational convenience, we write the decisions, which we also refer to as *action*, made at the beginning of time t in a compact form as $\mathbf{z}(t) = \{r_{i,j}(t), h_{i,j}(t), u_{i,k}(t),\ i = 1, 2, \ldots, N,\ j = 1, 2, \ldots, J,\ k = 1, 2, \ldots, K_i(t)\}$. Finally, we impose a mild assumption on the scheduling decisions that the following boundedness conditions are satisfied:

$$0 \leq r_{i,j}(t) \leq r_{i,j}^{\max}, \tag{11}$$

$$0 \leq h_{i,j}(t) \leq h_{i,j}^{\max}, \tag{12}$$

for some finite $r_{i,j}^{\max}$ and $h_{i,j}^{\max}$.

4. Our analysis also applies if other fairness functions are considered.

4 Scheduling Algorithm

This section presents an offline optimal formulation of the problem and a provably efficient online algorithm "GreFar".

4.1 Problem Formulation

To jointly consider the energy cost and fairness, we define the instantaneous energy-fairness cost function as follows:

$$g(t) = e(t) - \beta \cdot f(t) = \sum_{i=1}^{N} e_i(t) - \beta \cdot f(t), \tag{13}$$

where $\beta \geq 0$ is a scaler (called energy-fairness parameter) that transfers the achieved fairness into energy cost saving, and $e_i(t)$ and $f(t)$ are given in (9) and (10), respectively. In special cases, when $\beta = 0$, the scheduler does not consider the fairness in resource allocation, whereas when $\beta \to \infty$, the scheduler does not consider the energy cost. Affine combination of two performance metrics is a common approach in multi-objective optimization (e.g., [5], [7], [24]). Moreover, β is equivalent to the corresponding Lagrangian multiplier if we formulate the fairness as a constraint [23].

For data center operation, minimizing the time-average cost is crucial in cutting electricity bills. Let $\overline{g}$ be the time average cost of $g(t)$ under a particular control policy implemented over a sufficiently large but finite time horizon with t_{end} time slots:

$$\overline{g} \triangleq \frac{1}{t_{end}} \sum_{\tau=0}^{t_{end}-1} g(\tau). \tag{14}$$

Similarly, we define

$$\overline{a}_i \triangleq \frac{1}{t_{end}} \sum_{\tau=0}^{t_{end}-1} a_i(\tau), \overline{r}_{i,j} \triangleq \frac{1}{t_{end}} \sum_{\tau=0}^{t_{end}-1} r_{i,j}(\tau), \text{and}$$

$$\overline{h}_{i,j} \triangleq \frac{1}{t_{end}} \sum_{\tau=0}^{t_{end}-1} h_{i,j}(\tau).$$

The problem of minimizing the cost can be formulated as follows:

$$\min_{\mathbf{z}(t),\, t=0,1,2,\ldots,t_{end}-1} \overline{g} \tag{15}$$

$$\text{s.t.,} \quad a_j(t) = \sum_{i \in \mathcal{D}_j} r_{i,j}(t), \forall j = 1, 2, \ldots, J, \tag{16}$$

$$\mathbf{L}_i^{sup}(t) + \max[\mathbf{D}_i(t) \cdot \mathbf{b}_i(t)] \leq L_i^{\max}, \tag{17}$$

$$\sum_{j=1}^{J} h_{i,j}(t)d_j \leq \sum_{k=1}^{K_i(t)} u_{i,k}(t)s_k \leq \sum_{k=1}^{K_i(t)} s_k, \tag{18}$$

$$\overline{r}_{i,j} \leq \overline{h}_{i,j}, \forall i \in \mathcal{D}_j, j = 1, 2, \ldots, J, \tag{19}$$

where (17) is the server inlet temperature constraint, and $\sum_{k=1}^{K_i(t)} s_k$ in (18) is the maximum amount of work that can be processed in data center i during time t and specifies that the

scheduling decisions $h_{i,j}(t)$, for $j = 1, 2, \ldots, J$, are bounded in a convex polyhedron.

To solve the optimization problem (15)-(18), we need offline information (e.g., future job arrivals, supply temperature) which is unavailable in practice. Moreover, even if the complete information is perfectly known in advance, it requires dynamic programming to obtain the optimal solution and hence the curse of dimensionality occurs when the problem scale increases. Thus, we develop an efficient online algorithm that makes scheduling decisions based on the currently available information only.

4.2 Online Algorithm

Based on the recently developed Lyapunov optimization technique [16], this section presents an online algorithm "GreFar", whose performance is provably "good" compared to that of the optimal offline policy with T-step look-ahead information. The intuition of GreFar is to trade the delay for energy-fairness cost saving by using the queue length as a guidance for making scheduling decisions: jobs are processed only when the queue length becomes sufficiently large and/or electricity prices are sufficiently low.

Before presenting the algorithm, we need to introduce "queue dynamics", which specifies the queue length changes governed by the scheduling decisions (and job arrivals). The queue dynamics is instrumental for the scheduler to make online decisions. We express the queue dynamics governed the action $\mathbf{z}(t) = \{r_{i,j}(t), h_{i,j}(t),\ u_{i,k}(t),\ i = 1, 2, \ldots, N,\ j = 1, 2, \ldots, J,\ k = 1, 2, \ldots, K_i(t)\}$ as below:

$$q_{i,j}(t+1) = \max\big[q_{i,j}(t) - h_{i,j}(t), 0\big] + r_{i,j}(t), \qquad (20)$$

where $q_{i,j}(t)$ is the queue length for type-j jobs in data center i during time t.

We describe GreFar in Algorithm 1, which is purely online and requires only the current data center state and queue lengths as the input. Solving (21) subject to the constraints (12), (17), (18) is a convex optimization problem, to which efficient numerical algorithms (e.g., interior point method) exist [23]. In particular, if the fairness is not taken into account (i.e., $\beta = 0$ in (13)), solving (21) becomes a standard linear programming problem. Moreover, it can be easily observed that the rule of dispatching jobs from the central scheduler to data centers is "Join the Shortest Queue", i.e., all incoming type-j jobs are scheduled to the data center that has the shortest queue length of type-j jobs.

The parameter $V \geq 0$ is a control variable which we refer to as cost-delay parameter, and it can be tuned to different values to trade the queueing delay for the energy-fairness cost. For simplicity, let us consider $\beta = 0$ and explain the role of V in making scheduling decisions. By substituting (13) into (21), we see that the scheduler chooses $h_{i,j}(t) > 0$ only when the electricity price $\varphi_i(t)$ is sufficiently low such that $V \cdot (W\varphi_i(t)d_j)$ is smaller than the current queue length $q_{i,j}(t)$, where W is a positive constant that only depends on the server speeds and power consumptions. When V is larger, the scheduler will wait longer until the electricity price is sufficiently low relative to the queue length before scheduling the jobs for processing (i.e., opportunistically take the advantage of low electricity prices). As a result, the energy cost reduces at the expense of the increased delay. On the other hand, when V is smaller, the scheduler will schedule jobs even though the electricity price is not sufficiently low, resulting in an increased energy cost but reduced delay. When the fairness is taken into account in scheduling (i.e., $\beta > 0$), the role of V is similar and can achieve a tradeoff between the energy-fairness cost and delay.

Algorithm 1 GreFar Scheduling Algorithm

1: At the beginning of every time slot t, observe the data center state $\mathbf{x}(t)$ and the vector of current queue states $\Theta(t)$

2: Choose the control action $\mathbf{z}(t)$ subject to (12)(16)(17)(18) to minimize

$$V \cdot g(t) + \sum_{j=1}^{J} \sum_{i \in \mathcal{D}_j} q_{i,j}(t) \cdot \big[r_{i,j}(t) - h_{i,j}(t)\big], \qquad (21)$$

where the cost $g(t)$ is defined in (13).

3: Update $q_{i,j}(t)$ according to (20)

5 Algorithm Analysis

This section shows that the proposed online algorithm is provably-efficient against an optimal algorithm with T-step of look-ahead information. This section first describes the T-step look-ahead policy, and then analyzes the performance of GreFar against it. More specifically, we show that, given a cost-delay parameter V, our algorithm is $O(1/V)$-optimal with respect to average cost against the optimal T-step lookahead policy, while the queue length is bounded by $O(V)$.

5.1 T-Step Lookahead Policy

Here, we present the T-step lookahead policy, which has full knowledge of the data center states and job arrivals in the next (up to) T time steps. If T is sufficiently large (e.g., in the extreme case $T = t_{end}$), the T-step lookahead policy also "approximately" (or exactly if $T = t_{end}$) minimizes the average cost in (15).

We divide the time horizon of t_{end} time steps into $R \in \mathbb{Z}^+$ frames, each of which contains T time steps such that $t_{end} = RT$. In the T-step lookahead algorithm, the scheduler has future information (i.e., data center states and job arrivals) up to the next T time steps and minimizes the cost subject to certain constraints. Specifically, the cost minimization problem over the rth frame, for $r = 0, 1, \ldots, R-1$, can be formulated as

$$\min_{\mathbf{z}(t), t=rT, rT+1, \ldots, rT+T-1} \frac{1}{T} \sum_{t=rT}^{t=rT+T-1} g(t) \qquad (22)$$

$$\text{s.t.,} \quad \text{constraints } (16), (17), (18), \qquad (23)$$

$$\sum_{t=rT}^{t=rT+T-1} \big(r_{i,j}(t) - h_{i,j}(t)\big) \leq 0, \forall i \in \mathcal{D}_j, \forall j. \qquad (24)$$

In the problem (22)-(24), we denote the infimum of $\frac{1}{T}\sum_{t=rT}^{t=rT+T-1} g(t)$ by G_r^*, which is achievable over the rth frame considering all the actions including those that are chosen with

the perfect knowledge of data center states and job arrivals over the entire frame. Thus, the minimum cost over R frames achieved by the optimal T-step lookahead policy is

$$\frac{1}{R}\sum_{r=0}^{R-1} G_r^*. \tag{25}$$

In essence, the problem (22)-(24) encapsulates a family of offline algorithms parameterized by the lookahead information window size T. We shall show that our online algorithm can achieve a cost close to the value of (25).

5.2 Online Algorithm Analysis

This section presents the performance analysis of our proposed online algorithm compared with the optimal T-step lookahead policy.

Before showing the main theorem, we first present the slackness conditions, which bound the relationship between the resource demand and availability. These conditions are prerequisites of Theorem 1.[5]

Slackness condition: There exists a value $\delta > 0$ and a sequence of scheduling decisions $r_{i,j}(t)$ and $h_{i,j}(t)$ such that, for data center states $\mathbf{x}(t)$, $t = 0, 1, \ldots t_{end} - 1$, the following condition is satisfied:

$$r_{i,j}(t) \leq h_{i,j}(t) - \delta, \ \forall i \in \mathcal{D}_j, \forall j. \tag{26}$$

The condition (26) ensures that the scheduler can successfully schedule all arrived jobs with δ slackness. In practice, these three conditions are mild and can be easily satisfied. In particular, the computing resource in a data center is provisioned for the peak load, and thus the available computing resource is (almost) always sufficient for processing workloads. In the worst case where the data center is overloaded, admission control techniques can be applied to complement our scheme.

Theorem 1 shows the cost bound and the queue length bound of GreFar algorithm.

Theorem 1. *Suppose that the boundedness conditions (6), (11), (12) and the slackness condition (26) are satisfied for some $\delta > 0$, that the data center states $\mathbf{x}(t)$ and job arrivals $a_j(t)$ are arbitrarily random, for $t = 0, 1, \ldots t_{end} - 1$ and $j = 1, 2, \ldots J$, and that all the queue lengths are initially zero. Then, the following statements hold:*

a. *All queue lengths are bounded. For any time step $t = 0, 1, \ldots t_{end} - 1$, we have*

$$q_{i,j}(t) \leq \frac{VC_3}{\delta}, \tag{27}$$

where $V \geq 0$ and C_3 is a finite number defined in (47) in the appendix.

b. *For any $T \in \mathbb{Z}^+$ and $R \in \mathbb{Z}^+$ such that $t_{end} = RT$, the energy-fairness cost achieved by GreFar satisfies*

$$\overline{g}^* \leq \frac{1}{R}\sum_{r=0}^{R-1} G_r^* + \frac{B + D(T-1)}{V}, \tag{28}$$

where $\overline{g}^*$ is the cost achieved by GreFar for the problem (15)-(18), B and D are (finite) constants defined in the appendix and G_r^* is the minimum cost in the rth frame achieved by the T-step lookahead policy.

Proof. The proof is provided in the appendix. □

Theorem 1 shows that, given a cost-delay parameter V, our algorithm is $O(1/V)$-optimal with respect to the average energy-fairness cost against the optimal T-step lookahead policy, while the queue length is bounded by $O(V)$. More specifically, the inequality (27) bounds the queue length: the queue length (which is closely related to the average delay) is bounded by $O(V)$ where V is the cost-delay parameter in Algorithm 1. The queue length bound is tighter when V is smaller. The inequality (28) shows that the average energy-fairness cost is bounded within an additional $O(1/V)$ cost of that achieved by the optimal T-step lookahead policy. The cost bound is closer to the minimum cost when V is larger.

Theorem 1 also shows that, by appropriately tuning the cost-delay parameter V, we can achieve a desired tradeoff between the cost and queue lengths. In particular, by increasing the value of $V \geq 0$, the (energy-fairness) cost becomes closer to that achieved by the optimal T-step lookahead policy, whereas the upper bounds on the queue lengths become larger.

5. If we only assume that the problem (15)-(18) is *strongly* feasible without slackness conditions, the performance analysis of energy-fairness cost remains similar while the upper bound on the queue length may grow as the time passes [15].

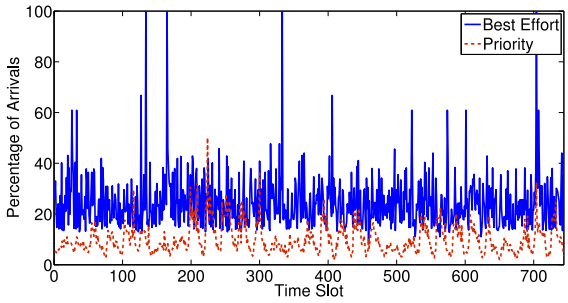

Fig. 2. Job arrival traces.

6 Simulation Setup

We build a discrete-time-based simulator and drive the simulation to evaluate the performance of GreFar scheduling algorithm. In order to highlight the improvement in performance by incorporating temperature constraints, we compare the GreFar performance with the one obtained by the algorithm defined in [3]. We refer to this algorithm as T-unaware.

The simulation setup is described in the following sections and characterized by:

- Job Arrival;
- Data center features;
- Electricity Cost;
- Thermal parameters.

6.1 Job Arrival

We consider two different types of jobs: best-effort and priority. Job arrival traces are depicted in Fig. 2. Priority jobs must be served in the same time slot as they arrive. Clearly, when the rate of priority arrivals grows, the number of servers required to process them increases too, resulting in an

TABLE 1
Data Center Description

DC	CPU Speed	Full Power	Idle Power
1	295	27250	13625
2	310	28100	14050
3	305	28350	14175
4	259	25450	12725

increase in the server inlet temperature due to processing and decreasing the amount of resources available to serve best-effort jobs.

To evaluate the behavior of GreFar under different workloads, we define two scaling factors, α (workload scaling factor for best-effort jobs) and α_P (workload scaling factor for priority jobs). We vary the workload of best-effort and priority jobs by multiplying the traces reported in Fig. 2 for α and α_P respectively. In the simulation, 744 time slot are considered. Every time slot takes 1 hour, and the entire simulation is one month long.

Job types are modeled as in [25] (nine job types with different sizes).

6.2 Data Center Features

We consider four data centers, each one consisting of up to 200 blade servers arranged in four racks. Four different server types are considered, with the following configurations:

- normalized processing speed (CPU cycles): 1, 1.5, 2, 2.8 respectively;
- full power consumption: 100, 130, 145, 170 W, respectively.
- idle power consumption: 50 percent of the full power consumption;

Table 1 reports the configuration of each data center. During each time slot, some servers could be not available for processing jobs (e.g., due to failures), and hence the values reported in column 2 of Table 1 represent an upper bound of the processing capabilities of every data center.

6.3 Electricity Cost

The electricity price used in the simulation is obtained from [26]. In Fig. 3, the trend of electricity price from the ninth of December 2012 to the ninth of January 2013 is shown.

6.4 Thermal Parameters

The thermal parameters are obtained from the 1D model described in [13] with $K = 4$ and $L^{max} = 25°\mathrm{C}$ (equal for all data centers). For the GreFar algorithm, we fixed

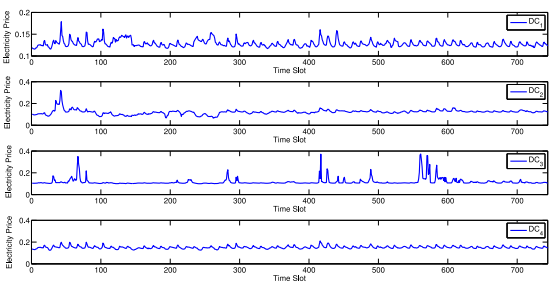

Fig. 3. Electricity price trend.

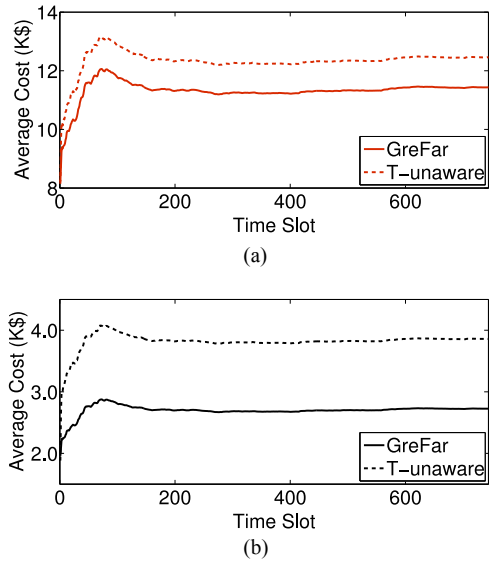

Fig. 4. Average cost: a) total; b) cooling.

$L^{sup} = 20°\mathrm{C}$, and, as explained in previous section, GreFar is able to maintain the maximum temperature constraint. In the case of T-unaware algorithm, it is not possible to fix a supply temperature value assuring that L^{max} constraint is respected, since no mechanisms to control L^{max} are present in the scheduling algorithm. To have a fair comparison among the two algorithms we compute, after the scheduling decision of the T-unaware algorithm at each time slot, the L^{sup} value that allows to satisfy the temperature constraint on L^{max}. In other words, we are simulating a scheduling-aware air conditioning system, able to modify its supply temperature as a consequence of jobs assignments.

The power consumption of the computer room air conditioner (CRAC), that depends of the value of L^{sup}, is calculated according to the model described in [27].

7 Performance Evaluation

In this section we evaluate the performance of GreFar. We first provide a performance comparison among GreFar and T-unaware in terms of cost and delay; after that we focus on the various parameters that impact the effectiveness of our solution.

7.1 Cost Evaluation

The most interesting performance index to evaluate is the total cost, as defined in Eq. (13); it takes into account processing, cooling and fairness cost. In this first evaluation we fix $V = 1$ while other parameters such as L^{max}, L^{sup} and α are fixed as explained in Section 6.

In Fig. 4a the total cost at each time slot for GreFar and T-unaware is reported: Fig. 4a highlights that the GreFar algorithm allows to reduce the total cost of about 10 percent with respect to T-unaware. This cost saving is mainly due to the reduction of the cooling cost, as shown in Fig. 4b, where only cooling costs for both algorithms are reported. This first result highlights the ability of GreFar to reduce the overall cost acting on the cooling system. In particular, the cost reduction of GreFar is due

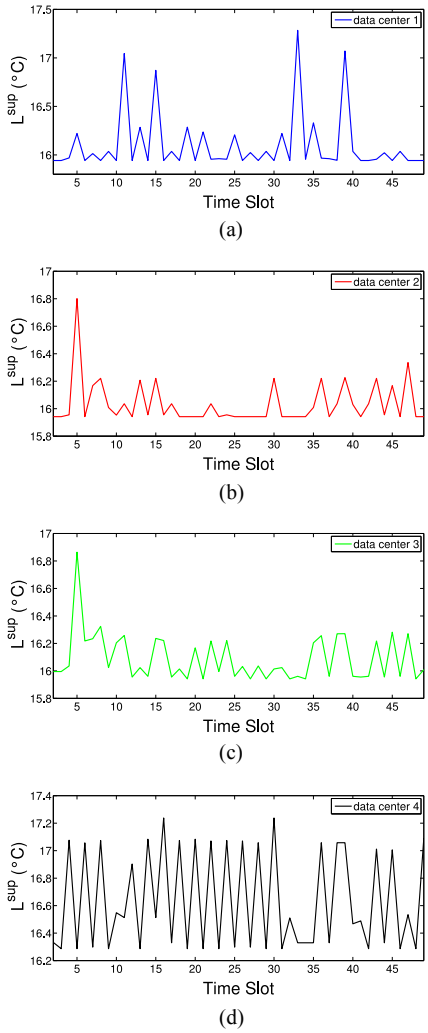

Fig. 5. L^{sup} value when T-unaware algorithm is performed in the four data centers.

to the introduction of the cooling cost in the objective function. In other words GreFar uses a higher cost delay parameter with respect to T-unaware, referred to as equivalent cost-delay parameter V_{eq}:

$$V_{eq} = V\left(1 + \frac{1}{CoP(L^{sup})}\right). \quad (29)$$

To better understand how the cooling cost can be reduced we focus on the supply temperature L^{sup}. As already explained in Section 6.4, L^{sup} is fixed for GreFar, while it must be changed for T-unaware so that to satisfy the L^{max} constraint. The L^{sup} value required to keep the L^{max} under the limit of $25°\mathrm{C}$ when the T-unaware algorithm is used, is shown in Figs. 5a, 5b, 5c, 5d for the four data centers over a time period of 48 hours. Considering that for GreFar the L^{sup} value is equal to $20°\mathrm{C}$, T-unaware requires an L^{sup} value that is on average $3.5°\mathrm{C}$ lower than GreFar one: for this reason the cooling cost in the case of T-unaware is higher than the cooling cost incurred by GreFar.

7.2 The Cost-Delay Parameter

Previous results highlighted the ability of GreFar to reduce cooling cost and consequently the total cost of the system.

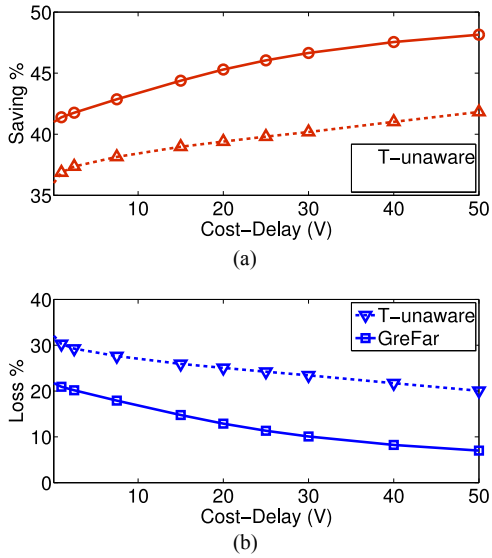

Fig. 6. (a) Saving and (b) Loss functions for different values of V.

Anyway the performance of GreFar depends on many parameters: in this section we focus on the cost-delay parameter V. To perform a detailed analysis we first introduce two cost-related parameters:

- the *Delay-Greedy* parameter, denoted by $C(0)$: it represents the total cost in the case of T-unaware algorithm with $V = 0$, i.e., the cost obtained when only delay minimization is performed;
- the *Cost-Greedy* parameter, denoted by $C(\infty)$: it represents the total cost in the case of GreFar algorithm with $V = \infty$, i.e., the cost obtained when only cost minimization is performed.

$C(0)$ and $C(\infty)$ represent the upper bound and the lower bound costs for a generic scheduling algorithm using objective function 15. In other words it is possible to characterize the performance of a specific algorithm, i.e., GreFar or T-unaware with a fixed V value, comparing its cost with $C(0)$ and $C(\infty)$. Starting from previous consideration two performance indexes can be defined: (i) *Saving* and (ii) *Loss*, expressed as:

$$Saving = \frac{C(0) - C_x(V)}{C(0)} \quad x = GreFar, T-unaware, \quad (30)$$

$$Loss = \frac{|C(\infty) - C_x(V)|}{C(\infty)} \quad x = GreFar, T-unaware, \quad (31)$$

where $C_x(V)$ is the cost of the algorithm (GreFar or T-unaware) with V value. The *Saving* value represents the ability of an algorithm to reduce costs with respect to the most expensive case, while the *Loss* value represents the cost inefficiency with respect to the less expensive case.

Saving and *Loss* values are reported in Figs. 6a and 6b as a function of V, respectively. As expected the results highlight that GreFar show better performance with respect to T-unaware. In more detail, the *Saving* value of GreFar is always greater than 40 percent; moreover the cost saving of GreFar with respect to T-unaware varies from 6 to 16 percent. A similar result is obtained for the

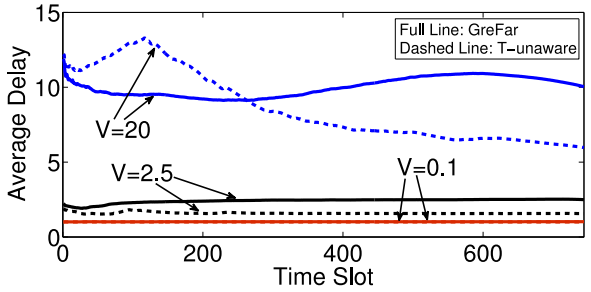

Fig. 7. Average delay versus time.

Loss parameter: the Loss value of GreFar is not greater than 20 percent and for high values of V it reaches the 6 percent.

The main result of the previous analysis is that increasing V leads to obtain better performance in terms of cost saving. This aspect is counterbalanced by the impact of V on other performance indexes: in particular the impact on jobs delay must be evaluated. In Fig. 7 we report the average delay, expressed in number of time slots, of best-effort jobs for different values of V: we only report results for data center 4 since it experienced the higher delays. The first result is that for small values of V, the average delay is very low, while it increases when V increases. Moreover the T-unaware algorithm experiences lower delay with respect to GreFar, and the differences among the two algorithms increase for high value of V: this is the side effect of GreFar that introduces additional delay to save cost in jobs execution.

The conclusion of this analysis is that V setting is a very important aspect of GreFar implementation and it strongly depends on the performance policies (such as delay) implemented into the data center.

7.3 Temperature and Traffic Parameters

In the previous analysis the temperature and traffic parameters have been considered fixed; in this section we evaluate their impact on the performance of GreFar. To better evaluate the results we consider the whole system, composed of four data centers and hundreds of servers, as a queue system with a single server; the server capacity (S) in this model is equal to the sum of all blade servers capacities in the simulation setup.

The temperature parameters L^{max} and L^{sup}, and the traffic parameter α_p, directly affect S: L^{max} and L^{sup} influence the scheduling decision, while α_p represents the priority jobs, that must be served with no delay and so cause a reduction of the server capacity available for best effort jobs. In other

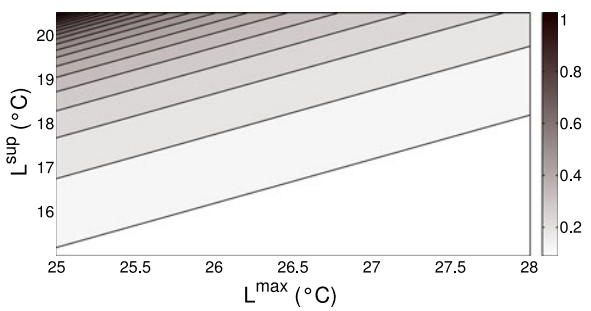

Fig. 8. Effect of temperature constraint on γ in case of GreFar.

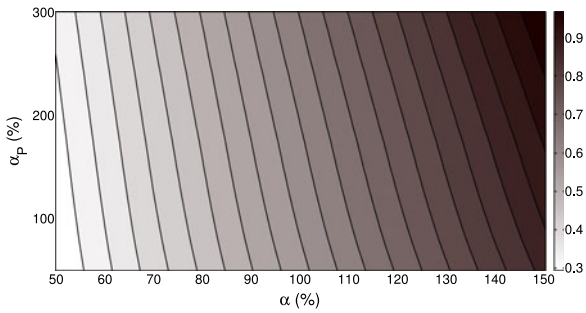

Fig. 9. Effect of workload on γ in case of GreFar.

words S is a function of previous parameters. To evaluate the effects of the previous parameters on the system performance we analyze the mean server utilization, denoted with γ, which indicates the mean fraction of time that the server is busy. γ is computed as the ratio of the average best-effort jobs to the actual server capacity, as reported in

$$\gamma = \frac{\overline{A}_{BE}}{S(L^{max}, L^{sup}, \overline{A}_P)}, \tag{32}$$

where $\overline{A}_{BE}$ and $\overline{A}_P$ are the average requests of best-effort and priority jobs, respectively: $\overline{A}_{BE}$ and $\overline{A}_P$ are the average values of the two traces shown in Fig. 2 multiplied for α and α_P.

In Fig. 8, γ as a function of L^{max} and L^{sup} is reported; the traffic parameter are fixed so that $\alpha = \alpha_P = 1$. In the considered temperatures region, γ is always lower than 1, which represents the bound between stability and instability of the queue system. As expected, increasing L^{max} corresponds to increase servers availability, since processing capabilities can be used without exceeding the temperature constraint, while increasing L^{sup} corresponds to reduce system ability to perform jobs execution without reaching the temperature constraint and so leads to a reduced processing capability.

In Fig. 9 γ as a function of α and α_P is reported; the temperature parameters are fixed so that $L^{max} = 25°\text{C}$ and $L^{sup} = 20°\text{C}$. The results show that both α and α_P has a direct impact on server availability: the jobs arrival increase leads to server availability reduction.

7.4 The Heat Transfer Matrix

As already stressed throughout the paper, GreFar algorithm uses the Heat Transfer Matrix D to predict the temperature increase due to jobs assignment, and so it represents an important input for GreFar algorithm. From an implementation point of view, D can be estimated using direct measurements and hence, it could be affected by errors. In order to evaluate the impact of an error in the estimation of D, we assume a modified heat transfer matrix, referred to as D^*, obtained increasing all the elements $d_{i,j}$ by an estimation error (e.g., 5 percent error means that each $d_{i,j}$ is increased by 5 percent): we focus our analysis on over-estimation of D but similar results have been obtained in the case of underestimation of D. The performance evaluation is realized executing GreFar on D and on D^*, and evaluating the cost variation parameter (Δ_C): it represents the normalized difference between the average costs of GreFar executed on D^* and on D. In Fig. 10 the cost variation Δ_C as a function of the estimation error is reported.

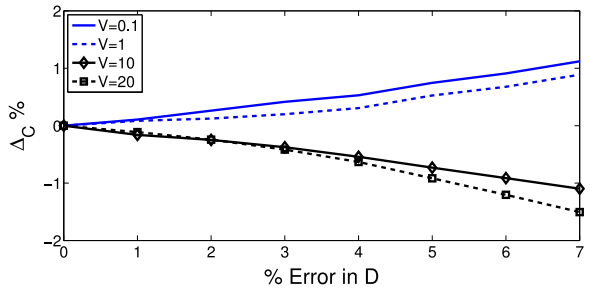

Fig. 10. Effect of error in the estimation of D on the average cost.

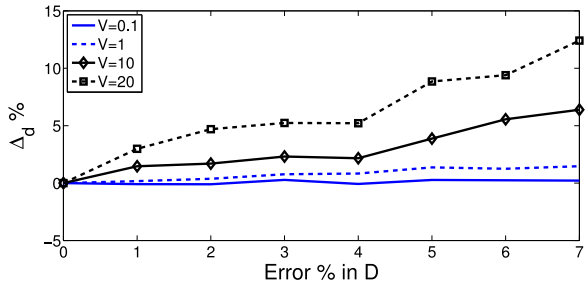

Fig. 11. Effect of error in the estimation of D on the average delay.

Fig. 10 shows not expected results: for low values of V, the increase of the estimation error leads to a higher Δ_C, and so to a higher cost, with respect to a correct estimation of D; on the contrary when V is high, Δ_C decreases for higher values of the estimation error.

To better understand the previous results, the effects of D over-estimation must be considered. From one side D^* has an impact on the scheduling algorithm execution: the increase of the server inlet temperature due to cross-dissipation among servers is over-estimated and so the assignment of jobs is conservative, resulting in a lower cost with respect to a correct (lower) D value case. On the other side D^* influences the setting of the cooling system: the power dissipated by the servers is over-estimated, and so the L^{sup} value able to maintain the maximum temperature constraint must be smaller than L^{sup} value in case of a correct D estimation; in this way the cooling power consumption increases. The first effect is predominant for high value of V, since in this case the scheduling algorithm is mainly based on cost reduction and so on D; the second effect is predominant for low values of V, since in this case the scheduling algorithm is mainly based on delay minimization and so D is not considered. These considerations are confirmed by the evaluation reported in Fig. 11, showing the delay increase (Δ_D) due to D estimation error. As expected, for low values of V the delay variation is almost negligible, while it is significant for high values of V; in the latter case the scheduling decision is influenced by D^*.

8 Conclusion

This paper presents a provably-efficient online algorithm, GreFar, for scheduling batch jobs among multiple geographically distributed data centers. For arbitrarily random job arrivals, server availability and electricity prices, GreFar minimizes the energy-fairness cost while providing queueing delay guarantees. It opportunistically schedules jobs when electricity prices are sufficiently low and to places where the energy cost per unit work is low. Given the cost-delay parameter V, GreFar achieves a cost within $O(1/V)$ of the optimal T-step lookahead algorithm, while bounding the queue length by $O(V)$.

The performance evaluation of GreFar proves its ability of reducing the energy-fairness cost. In particular, comparing the performance of GreFar with ones of T-unaware, a similar algorithm with no servers temperature awareness, highlights that GreFar reduces the power consumption of the cooling system, maintaining the maximum server inlet temperature constraint. The impact of the cost reduction on the delay of jobs is limited and can be appropriately tuned acting on the cost-delay parameter V.

Appendix

Proof. Here, we provide the proof of Theorem 1. First, we define the vector of all queue states during time t as

$$\Theta(t) \triangleq \{q_{i,j},\ \forall i \in \mathcal{D}_j, \forall j = 1, 2, \ldots, J\}, \tag{33}$$

where $q_{i,j}(t)$ is the queue length for type-j jobs in data center i during time t. As a scalar measure of all the queue lengths, we define the quadratic Lyapunov function as

$$L(\Theta(t)) \triangleq \frac{1}{2}\sum_{j=1}^{J}\sum_{i\in\mathcal{D}_j} q_{i,j}^2(t). \tag{34}$$

Let $\triangle_T(t)$ be the T-step Lyapunov drift yielded by some control policies over the interval $t, t+1, \ldots, t+T-1$:

$$\triangle_T(t) \triangleq L(\Theta(t+T)) - L(\Theta(t)). \tag{35}$$

Similarly, the 1-step drift is

$$\triangle_1(t) \triangleq L(\Theta(t+1)) - L(\Theta(t)). \tag{36}$$

Then, it can be shown that the 1-step drift satisfies

$$\triangle_1(t) \leq B + \sum_{j=1}^{J}\sum_{i\in\mathcal{D}_j} q_{i,j}(t)\cdot[r_{i,j}(t) - h_{i,j}(t)], \tag{37}$$

where B is a constant satisfying, for all $t = 0, 1, \ldots, t_{end}$,

$$B \geq \frac{1}{2}\left[a_j(t) + \sum_{i\in\mathcal{D}_j} r_{i,j}(t)\right]^2 + \frac{1}{2}\left[r_{i,j}(t) + h_{i,j}(t)\right]^2, \tag{38}$$

where is finite due to the boundedness conditions (6), (11), (12).

Part (a):. Based on (37), we can easily show that

$$\begin{aligned}\triangle_1(t) + V\cdot g(t) \leq\ & B + V\cdot g(t) \\ & + \sum_{j=1}^{J}\sum_{i\in\mathcal{D}_j} q_{i,j}(t)\cdot\left[r_{i,j}(t) - h_{i,j}(t)\right].\end{aligned} \tag{39}$$

Thus, GreFar actually minimizes the upper bound on the 1-step Lyapunov drift plus a weighted cost shown on the right hand side of (39).

Let us choose a control action $\mathbf{z}'(t)$ satisfying the slackness condition (26). The corresponding 1-step Lyapunov drift plus a weighted cost achieved satisfies

$$\triangle_1(t) + V \cdot g(t) \leq B + V \cdot g(t) - \delta \sum_{j=1}^{J} \sum_{i \in \mathcal{D}_j} q_{i,j}(t). \quad (40)$$

Since GreFar minimizes the right hand side of (39), the 1-step Lyapunov drift plus a weighted cost achieved by GreFar must be less than or equal to that achieved by $\mathbf{z}'(t)$. In other words, the following inequality can be established:

$$\triangle_1^*(t) \leq B + V \cdot (g^{\max} - g^{\min}) - \delta \sum_{j=1}^{J} \sum_{i \in \mathcal{D}_j} q_{i,j}(t), \quad (41)$$

where $g^{\max}$ and $g^{\min}$ are the maximum and minimum 1-step costs,[6] respectively, and $\triangle_1^*(t)$ is the 1-step Lyapunov drift achieved by GreFar. Now, we define

$$P \triangleq B + V \cdot (g^{\max} - g^{\min}). \quad (42)$$

Thus, if the sum of the queue lengths, $\sum_{j=1}^{J} \sum_{i \in \mathcal{D}_j} q_{i,j}(t)$, is greater than orequal to P/δ, the 1-step Lyapunov drift in (41) is non-positive. Moreover, we can show that the maximum value of the Lyapunov function is $[P/(\sqrt{2}\delta)]^2$ under the constraint that the sum of all the queue lengths is less than or equal to P/δ. Thus, if the Lyapunov function is greater than $[P/(\sqrt{2}\delta)]^2$, the sum of all the queue lengths will be greater than P/δ and the Lyapunov function in the next step will not increase, since the 1-step Lyapunov drift is negative. Nevertheless, if the sum of all the queue lengths is less than or equal to P/δ during time t, we have

$$\begin{aligned} L(\mathbf{\Theta}(t+1)) &\leq \frac{1}{2} \sum_{j=1}^{J} \sum_{i \in \mathcal{D}_j} \left[q_{i,j}(t) + q_{i,j}^{diff}(t) \right]^2 \\ &\leq L(\mathbf{\Theta}(t)) + D + \frac{q^{\max} P}{\delta} \\ &\leq \left(\frac{P}{\sqrt{2}\delta} \right)^2 + D + \frac{q^{\max} P}{\delta}, \end{aligned} \quad (43)$$

where $q_{i,j}^{diff}(t)$ represent the absolute values of changes in the respective queue lengths, with maximum values being $q_{i,j}^{diff} = \max[r_{i,j}^{\max}, h_{i,j}^{\max}]$, respectively,

$$q^{\max} = \max_{i \in \mathcal{D}_j, j=1,2,\cdots,J} \{ q_{i,j}^{diff} \},$$

and D is a constant satisfying, for all $t = 0, 1, \ldots, t_{end} - 1$,

$$D \geq \frac{1}{2} \sum_{j=1}^{J} \sum_{i \in \mathcal{D}_j} q_{i,j}^{diff} \max \left[r_{i,j}(t), h_{i,j}(t) \right], \quad (44)$$

6. Both $g^{\max}$ and $g^{\min}$ are finite due to boundedness conditions (11), (12).

which is finite due to the boundedness conditions. Clearly, $L(\mathbf{\Theta}(0))$ satisfies (43), as all the queue lengths are initially zero. Then, by mathematical induction, we can show that, for any $t = 0, 1, \ldots, t_{end} - 1$,

$$L(\mathbf{\Theta}(t)) \leq \left(\frac{P}{\sqrt{2}\delta} \right)^2 + D + \frac{q^{\max} P}{\delta}, \quad (45)$$

following which we see that all the queue lengths are bounded by

$$\begin{aligned} q_{i,j}(t) \leq & \sqrt{\left(\frac{P}{\delta} \right)^2 + 2D + \frac{2q^{\max} P}{\delta}} \\ = & \frac{V \sqrt{\frac{P^2}{V^2} + \frac{2D\delta^2}{V^2} + \frac{2q^{\max}\delta P}{V^2}}}{\delta} = \frac{VC_3}{\delta}, \end{aligned} \quad (46)$$

where

$$C_3 = \sqrt{D_1 + D_2 + D_3}, \quad (47)$$

in which

$$D_1 \triangleq \left[\frac{B}{V} + g^{\max} - g^{\min} \right]^2, \quad (48)$$

$$D_2 \triangleq \frac{2D\delta^2}{V^2}, \quad (49)$$

$$D_3 \triangleq \frac{2q^{\max}\delta}{V} \sqrt{D_1}. \quad (50)$$

This proves part (a) of Theorem 1.

Part (b): Based on (39), we can show that, for $r = 0, 1, \ldots, R-1$, the T-step drift plus weighted cost satisfies

$$\begin{aligned} &\triangle_T^*(rT) + V \sum_{t=rT}^{rT+T-1} g^*(t) \\ &\leq BT + V \sum_{t=rT}^{rT+T-1} g(t) \\ &\quad + \sum_{j=1}^{J} \sum_{i \in \mathcal{D}_j} \sum_{t=rT}^{rT+T-1} (t - rT) q_{i,j}^{diff} [r_{i,j}(t) - h_{i,j}(t)] \\ &\quad + \sum_{j=1}^{J} \sum_{i \in \mathcal{D}_j} q_{i,j}(rT) \sum_{t=rT}^{rT+T-1} [r_{i,j}(t) - h_{i,j}(t)]. \end{aligned} \quad (51)$$

Then, after some simple mathematic manipulations based on (51), we can derive the following inequality:

$$\begin{aligned} &\triangle_T^*(rT) + V \sum_{t=rT}^{rT+T-1} g^*(t) \\ &\leq BT + V \sum_{t=rT}^{rT+T-1} g(t) + DT(T-1) \\ &\quad + \sum_{j=1}^{J} \sum_{i \in \mathcal{D}_j} q_{i,j}(rT) \sum_{t=rT}^{rT+T-1} [r_{i,j}(t) - h_{i,j}(t)], \end{aligned} \quad (52)$$

where D is a finite constant satisfying (44). In (52), the left hand side is the T-step Lyapunov drift plus weighted cost

achieved by GreFar, which explicitly minimizes the right hand side of (39). Note that the right hand side of (39) is smaller than or equal to that of (52). Thus, by considering the optimal T-step lookahead policy on the right hand side of (52), we obtain the following inequality

$$\begin{aligned}
&\triangle_T^*(rT) + V \sum_{t=rT}^{rT+T-1} g^*(t) \\
&\quad \leq BT + VTG_r^* + DT(T-1) \\
&\qquad + \sum_{j=1}^{J} \sum_{i \in \mathcal{D}_j} q_{i,j}(rT) \sum_{t=rT}^{rT+T-1} [r_{i,j}(t) - h_{i,j}(t)] \\
&\quad \leq BT + VTG_r^* + DT(T-1),
\end{aligned} \tag{53}$$

where the second inequality follows from the constraints in (23) and (24) satisfied by the optimal T-step lookahead policy. Therefore, by summing (53) over $r = 0, 1, \ldots, R-1$, and considering that all the queues are initially empty, it follows that

$$\begin{aligned}
V \sum_{t=0}^{RT-1} g^*(t) \leq & BTR + VT \sum_{r=0}^{R-1} G_r^* \\
& + RDT(T-1) - L(\Theta(RT-1)) \qquad (54) \\
\leq & BTR + VT \sum_{r=0}^{R-1} G_r^* + RDT(T-1).
\end{aligned}$$

Finally, by dividing both sides in (54) by VTR, we have

$$\overline{g}^* = \frac{1}{TR} \sum_{t=0}^{RT-1} g^*(t) \leq \frac{1}{R} \sum_{r=0}^{R-1} G_r^* + \frac{B + D(T-1)}{V}, \tag{55}$$

which shows that the online algorithm can achieve a cost within $O(1/V)$ to the minimum cost achieved by the optimal T-step lookahead policy. This proves part (b) of Theorem 1. □

REFERENCES

[1] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs, "Cutting the Electric Bill for Internet-Scale Systems," *Proc. ACM SIGCOMM*, 2009.

[2] J. Chen, R. Tan, Y. Wang, G. Xing, X. Wang, X.-D. Wang, B. Punch, and D. Colbry, "A High-Fidelity Temperature Distribution Forecasting System for Data Centers," *Proc. IEEE 33rd Real-Time Systems Symp. (RTSS)*, 2012.

[3] S. Ren, Y. He, and F. Xu, "Provably-Efficient Job Scheduling for Energy and Fairness in Geographically Distributed Data Centers," *Proc. IEEE 32nd Int'l Conf. Distributed Computing Systems (ICDCS)*, pp. 22-31, 2012.

[4] N. Buchbinder, N. Jain, and I. Menache, "Online Job Migration for Reducing the Electricity Bill in the Cloud," *Proc. 10th Int'l IFIP TC 6 Conf. Networking (IFIP Networking)*, 2011.

[5] M. Lin, A. Wierman, L.L.H. Andrew, and E. Thereska, "Dynamic Right-Sizing for Power-Proportional Data Centers," *Proc. IEEE INFOCOM*, 2011.

[6] B. Guenter, N. Jain, and C. Williams, "Managing Cost, Performance and Reliability Tradeoffs for Energy-Aware Server Provisioning," *Proc. IEEE INFOCOM*, 2011.

[7] Z. Liu, M. Lin, A. Wierman, S. Low, and L.H. Andrew, "Greening Geographical Load Balancing," *Proc. ACM SIGMETRICS Joint Int'l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2011.

[8] L. Rao, X. Liu, L. Xie, and W. Liu, "Reducing Electricity Cost: Optimization of Distributed Internet Data Centers in a Multi-Electricity-Market Environment," *Proc. IEEE INFOCOM*, 2010.

[9] D. Xu and X. Liu, "Geographic Trough Filling for Internet Datacenters," http://arxiv.org/abs/1108.5494, 2014.

[10] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M.J. Neely, "Data Centers Power Reduction: A Two Time Scale Approach for Delay Tolerant Workloads," *Proc. IEEE INFOCOM*, 2012.

[11] Y. Guo, Z. Ding, Y. Fang, and D. Wu, "Cutting Down Electricity Cost in Internet Data Centers by Using Energy Storage," *Proc. IEEE GLOBECOM*, 2011.

[12] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S.K.S. Gupta, and S. Rungta, "Spatio-Temporal Thermal-Aware Job Scheduling to Minimize Energy Consumption in Virtualized Heterogeneous Data Centers," *Elsevier Computer Networks*, vol. 53, no. 17, Dec. 2009.

[13] K. Mukherjee, S. Khuller, and A. Deshpande, "Saving on Cooling: The Thermal Scheduling Problem," *Proc. 12th ACM SIGMETRICS/Performance Joint Int'l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2012.

[14] T. Mukherjee, Q. Tang, C. Ziesman, S.K.S Gupta, and P. Cayton, "Software Architecture for Dynamic Thermal Management in Datacenters," *Proc. Second Int'l Conf. Comm. Systems Software and Middleware (COMSWARE '07)*, pp. 1-11, Jan. 2007.

[15] M.J. Neely, "Universal Scheduling for Networks with Arbitrary Traffic, Channels, and Mobility", technical report, 2010.

[16] L. Georgiadis, M.J. Neely, and L. Tassiulas, "Resource Allocation and Cross-Layer Control in Wireless Networks," *Foundations and Trends in Networking*, vol. 1, no. 1, 2006.

[17] A.H. Beitelmal and C.D. Patel, "Thermo-Fluids Provisioning of a High Performance High Density Data Center," *Distributed and Parallel Databases*, vol. 21, no. 2/3, pp. 227-238, June 2007.

[18] B. Hayes, "Cloud Computing," *Comm. ACM*, vol. 51, no. 7, pp. 9-11, July 2008.

[19] S. Li, T. Abdelzaher, and M. Yuan, "Tapa: Temperature Aware Power Allocation in Data Center with Map-Reduce," *Proc. Int'l Green Computing Conf. and Workshops (IGCC)*, 2011.

[20] J. Moore, J. Chase, P. Ranganathan, and R. Sharma, "Making Scheduling Cool: Temperature-Aware Workload Placement in Data Centers," *Proc. USENIX Ann. Conf. USENIX Ann. Technical Conf. (ATEC)*, 2005.

[21] J.W. Lee, M. Chiang, and R.A. Calderbank, "Utility-Optimal Random-Access Control," *IEEE Trans Wireless Comm.*, vol. 6, no. 7, pp. 2741-2751, July 2007.

[22] Federal Energy Regulatory Commission, http://www.ferc.gov, 2014.

[23] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge Univ. Press, 2004.

[24] D.P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1995.

[25] M. Zaharia, D. Borthakur, J.S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling," *Proc. Fifth European Conf. Computer Systems (EuroSys '10)*, pp. 265-278, 2010.

[26] Market Information System, http://nodal.ercot.com/docs/pd/mis/index.html, 2014.

[27] 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 Trans. Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1458-1472, Nov. 2008.

Marco Polverini received the Laurea degree in telecommunications engineering, in April 2010 from the University of Rome "Sapienza," Italy. Since November 2010, he has been working toward the PhD degree in information and communication engineering at the same Department. His current research interests are protocols and models for energy saving in IP networks.

Antonio Cianfrani received the "Laurea" degree in telecommunications engineering in 2004 and the PhD degree in information and communication engineering in 2008, both from the University of Rome "Sapienza." He is an assistant professor at the DIET Department of the University of Rome "Sapienza." His field of interests includes routing algorithms, network protocols, performance evaluation of software routers and optical networks. His current research interests are focused on green networks and future Internet architecture.

Shaolei Ren (M '13) received the BE, MPhil, and PhD degrees, all in electrical engineering, from Tsinghua University in July 2006, Hong Kong University of Science and Technology in August 2008, and the University of California, Los Angeles, in June 2012, respectively. Since August 2012, he has been with the School of Computing and Information Sciences, Florida International University, as an assistant professor. His research interests include cloud computing, data center resource management, and sustainable computing. He received the Best Paper Award from IEEE International Conference on Communications in 2009 and from International Workshop on Feedback Computing in 2013. He is a member of the IEEE.

Athanasios V. Vasilakos is currently a professor at the University of Western Macedonia, Greece. He has authored or coauthored more than 200 technical papers in major international journals and conferences. He is an author/coauthor of five books and 20 book chapters in the areas of communications. He has served as a general chair, a Technical Program Committee chair for many international conferences. He served or is serving as an editor or/and guest editor for many technical journals. He is the founding editor-in-chief of the *International Journal of Adaptive and Autonomous Communications Systems (IJAACS)* and the *International Journal of Arts and Technology (IJART)*. He is the general chair of the Council of Computing of the European Alliances for Innovation.