Received July 29, 2020, accepted August 16, 2020, date of publication September 1, 2020, date of current version September 16, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3020940

Proactive Data Center Management Using Predictive Approaches

RUBEN MILOCCO[1], PASCALE MINET[2], ÉRIC RENAULT[3], AND SELMA BOUMERDASSI[4], (Associate Member, IEEE)
[1]GCAyS, UNComahue, Neuquén 8300, Argentina
[2]INRIA, 75589 Paris, France
[3]LIGM, Univ. Gustave Eiffel, CNRS, ESIEE Paris, 93162 Noisy-le-Grand, France
[4]CNAM/CEDRIC, 75003 Paris, France

Corresponding author: Selma Boumerdassi (selma.boumerdassi@cnam.fr)

ABSTRACT Data Center (DC) management aims at promptly serving user requests while minimizing the energy consumed. This is achieved by turning off unnecessary servers to save energy and adapting the number of servers that are *on* to the time-varying and heterogeneous user requests. A great change in the number of servers *on* leads to a considerable management effort, also called control effort in the literature, which should be reduced as much as possible. Since feedback control can improve the performance of computing systems and networks, we propose to use it to achieve this dynamic capacity provisioning of the DC. In order to design this feedback control, first, we developed a dynamic model of the DC. The purpose of this paper is to design a feedback control strategy based on the DC model, able to optimize i) the Quality of Service, ii) the energy consumed and iii) the management effort. A simple Reactive open-loop Control which provides an amount of energy equal to the amount requested in the previous time interval is considered as a benchmark for comparison. Second, two feedback controls based on the balance equations of the DC are studied, namely i) Reactive Feedback Control providing an amount of energy equal to that provided by the reactive open-loop control but adding the accumulated demand that has not yet been served, and ii) Model Predictive Control optimizing a constrained cost that weights the management effort and the prediction error. Reactive Control, Reactive Feedback Control and Model Predictive Control are compared in terms of energy consumed, energy error and management effort. Quantitative results of the comparative performance evaluation are given, based on a data set collected from a real DC.

INDEX TERMS Data center management, energy efficiency, quality of service, dynamic capacity provisioning, reactive control, reactive feedback control, model predictive control.

I. INTRODUCTION

Data Center (DC) management must achieve two fundamental goals, which are conflicting. On the one hand, it must serve the time varying user requests in the shortest possible time and, on the other hand, it must minimize its energy consumption. The former constitutes an important parameter to quantify the Quality of Service (QoS). For example, having a high number of servers powered *on* that exceeds peak demand is an optimal strategy from the QoS point of view, but it consumes excessive energy since much of it is not being used. An efficient way to reduce power consumption in a DC is to shut down servers that are not being used and turn them *on* only when they are needed.

The associate editor coordinating the review of this manuscript and approving it for publication was Peng-Yong Kong.

Time-varying jobs arriving at the DC require that the number of servers that are *on* at any given time is periodically adjusted in order to process the job requests. This is called dynamic capacity provisioning. If the amount of power supplied matches the power required to serve requests in the current DC management period, these requests will be served. If the provisioning is less than the power required to process the requests, there will not be enough servers running in the DC to process all the incoming requests. Requests that cannot be served with the resources available in the current period have to wait for the next DC management period, when some servers can be added (i.e. reactive action). If the supply power is greater than the requested, the DC resources will be underutilized and a part of the available power will be wasted.

Considering a periodic DC management (i.e. one that is done at constant time intervals), there are two main strategies

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

to achieve this provisioning: reactive and proactive. A *reactive strategy* consists in providing energy only expecting that the demand in the next period is the same as in the previous period. There are two approaches in the reactive paradigm, one is based on providing an amount of energy for the next period equal to the amount of energy requested in the previous period. This is an open-loop strategy, since it does not require measuring the requests served in the previous period. This strategy is optimal from the energy point of view, since the amount of energy provided and the energy requested are equal. However, it does not provide a good QoS due to the possible significant mismatch between demand and provisioning. Since this approach works in open loop, it is not able to correct possible mismatches between demand and provisioning. Therefore, a second approach that uses on-line feedback of the requests served allows such mismatches to be corrected.

On the other hand, a *proactive strategy* which is based on predicting the amount of energy requested, by using past and present information, is better from the QoS point of view, but not necessarily from the energy point of view, unless the prediction is perfect, which, in practice, is impossible. Thus, we are faced with a trade-off between energy saving and QoS.

Since this trade-off is the main desired objective when designing dynamic control strategies, there are several approaches that can be used for DC management. The problem is, in fact, a multi-objective optimization subject to constraints. It consists of maximizing the quality of the service or the latency time while minimizing the cost in terms of energy to process the demand by controlling efficiently the energy provided. Since these two goals are contradictory, the solution is a set of optimal compromises called a Pareto set. Then, a compromise can be selected according to the user's preference. Instead of obtaining the whole set of optimal compromises at each period, which is infeasible in real time, we use a classical feedback control to optimize a quadratic cost of the weighted objectives, the *Model Predictive Control* (MPC). Each weight of the cost function gives an optimal solution, thereby instead of choosing the best solution in a set of optimal compromises the user chooses a particular weight. To this end, three different control strategies using a DC model based on balance equations are analyzed. The first is the *Reactive Control* (RC), the second is the *Reactive Feedback Control* (RFC) and the third is the *Model Predictive Control* (MPC) which takes into account the management effort. To use these control approaches, it is necessary to formulate a dynamic DC model as done in Section IV.

In this paper, we show that Reactive Control is very efficient in terms of energy but provides a poor Quality of Service. Based on the DC model developed in Section IV, Reactive Feedback Control uses a periodic measure of the imbalance. RFC improves the Quality of Service while consuming an amount of energy close to RC, but uses a great management effort. For this reason, we introduce MPC, which provides a better Quality of Service than RFC and additionally has a control on the management effort. This is the main contribution of our paper: MPC is by far the most efficient strategy to optimize 1) the management effort, 2) energy efficiency and 3) Quality of Service.

This paper is organized as follows. Section II presents some related work. Section III deals with the computation of the energy demand. Section IV defines the DC model and the three different control strategies studied. Results of the comparative performance evaluation of those three control strategies are given in Section V, based on a data set collected from a real DC. Finally, we conclude in Section VI.

II. RELATED WORK

In the literature, several authors address the trade-off between energy consumption of servers and user Quality of Service (QoS) satisfaction in Mobile Edge Computing (MEC), [24], [25] or in DC, [2], [12].

In MEC, the problem consists in selecting the MEC servers able to offload the computation of mobile devices while satisfying the QoS requested by users. In [24], this satisfaction problem is solved by a non-cooperative game where MEC servers learn how to decide to be "on" or "off", whereas mobile devices learn how to select an "on" server for computation offloading. The distributed solutions proposed in [24], [25] are based on game theory and machine learning. Both require message exchanges which consume time and network bandwidth. In this paper, we propose a centralized solution based on a dynamic DC model. This is a very powerful tool able to provide a very fast solution based on prediction.

The bibliography on DC management using proactive actions based on user request prediction is increasing due to the interest in energy efficiency and Quality of Service (QoS). An energy-efficient resource provisioning framework for cloud data centers is proposed in [2]. The authors report significant energy saving by predicting the number of user requests per category to compute the number on machines required to serve these requests. Unneeded machines are turned off to save energy. In [3] a proactive DC management strategy is presented together with its upper bound of cost savings obtained with respect to a purely reactive management. Delimitrou *et al.* proposed Quasar [4] to increase resource utilization in clusters, while providing high application performance.

An energy-efficient DC management should take into account the dynamicity and heterogeneity of jobs submitted by users. The aim of the DC management is to maximize its revenues. This profit maximization problem, is formulated as a mixed integer non-linear programming problem in [26] and [29]. Various methods are used to solve this problem such as i) simulated annealing and particle swarm optimization in [26], ii) genetic algorithm in [27], and iii) Lagrange relaxation followed by some adjustments of task placement and resource allocation in [29], to name but a few.

None of these strategies uses feedback explicitly. It is well known that feedback control can improve performances in computing systems and networks. The authors of [5] show

how feedback control can be used for memory management, CPU utilization in distributed real-time embedded systems, stable and automated workload management in virtualized Data Centers (DCs), power and QoS in DCs, and many other applications. In [6], the authors discuss the application of both centralized and distributed state-feedback for resource allocation of tasks in DCs to significantly improve performance and also resource saving. In [7], properties of several predictive controllers to reduce resource over-provisioning for enterprise applications using dynamic resource allocation are compared.

In the case of linear systems, there exist well-known optimal feedback solutions, see for example [8]. Applications in computer systems are generally presented as non-linear system control problems subject to constraints in states, input and output variables. In these conditions, it is very difficult to ensure optimal solutions using feedback. A widely used strategy for this type of problem is Model Predictive Control [9]. Using a model of the system, MPC consists in optimizing the control over a finite time-horizon (i.e. a sliding window of time intervals), while implementing only the control for the next time interval. Then again, MPC optimizes on the next sliding window, repeatedly.

A linear constrained model predictive control approach is applied in [10] to a coordinated dynamic server provisioning and thermal dynamics for energy control of data centers. In [11], it is applied to critical time-sensitive cyber-physical systems, taking advantage of cloud and edge computing. In [12], which is the closest work to ours, the objective is to adjust the incremental number of machines that are turned on or off in a cloud data center. In this study, the problem is formulated as a pseudo-optimization problem in two steps. First, by obtaining the number of machines that minimizes the operational cost expressed as the sum of the energy cost and the delay violation penalty cost. Second, an MPC strategy is used to minimize a constrained quadratic cost taking into account, on the one hand, the error between the number of machines and its optimal number and, on the other hand, the control penalty corresponding to the reconfiguration cost paid each time a machine is turned *on* or *off*. The weighting coefficients in the quadratic cost allow the DC manager to favor either an aggressive control to maximize power savings or, on the contrary, a smoother control to minimize the reconfiguration cost. Thus, the strategy consists in solving two costs by optimizing only the last one leading to a pseudo-optimal solution. This study is the closest to ours since we also use MPC to maximize energy savings and user satisfaction, while taking into account the reconfiguration. However, we will see that in our approach we use only one cost to optimize the energy cost, the delay violation and the reconfiguration cost, thereby making our solution optimal.

Several papers cited use a publicly available set of traces collected in one of Google's DCs over a period of 29 days [14], [16]. This DC data set has been extensively studied by researchers [15], [17]–[20]. The analysis of this data set can help researchers i) to make valid assumptions, ii) design more accurate models of jobs, tasks and machines, and iii) propose more efficient job scheduling algorithms. We apply our theoretical study to this data set and make a quantitative comparison.

TABLE 1. Notations.

	Name	Meaning	Unit
Load	T	DC management period, also called sampling interval	s
	$r_i^j(t)$	Requests the amount of resources r_i at time t_i on server j	Mips or Mbits*
	$r^j(t)$	Sum of resources requested at time t on server j	Mips or Mbits*
Energy	$y^j(k)$	Energy demanded in the time interval $[k-1)T, kT)$ on server j	J
	$y(k)$	Total energy demanded in the time interval $[k-1)T, kT)$	J
	$u(k)$	Total energy provided during $[k-1)T, kT)$	J
	$e(k)$	Error at step k taking into account the requests waiting to be served at time $(k-1)T$	J
Power	P^j_{idle}	Power consumed at null load by server j	W
	P^j_{peak}	Power consumed at full load by server j	W
	$P(t)$	Power consumed by the DC at time t	W
	$P_{idle}(k)$	Power consumed by the servers on in $[(k-1)T, kT)$	W
Resources	$M(k)$	Number of servers turned-on in the time interval $[k-1)T, kT)$	-
	$\mu^j_x(t)$	Utilization factor at time t of resource x on server j	rate
	η^j_x	Power variation coefficient as a function of utilization factor of resource x on server j	W
	$P_x(t)$	Power consumed in the DC by the utilization factor of resource x, it does not take into account the power needed to turn on the servers required	W
Cost	ρ	Weight of the management effort, $\rho \in [0, 10]$	-

* Mips for CPU and Mbits for memory.

III. ENERGY DEMAND

A. NOTATIONS AND ASSUMPTIONS

The notations used in this paper are listed in Table 1. For the energy computation, the following assumptions are used:

- A_0 Job arrivals are random.
- A_1 Amounts of requested resources are random.
- A_2 Request durations are random.
- A_3 DC management is in charge of maintaining the balance between the number of servers *on* and the amount of resources used during the period T. This is done by either turning *on* the necessary servers or turning *off* the unnecessary ones periodically with period T. The time interval $[(k-1)T, kT)$ is also denoted step k for simplicity reasons.

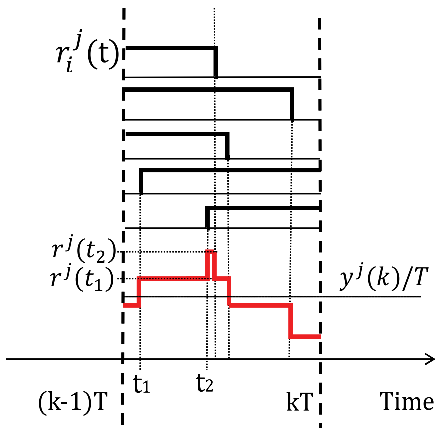


FIGURE 1. $r_i^j(t_i)$ **requests the amount of resources** r_i^j **from time** t_i **for duration** τ_i^j **(not represented here). The request is represented by a rectangular pulse of amplitude** r_i^j **for a time** t **meeting** $t_i \leq t \leq t_i + \tau_i^j$**, and zero otherwise; in red** $r^j(t)$ **is the sum of the amplitude of all rectangular pulses counted at time** $t \in [(k-1)T, kT)$**;** $y^j(k)$ **is the energy provided by server** j **during the interval** $[(k-1)T, kT)$**.**

- A_4 The allocation algorithm is in charge of assigning available servers to globally provide the requested resources in terms of CPU and memory, etc.
- A_5 When a job is waiting to be scheduled, it does not consume any energy.

B. ENERGY COMPUTATION

The allocation algorithm is in charge of assigning servers to globally provide the requested resources, of which concern there are different classes (e.g. CPU, memory, disk). Hence, any server j that is *on* in the time interval $[(k-1)T, kT)$ has to serve the i^{th} request in the DC that has been assigned to it. This request, denoted by $r_i^j(t)$, requires an amount of resources r_i^j from time t_i and for a duration τ_i^j, which is represented by a rectangular pulse of amplitude r_i^j and zero otherwise, as shown in Fig 1. The sum of the amplitude of all rectangular pulses at time $t \in [(k-1)T, kT)$ is the amount of requested resources at this time on server j and is given by:

$$r^j(t) = \sum_i r_i^j(t) \tag{1}$$

For example, Fig 1 depicts several requests, some arriving during the interval $[(k-1)T, kT)$ and others whose service started in a previous period but has not yet finished. The total amount of resources requested on j is depicted in red. The area under $r^j(t)$ during the interval time $(k-1)T$ to kT is proportional to the energy needed by server j to serve its the requests in this interval.

We denote by $P^j(t)$ the power consumed at time $t \in [(k-1)T, kT)$ by server j that is *on*, during this time interval. In order to obtain the expression of $P^j(t)$, we adopt the linear power model of the server, which has been extensively used by many authors, [21], [22], where the total power consumed is the sum of the idle power, P_{idle}^j, and the power due to each resource consumption. By defining the resource utilization factor of a given resource at time t equal to $\mu^j(t) = r^j(t)/C_{ap}^j$, the ratio between the amount of resource requested at time t and the resource capacity; $\mu_x^j(t)$ the utilization factor of resource $x \in \{x_1, \ldots, x_n\}$ in server j at time t; and η_x^j the power variation coefficient associated with the utilization factor of resource x on server j, the power consumed by server j at time $t \in [(k-1)T, kT)$ is given by:

$$P^j(t) = P_{idle}^j + \sum_{x \in DC} \eta_x^j \mu_x^j(t). \tag{2}$$

The summation is done over all n resource classes (e.g. CPU, memory, disk, etc.) that are made available by the DC, these resource classes do not vary over time. The power variation coefficients, η_x^j, can be obtained by filling with $N \gg n$ sampled data at different times t the following vectorized equation:

$$\left[\mu_{x_1}^j(t), \ldots, \mu_{x_n}^j(t)\right] \begin{bmatrix} \eta_{x_1}^j \\ \vdots \\ \eta_{x_n}^j \end{bmatrix} = P^j(t) - P_{idle}^j, \tag{3}$$

and solving it via the Least Squares method.

Let $M(k)$ denote the number of servers turned *on* in the DC at any time $t \in [(k-1)T, kT)$, the total power $P(t)$ consumed by the DC taking into account all resources used is equal to the sum of two components:

$$P(t) = P_{idle}(k) + \sum_{x \in DC} P_x(t) \tag{4}$$

where

$$P_{idle}(k) = \sum_{j=1}^{M(k)} P_{idle}^j \tag{5}$$

$$P_x(t) = \sum_{j=1}^{M(k)} \eta_x^j \mu_x^j(t) \tag{6}$$

The energy associated with all resource requests in the time interval $[(k-1)T, kT)$ can be written as:

$$y(k) = y_{idle}(k) + \sum_{x \in DC} y_x(k) \tag{7}$$

where:

$$y_{idle}(k) = TP_{idle}(k) \tag{8}$$

$$y_x(k) = \int_{(k-1)T}^{kT} P_x(t)dt \tag{9}$$

IV. DC MODEL AND CONTROL

The goal is to satisfy user requests while minimizing the energy consumed. This is accomplished by shutting down unnecessary servers to save power or turning *on* the necessary ones while serving heterogeneous user requests. For this purpose, we propose different control strategies by providing the necessary energy, $u_x(k)$, to meet the demand of each requested resource, $y_x(k)$. Each of these strategies applies to each of the n resource classes requested in the time interval

$[(k-1)T, kT)$, although here, for the sake of simplicity, we consider generically only one. Thus, the suffix x will be omitted. Since the power P_{idle} cannot be explicitly controlled, it is assumed that within the set of admissible servers with the same resource capacity, those with a minimum P_{idle} are chosen. Three control strategies based on the DC balance equations are studied, namely: i) Reactive Control (RC), which is an open-loop control that provides an amount of energy equal to the energy requested in the previous time interval, ii) Reactive Feedback Control (RFC), which is the same as RC but includes the accumulated demand that has not yet been served, and iii) Predictive Control Model (MPC), which optimizes a cost that weights management effort and prediction error. All these strategies are based on the dynamic model of DC determined by its balance equations.

A. DYNAMIC ENERGY BALANCE

The input variables of the dynamic DC model are, on the one hand, $u(k)$, the energy provided to serve user requests by managing the number of available servers and, on the other hand, $y(k)$ the energy required to serve user requests (e.g. CPU and memory). The demand is served proactively, providing at time $(k-1)T$, the energy $u(k-1)$ provisioned for the next period $[(k-1)T, kT)$, see Fig 2.

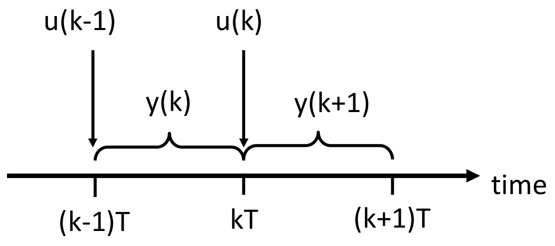


FIGURE 2. **DC management: $y(k)$ and $u(k-1)$ are the energy demanded and the energy provided in the DC during the interval $[(k-1)T, kT)$, respectively.**

The balance equation represents the dynamic accumulation of energy during a time interval T in the DC. It is equal to the difference between:

- the energy requested to serve the demands arrived during the period $y(k)$ plus the demands arrived in previous periods but not yet served, $\max\{0, e(k-1)\}$,
- and the energy provided, $u(k-1)$.

Then, the dynamic energy balance of the accumulated error is given by:

$$e(k) = \max\{0, e(k-1)\} + y(k) - u(k-1) \quad (10)$$
$$u(k) = u(k-1) + \Delta_u(k) \quad (11)$$

where $\max(0, e(k-1))$ stands for the amount of accumulated demands that have not been served when starting the time interval $[(k-1)T, kT)$. Considering the case where $e(k-1) = 0$, the error is positive when the demand is greater than the provision, $y(k) > u(k-1)$, and conversely, if the error is negative, $y(k) < u(k-1)$, there is more provision than demand. Hence, in Equation (10) the positive part of the error represents the accumulated demand not served previously and which needs to be served as soon as possible. On the other hand, when the energy provided is in excess, it is lost. As a consequence, only the positive part of the error is accumulated. The incremental change of energy provided, given by Equation (11), $\Delta_u(k) = u(k) - u(k-1)$, quantifies the variation in energy supplied to maintain the balance. This is called the management effort. Optimal performance consists in minimizing the error while keeping minimal the management effort. However, these two objectives are conflicting. Then, to obtain a good balance control, some strategies are required.

B. THREE CONTROL STRATEGIES

The different strategies proposed to obtain the requested energy are obtained from the error given by Equation (10) and under the assumption that the error $e(k)$ can be measured at step k.

1) REACTIVE CONTROL

The first strategy consists in providing at time kT for the time interval $[kT, (k+1)T)$, the amount of energy requested at step k, that is during the time interval $[(k-1)T, kT)$. Thus,

$$u(k) = y(k). \quad (12)$$

Initializing $e(0) = u(0) = 0$ and replacing $u(k) = y(k)$ in (10), the error is $e(k) = y(k)$ for all k. This strategy has two properties:

i) There is no waste of energy, since it is used to serve all requests exactly.
ii) All requests from the previous period whose service has not yet started at the end of the period are served in the next period.

Another strategy could be to accumulate the demand from the previous period and release it in the next, achieving zero error. Both strategies are energy-efficient but relatively poor from the point of view of Quality of Service due to the significant mismatch. Note that starting with $e(0) = 0$, the error in (10) is zero only if $y(k+1) = y(k)$.

2) REACTIVE FEEDBACK CONTROL

The problem with Reactive Control is that there is always a mismatch between the energy requested and the energy provided in the same period. In order to improve this, it is possible to take into account the accumulated error given by the amount of non-served requests in (10). Thus, an improved reactive energy provisioning is as follows:

$$u(k) = \max\{0, e(k)\} + y(k). \quad (13)$$

Now, initializing $e(0) = u(0) = 0$ and replacing $u(k)$ in (10), the error is reduced to $e(k) = y(k) - y(k-1)$ for all k. Note that also starting with $e(0) = 0$, the error is zero if $y(k+1) = y(k)$.

3) MODEL PREDICTIVE CONTROL

Generally, only a countable number of servers, say N, are available in the DC. Then, one possible solution is to use servers such that their global capacities are the closest as possible to the amount required. This procedure could lead to sub-optimal solutions. Thus, we propose a method that allows the best solution to be chosen, given the available servers, that minimizes the management effort. The strategy proposed is called *Model Predictive Control, MPC,* which can be stated as the following constrained minimization problem:

$$\min_{\Delta_u(k)} \left\{ \left(\max\{0, e(k)\} + \hat{y}(k+1) - u(k) \right)^2 + +\rho \Delta_u^2(k) \right\}$$

$$st: \begin{cases} e(k) = \max\{0, e(k-1)\} + y(k) - u(k-1) \\ u(k) = u(k-1) + \Delta_u(k) \\ \Delta_u(k) \in \{\Delta_1, \Delta_2, \ldots, \Delta_N, \} \\ \rho \geq 0 \text{ weight of the management effort} \\ e(0) = 0 \end{cases} \tag{14}$$

Note that in the case where $\rho = 0$ and $\hat{y}(k+1) = y(k)$, the RFC control strategy, defined in Equation (13), coincides with the optimal solution of this constrained optimization problem. This strategy consists in providing the demand predicted for the next period plus the error correction of the previous period. Then, the improvement of MPC for $\rho = 0$ with respect to RFC depends on how good the prediction of the future demand is. In order to proceed with MPC, we need a prediction.

We are interested now in finding a prediction of the requested energy to be used in the MPC. The optimal predictor, $\hat{y}(k) = g(Y)$, is a nonlinear function of previous samples given by the elements of the vector $Y = [y(k-1), \; y(k-2), \; \ldots]$. In order to find the function, let us minimize the expected value of the quadratic error, $\mathcal{E}[J]$, where

$$J = (y(k) - g(Y))^2, \tag{15}$$

This problem is equivalent to finding the function $g(Y)$ that minimizes the conditional expected value of $y(k)$ given the previous values $y(k-1), y(k-2), \ldots$, see Section 7.4 of [23]. Basically, written in terms of the conditional probabilities

$$g(Y) = \int_{-\infty}^{\infty} y \, p(y|Y) \, dy$$

where $p(y|Y)$ is the density of y, conditioned to a given Y. Given a vector $Y^* = [y^*(k-1), y^*(k-2), \ldots, y^*(k-n)]$, where n is the number of previous samples considered, the conditional distribution $p(y|Y^*)$ is obtained by i) finding in a set of training samples all the vectors $Y = [y(k-1), y(k-2), \ldots, y(k-N)]$ that match Y^* and then, ii) obtaining the subset $\mathcal{Y}$ with the corresponding value $y(k)$ corresponding to each. The optimal predictor, is given by the mean value of the subset $\mathcal{Y}$. The values of y are associated with vectors Y which are obtained by selecting them in a previously stored database. This database can be built off-line to be accessed in real time to compute the MPC solution.

The minimization procedure in the MPC is as follows:

i) given the measured accumulated demand at time k, which is equivalent to $\max\{0, e(k-1)\}$ in Equation (10), the measured energy $y(k)$ requested, and the value of past control $u(k-1)$, the error $e(k)$ is obtained.

ii) using the prediction $\hat{y}(k+1)$ the cost (14) is evaluated for each Δ_i for $i = 1, \ldots, N$. The increment that gives the minimum cost is chosen.

iii) update $u(k)$ and go to step *i*).

The three proposed strategies are experimentally evaluated in Section V.

V. EXPERIMENTAL RESULTS

The results reported in this paper are obtained by comparing the three different DC management strategies under study, considering the requests in the Google DC whose traces [16] were collected over a period of 29 days. This data set has been selected, because 1) it corresponds to an operational data center and 2) it has been made publicly available by Google. The quantitative values reported in this section are representative of the data center considered. However, this paper shows how to apply the method proposed to any data center. In these traces, no energy data is explicitly provided. However, we use the amount of resource requested and the duration of each task to compute the energy consumed by task execution as explained in Section III-B, where Figure 1 depicts how different resource requests are taken into account in the energy computation.

Since in the data set considered, 92.65% of servers have the same CPU capacity of 0.5 as noted in [3], we consider that all servers have the same CPU capacity. Each server is such that for any admissible demand the limiting factor is the CPU. In other words, for all the tasks to be served, the CPU utilization factor is always greater than or equal to the memory utilization factor. Thus, the CPU is the bottleneck resource. Consequently, the control will be analyzed based on the imbalance only for the accumulated demand of CPU. The great heterogeneity of tasks in a DC, which has been pointed out by many authors [2], [13], [15], is taken into account by the method proposed. For instance, some tasks are more CPU-intensive than others and require more CPU. This is expressed by the task contribution to the CPU utilization factor on server j ($\mu^j_{CPU}(t)$), which may considerably differ from one task to another. We assume that the DC is made up of 1660 servers, each with the parameters given in Table 2. The peak power of the DC, without considering the idle power, is 5×10^5 [*Watts*]. Then, the peak energy consumed during a given time interval T, is $u_{peak} = 5 \times 10^5 \; T \; Joules$. In this paper, we use a DC management period T in the range $[60, 3600]$ *seconds*. The smallest value of T is determined in such a way that turning off a server during $T = 60$ s and then turning it on in the next time interval allows a very small amount of energy to be saved [2], [3]. The greatest value of $T = 3600$ s is chosen to limit user dissatisfaction in case of under-provisioning.

TABLE 2. Parameter values.

Param.	Meaning	Value
P_{peak}	Power required by a machine at full CPU load	600W
P_{idle}	Power required by a machine at null CPU load	300W
η	Power variation coefficient as a function of CPU load $\eta = P_{peak} - P_{idle}$	300W

A. ENERGY AND SAMPLED ENERGY

The amount of resources to serve user requests is quantified in terms of energy, that is, a quantity proportional to the amount of requested resources accumulated during the interval time T, named $y(k)$ as defined in (7). It is worth noting that this measure differs from the amount of resources sampled at time $t = kT_o$ and multiplied by T_o, named sampled energy and denoted by $y^*(k)$. For example, the usage of each type of resource is reported at each $T_o = 300s$ interval. The relative Euclidean norm of the difference between $y(kT)$ and the sampled energy $y^*(kT_o)$ is used to evaluate the relative error between these two approaches when $T_o = 300s$ for different values of time, T, as shown in Figure 3. It can be seen that this metric increases as the period T becomes smaller, reaching errors of up to 75% for $T = 60$ seconds. Figure 4 depicts the values of y, y^* and $y - y^*$ measured each $T = T_o = 300$ seconds. It is worth noting that the error is zero mean.

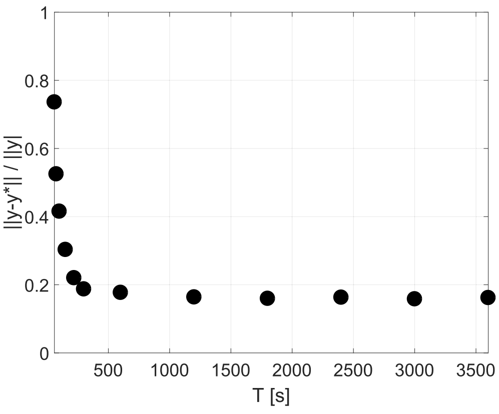


FIGURE 3. Relative Euclidean norm of the differences between energy $y(k)$ and sampled energy $y^*(k)$, for $T_o = 300s$..

B. ENERGY ERROR AND MANAGEMENT EFFORT

Control strategies provide the number of servers according to the energy balance at each period T. This balance assumes a constant arrival flow of demand during the interval, which is equivalent to consider constant demanded power during each period. However, the arrival power is generally time-varying so that it is possible that additional amount of energy supplied could be not fully utilized. For example, it could happen that for the same energy demanded in a period, it arrives towards the end of the period, causing an additional underutilized energy supplied. Then, to analyze QoS and energy waste

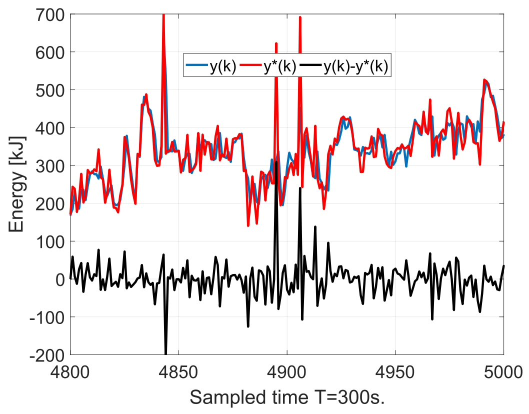
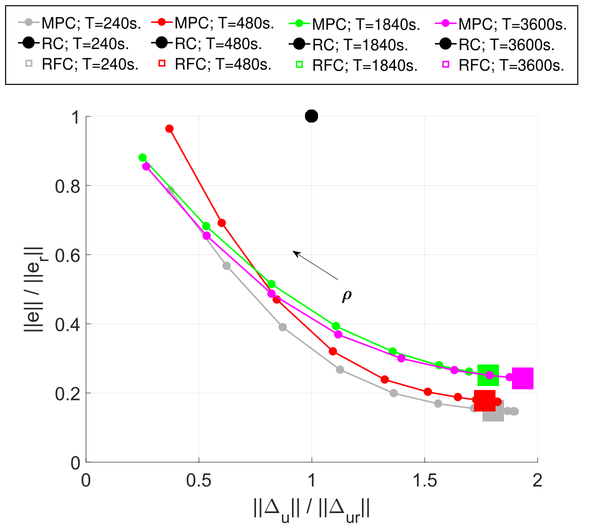


FIGURE 4. Evolution over time of the amount of energy $y(k)$ and sampled energy $y^*(k)$ and differences between both for $T = T_o = 300s$.

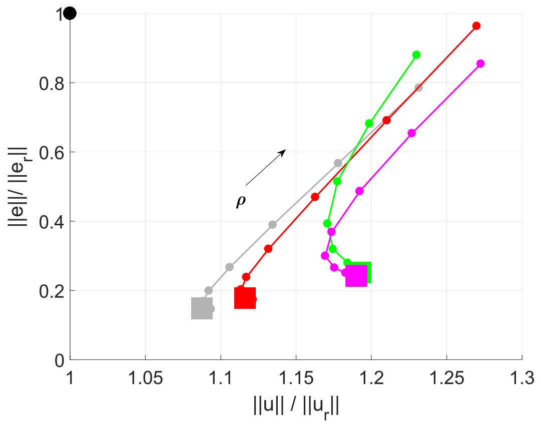
for different dispatch periods T, simulations are performed using multi-sampling. A small period of 10 s is used to take into account the time-distribution of the arrival of demand in the balance equation (10) and (11). The provisioned energy $u(kT)$ is dispatched each period $T = nT_s$, where n is an integer number of small periods periods. Figure 5 depicts the energy imbalance of the different control strategies versus the management effort for dispatch periods T equal to 240, 480, 1840, and 3600 seconds. For MPC, the increments or decrements, $\Delta_u(k)$, are chosen by turning on or turning off the necessary number of servers among the 1660 servers. The Euclidean norm of the error versus the incremental control, both relative to reactive control is depicted in Figure 5.a for different values of the management period T in a range of $\rho \in (0, \ 10)$. In this Figure, we can see all the solutions that optimize the quadratic cost where a stronger control of increment allows a reduction in the imbalance between energy demanded and energy provided.

Since the demands served are represented by the positive part of the imbalance while the negative part represents the wasted energy, the performance of each control strategies depends on how much the variable $e(k)$ can be reduced. Thus, from the Figures, MPC optimally improves QoS and control effort. In Figure 5a, it is observed that the norm of the imbalance can be reduced by app. 60% with the same management effort using MPC with respect to RC. By reducing the management effort, the error increases depending on the value of ρ and on the contrary, by increasing the management effort it is possible to reduce by app. 80% the imbalance. From Figure 5b this imbalance reduction costs approximately between 10% and 30% more energy. It is important to note that for $\rho = 0$ the differences between MPC and RFC are explained by the predictor used.

On the other hand, we know that RC is energy efficient, thus we can assert that the other strategies share the same property. This is corroborated by observing the histogram of the errors in Figure 6a, 6b, and 6c in which the error is almost always positive, $y(k) > u(k)$ for all k, which means that all energy provided is efficiently used. In Figures 6b and 6c, we observe, due to the feedback, that the accumulated

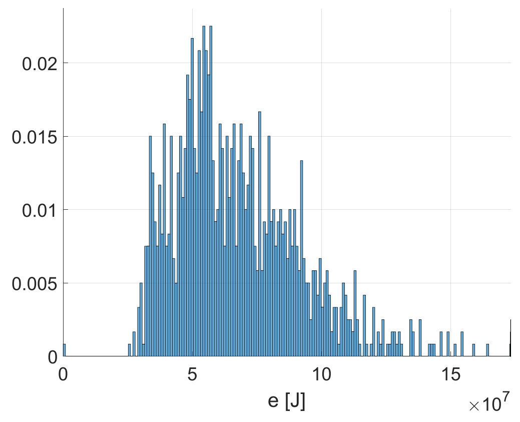


(a) Relative error norm of energy versus relative management effort.

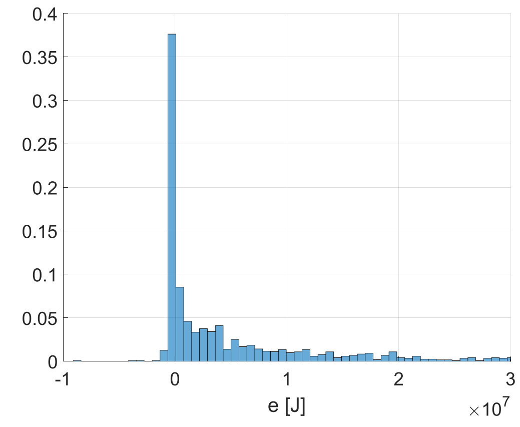


(b) Relative error norm of energy versus relative norm of energy provided.

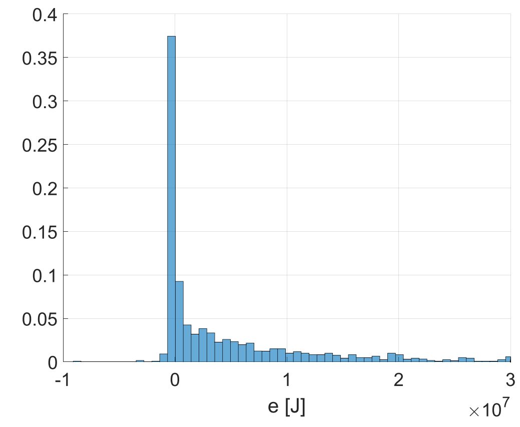
FIGURE 5. Model predictive control MPC, reactive Feedback control RFC, and reactive control RC strategies for different values of intervals T.



(a) Reactive Control.



(b) Reactive Feedback Control.



(c) Model Predictive Control.

FIGURE 6. Relative error probability for interval $T = 1840s$.

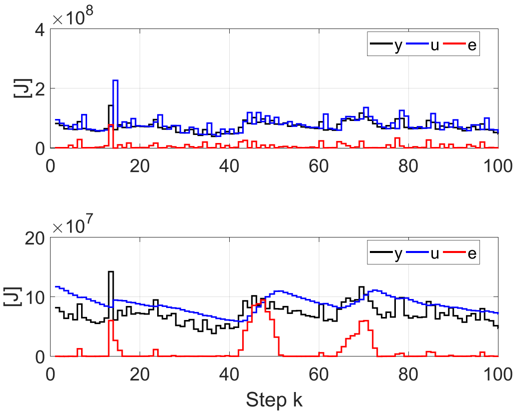
demand given by the distribution of positive values of the imbalance is significantly reduced in comparison with the reactive control in Figure 6a. Moreover, using feedback the error is optimally concentrated near zero. In the next subsection, we will see that this reduction of accumulated demand is related with the reduction of the scheduling time.

C. EVOLUTION OVER TIME

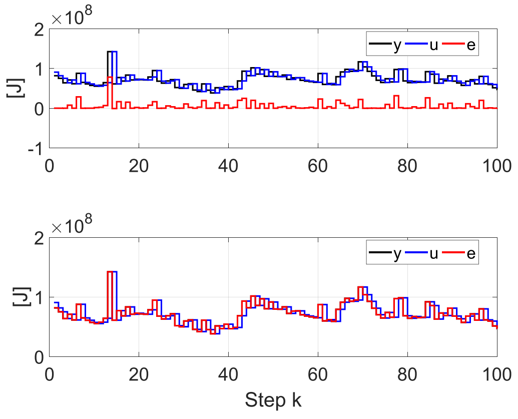
In order to observe the effect of the weight ρ on the time responses of the variables we choose as an example a portion of the record for different signals in the case of $T = 1840s$. Figure 7 depicts the evolution over time of the energy requested $y(k)$, the energy provided $u(k)$ and the error $e(k)$ for the three strategies evaluated. Figure 7a (upper) depicts the MPC time response for $\rho = 0$ which means no penalty of the reconfiguration effort, whereas Figure 7a (lower) depicts the time response for $\rho = 10$ meaning a moderate penalty. It can be seen that in the former the error of MPC is similar to the RFC depicted in Figure 7b (upper). In the case of MPC for $\rho = 10$, the control action is softer as can be seen from Figure 7b (lower).

D. AVERAGE SCHEDULING DELAY

From the standpoint of the data center manager, the main factor that defines the QoS is the scheduling delay which is also important for sustaining the SLA (Service Level Agreement). Several simulations were carried out to obtain the average scheduling delay which is depicted in Figure 8 for



(a) Model Predictive Control, MPC. Upper $\rho = 0$. Lower, $\rho = 10$.



(b) Upper reactive feedback control RFC and lower, reactive control RC.

FIGURE 7. Evolution over time of $y(k)$ the energy requested, $u(k)$ the energy provided, and $e(k)$ the error, for interval $T = 1840s$..

several time-intervals T and $\rho \in [0, 10]$ with respect to the energy consumed relative to the peak energy u_{peak}. It can be observed that for all time intervals T, the scheduling delay decreases with large values of ρ, corresponding to an increased control action. The minimum scheduling delay ranges from $30s$ for small T to $100s$ for large T, and the corresponding energy consumption remains less that 0.1 in all cases. This result confirms that the more reduced the dispatch interval is, the shorter the scheduling time is, because the system is better controlled. The energy consumed decreases with the reduction of the scheduling time due to a stronger action of the control effort that is achieved by increasing ρ. This is because both the non served demand, represented by the positive part of $e(k)$, and the energy waste, represented by the negative part of $e(k)$, are reduced simultaneously by the MPC through the optimization of the quadratic cost of $e(k)$. The energy consumed to provide a given average scheduling delay is considerably smaller than other approaches using the same data. For example, the solution presented in [12] requires a relative consumption of 0.5 to obtain an average scheduling time of $80s$, whereas optimal MPC requires less than 0.1 relative energy for the same average scheduling delay. This is due to the use of an optimal solution rather than

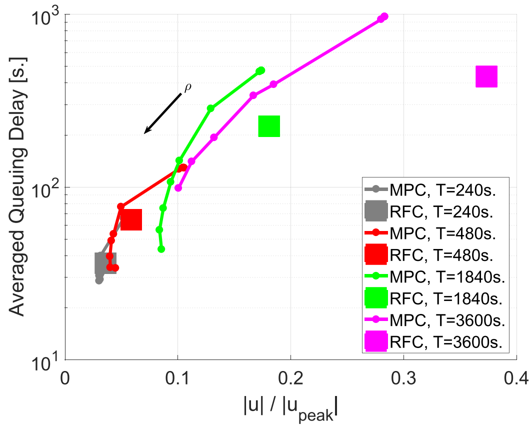


FIGURE 8. Average queuing delay versus energy relative to the peak energy.

a sub-optimal one. In the case of RFC, both the scheduling time and the energy consumed are greater than those obtained with the optimal MPC. Whatever the solution evaluated, both the scheduling time and the energy consumed decrease with a reduction of the DC-dispatch period. However, for periods T less than $500s$, the minimum scheduling time and the energy consumed do not change significantly.

VI. CONCLUSION

The DC manager is faced with two conflicting goals: on the one hand, to provide the best QoS to users by reducing the scheduling delay by increasing the number of servers *on*, and on the other hand, to save energy by turning *off* all unused servers. Dynamic capacity provisioning is applied periodically to tune the DC capacity according to the dynamic and heterogeneous workload submitted to the DC. In this paper, we compare three DC control strategies. The first one, Reactive Control (RC), is an open-loop control which provides the energy requested in the previous period. Consequently, it provides the exact amount of energy requested, which makes it energy efficient, but with a big large latency, leading to a significant scheduling delay. The second one, Reactive Feedback Control (RFC), is a closed-loop strategy; it needs to measure the service on-line. It uses the accumulated error from the previous period to decide the provisioning period allowing the requests to be served in less than two time periods, but is not energy efficient. The third one is Model Predictive Control (MPC) which is a feedback control and is also proactive, since it uses prediction of the demand. MPC is an optimal solution between weighted error and control effort. From the comparison of these three control strategies applied to the traces of a real DC, three conclusions can be drawn: 1) The proposed optimal proactive MPC, that simultaneously minimizes the waste of energy with maximum possible QoS by managing the control effort, significantly improves reactive open-loop control. This scheme achieves scheduling times of the order of 10% of those used by reactive control while only increasing the energy used by 10%. It is important to note that this reduction is the maximum possible with this type of proactive scheme since the design is optimal. 2) DC dispatch time is shown to affect Quality of Service in all three control settings, but not energy savings, which remain almost the same as RC energy efficiency. 3) It is shown that when the control effort is not taken into account, the same optimal performance as in MPC is obtained using RFC.

REFERENCES

[1] S. Mubeen, P. Nikolaidis, A. Didic, H. Pei-Breivold, K. Sandstrom, and M. Behnam, "Delay mitigation in offloaded cloud controllers in industrial IoT," *IEEE Access*, vol. 5, pp. 4418–4430, 2017.

[2] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Energy-efficient resource allocation and provisioning framework for cloud data centers," *IEEE Trans. Netw. Service Manage.*, vol. 12, no. 3, pp. 377–391, Sep. 2015.

[3] R. Milocco, P. Minet, E. Renault, and S. Boumerdassi, "Evaluating the upper bound of energy cost saving by proactive data center management," *IEEE Trans. Netw. Service Manage.*, early access, Apr. 16, 2020, doi: 10.1109/TNSM.2020.2988346.

[4] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and QoS-aware cluster management," in *Proc. 19th Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, 2014, pp. 127–144.

[5] T. Abdelzaher, Y. Diao, J. L. Hellerstein, C. Lu, and X. Zhu, "Introduction to control theory and its application to computing systems," in *Performance Modeling and Engineering*, Z. Liu and C. H. Xia, Eds. Boston, MA, USA: Springer, 2008, doi: 10.1007/978-0-387-79361-0_7.

[6] P. Marti, C. Lin, S. A. Brandt, M. Velasco, and J. M. Fuertes, "Optimal state feedback based resource allocation for resource-constrained control tasks," in *Proc. 25th IEEE Int. Real-Time Syst. Symp.*, Dec. 2014, pp. 161–172.

[7] W. Xu, X. Zhu, S. Singhal, and Z. Wang, "Predictive control for dynamic resource allocation in enterprise data centers," in *Proc. IEEE/IFIP Netw. Oper. Manage. Symp. (NOMS)*, Vancouver, BC, Canada, Apr. 2006, pp. 115–126.

[8] K. J. Åström, B. Wittenmark, *Computer-Controlled Systems*, 3rd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1997.

[9] E. F. Camacho and C. Bordons, *Model Predictive Control*. London, U.K.: Springer-Verlag, 2007.

[10] Q. Fang, J. Wang, H. Zhu, and Q. Gong, "Using model predictive control in data centers for dynamic server provisioning," in *Proc. 19th World Congr. Int. Fed. Autom. Control (IFAC)*, 2014, pp. 24–29.

[11] P. Skarin, J. Eker, M. Kihl, and K.-E. Årzén, "An assisting model predictive controller approach to control over the cloud," 2019, *arXiv:1905.06305*. [Online]. Available: http://arxiv.org/abs/1905.06305

[12] Q. Zhang, M. F. Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein, "Dynamic energy-aware capacity provisioning for cloud computing environments," in *Proc. 9th Int. Conf. Autonomic Comput. (ICAC)*, New York, NY, USA, 2012, pp. 145–154.

[13] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 14–28, Jan. 2014.

[14] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: Format + schema," Google Inc., Mountain View, CA, USA, Tech. Rep., Nov. 2011. [Online]. Available: https://github.com/google/cluster-data

[15] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proc. 3rd ACM Symp. Cloud Comput. (SoCC)*, San Jose, CA, USA, Oct. 2012, pp. 1–13.

[16] J. Wilkes. (Nov. 2011). *More Google Cluster Data, Google Research Blog*. [Online]. Available: http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html

[17] S. Di, D. Kondo, and W. Cirne, "Host load prediction in a Google compute cloud with a Bayesian model," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.* Salt Lake City, UT, USA, IEEE Computer Society Press, Nov. 2012, pp. 1–11.

[18] O. Beaumont, L. Eyraud-Dubois, and J.-A. Lorenzo-del-Castillo, "Analyzing real cluster data for formulating allocation algorithms in cloud platforms," in *Proc. IEEE 26th Int. Symp. Comput. Archit. High Perform. Comput.*, Paris, France, Oct. 2014, pp. 302–309.

[19] M. Alam, K. A. Shakil, and S. Sethi, "Analysis and clustering of workload in Google cluster trace based on resource usage," *CoRR*, vol. abs/1501.01426, 2015.

[20] P. Minet, E. Renault, I. Khoufi, and S. Boumerdassi, "Analyzing traces from a Google data center," in *Proc. 14th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Limassol, Cyprus, Jun. 2018, pp. 1167–1172.

[21] F. Chen, J. Grundy, Y. Yang, J.-G. Schneider, and Q. He, "Experimental analysis of task-based energy consumption in cloud computing systems," in *Proc. ACM/SPEC Int. Conf. Int. Conf. Perform. Eng. (ICPE)*, 2013, pp. 295–306.

[22] P. Xiao, Z. Hu, D. Liu, G. Yan, and X. Qu, "Virtual machine power measuring technique with bounded error in cloud environments," *J. Netw. Comput. Appl.*, vol. 36, no. 2, pp. 818–828, Mar. 2013.

[23] A. Papoulis, *Probability, Random Variables and Stochastic Process*. New York, NY, USA: McGraw-Hill Book Company, 1965.

[24] P. A. Apostolopoulos, E. E. Tsiropoulou, and S. Papavassiliou, "Game-theoretic learning-based QoS satisfaction in autonomous mobile edge computing," in *Proc. Global Inf. Infrastruct. Netw. Symp. (GIIS)*, Oct. 2018, pp. 1–5.

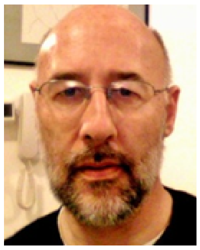
[25] S. Ranadheera, S. Maghsudi, and E. Hossain, "Mobile edge computation offloading using game theory and reinforcement learning," 2017, *arXiv:1711.09012*. [Online]. Available: http://arxiv.org/abs/1711.09012

[26] J. Bi, H. Yuan, W. Tan, M. Zhou, Y. Fan, J. Zhang, and J. Li, "Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 2, pp. 1172–1184, Apr. 2017.

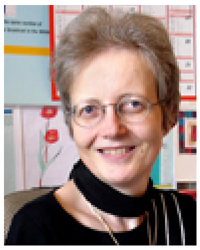
[27] W. Li, Y. Xia, M. Zhou, X. Sun, and Q. Zhu, "Fluctuation-aware and predictive workflow scheduling in cost-effective infrastructure-as-a-service clouds," *IEEE Access*, vol. 6, pp. 61488–61502, 2018.

[28] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li, and J. Li, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

[29] B. Hu, Z. Cao, and M. Zhou, "Scheduling real-time parallel applications in cloud to minimize energy consumption," *IEEE Trans. Cloud Comput.*, early access, Nov. 28, 2019, doi: 10.1109/TCC.2019.2956498.



RUBEN MILOCCO received the Ph.D. degree from the Universidad Nacional de La Plata, (UNLP). He was a Professor with UNLP. He is currently a Research Scientist with the Grupo de Control Automático y Sistemas (GCAySS), Universidad Nacional del Comahue (UNCo), and a Consulting Professor with the Department of Electrical Engineering, UNCo. He is also a member of the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Argentina. His research interests include filtering, estimation, and identification with applications to wireless and mobile networks and efficient energy storage management systems.



PASCALE MINET received the Engineering degree in computer science from ENSEEIHT, in 1980, the Ph.D. degree in computer science from the University of Toulouse, in 1982, and the HDR degree (qualification to supervise Ph.D. students) from the University of Versailles, in 1998. She is currently a Senior Researcher with the EVA Team, INRIA, Paris. More recently, she has focused her interest on machine learning techniques applied to networks. She studies how prediction can improve the energy efficiency of a data center, the hit ratio in content delivery networks, and the quality of service in wireless mesh networks. She has supervised 15 Ph.D. theses. She is the coauthor of the OLSR routing protocol standardized at IETF. She has published more than 200 papers in international conferences, journals, and book chapters. Her research topics relate to the Internet of Things. More particularly, she focuses on adaptive scheduling in multichannel wireless mesh networks, time-slotted channel hopping (TSCH) network formation, and path planning of mobile robots for data gathering.

ÉRIC RENAULT received the M.Sc. degree in computer engineering (diplôme d'ingénieur) from ISTY and the M.Sc. degree in computer science (DEA) from UVSQ in 1995, the Ph.D. degree in computer science from UVSQ in 2000, and the HDR degree (qualification to supervise PhD students) from UPMC, in 2011. He is currently a Full Professor with ESIEE Paris and a member of LIGM (UMR CNRS 8049) with Université Gustave Eiffel, France. He has worked on several European projects and served as an expert for the evaluation of French national projects (ANR), private company research studies (MESRI), and Eiffel Excellence Scholarship Program applications (Campus France). He has authored more than 100 articles in international journals and conferences. His research interests include high-performance computing and messaging, compilation, virtualization, positioning, and lightweight security for mobile, sensor, and vehicular networks.

SELMA BOUMERDASSI (Associate Member, IEEE) received the M.Sc. degree in computer engineering (diplôme d'ingénieur) from ESI, Algeria, and the M.Sc. degree in computer science and the Ph.D. degree in computer science from UVSQ, France. She was an Assistant Professor with UVSQ. Her research interests include wireless and mobile networks, with a special focus on the impact and use of social networks, trajectory prediction, information dissemination, and lightweight security. She is currently an Associate Professor with CNAM and a Research Associate with INRIA, Paris. She has worked on several national projects and served as an expert for the evaluation of national projects in France (ANR), Algeria (CDTA), and Canada (NSERC). She is an expert for MESRI, France, where she is in charge of evaluating the research work of private companies. She has authored more than 100 articles. She has served as a TPC Member for various international journals and conferences.