# Evaluating the upper bound of energy cost saving by proactive data center management

Ruben Milocco, Pascale Minet, Eric Renault, Selma Boumerdassi

# Evaluating the Upper Bound of Energy Cost Saving by Proactive Data Center Management

Ruben Milocco* Pascale Minet† Éric Renault‡ and Selma Boumerdassi§
* GCAyS, UNComahue, Buenos Aires 1400, 8300 Neuquén, Argentina. Email:milocco@uncoma.edu.ar
† Inria, 2 rue Simone Iff, CS 42112, 75589 Paris Cedex 12, France. Email: pascale.minet@inria.fr
‡ LIGM, Univ. Gustave Eiffel, CNRS, ESIEE Paris, 93162 Marne-la-Vallée, France. Email: eric.renault@esiee.fr
§ CNAM/CEDRIC, 292 rue Saint Martin, 75003 Paris, France. Email: selma.boumerdassi@cnam.fr

***Abstract*—Data Centers (DCs) need to periodically configure their servers in order to meet user demands. Since appropriate proactive management to meet demands reduces the cost, either by improving Quality of Service (QoS) or saving energy, there is a great interest in studying different proactive strategies based on predictions of the energy used to serve CPU and memory requests. The amount of savings that can be achieved depends not only on the selected proactive strategy but also on user-demand statistics and the predictors used. Despite its importance, it is difficult to find theoretical studies that quantify the savings that can be made, due to the problem complexity. A proactive DC management strategy is presented together with its upper bound of energy cost savings obtained with respect to a purely reactive management. Using this method together with records of the recent past, it is possible to quantify the efficiency of different predictors. Both linear and nonlinear predictors are studied, using a Google data set collected over 29 days, to evaluate the benefits that can be obtained with these two predictors.**

***Index Terms*—Data center management, Proactive management, Machine Learning, Prediction, Energy cost.**

## I. Introduction

In its "10 predictions for the data center and the cloud in 2019" [1], the Network World journal observes that due to the increasing demand in processing power, Data Center (DC) growth will continue, and machine learning techniques will play a major role in DC management, by optimizing the DC resources through continuous monitoring and adjustment. This periodic adjustment of active machines to serve job requests is made necessary by the high dynamicity of the jobs submitted to the DC. The main features of the strategy proposed in this paper can be summarized as follows:

- This strategy is based on prediction and reaction. The prediction consists in anticipating the energy required to serve the users requests that will arrive in the next time period, whereas the reaction consists in correcting the prediction error made over the previous period.
- The proposed strategy keeps the balance between demand and supply in one period for perfect prediction and at most two periods in the case of mismatches.
- The goal of this study is to quantify the energy cost reduction brought by this DC management. This reduction is evaluated by the Relative Error cost Saving (RES) defined as the relative difference between proactive and reactive costs. Then, optimal predictors, linear and nonlinear, are derived in order to maximize RES. Moreover, the upper bound of RES is analytically obtained. This upper bound can help the DC manager to select the predictors used.
- Numerical simulations with data collected from a real DC show that the proposed strategy used with two predictors, one linear and the other nonlinear, gets an improvement up to 70% with respect to a purely reactive action.

The original contribution of this paper is twofold. The first one lies in the direct prediction of the energy consumed in the next time interval, based on the recent past, whereas classical approaches predict resource requests. We first compute the energy consumed by the resources used: the magnitude of the resources multiplied by the time they are used. The energy prediction is made based on these measurements in a recent past horizon. The studies cited in the state-of-the-art 1) predict the resource requests, 2) use a placement algorithm to dispatch the requests, 3) deduce which machines to turn on or off, and finally 4) deduce the energy consumed. We directly predict the energy consumed. The second original contribution of the paper is to consider that the impact of overestimation and underestimation can be different. Hence, minimizing the absolute value of the prediction error does not necessarily maximize the energy cost saving.

The remainder of this paper is organized as follows: Section II presents a brief state-of-the-art about DC management based on prediction. Section III defines our framework for modeling and predicting energy consumption in a DC. It also introduces the methodology proposed. Section IV shows how to evaluate the energy consumed by the DC in past time intervals and how to predict the energy consumed in the next time interval, taking into account the heterogeneity of machines. The upper bound of the relative energy cost saving that can be achieved by a predictive DC management, is computed in Section V. Section VI and Section VII explain how to obtain the optimal linear predictor and the optimal nonlinear predictor, respectively. Section VIII deals with practical issues related to a real DC. In Section IX, all the theoretical results are applied to a real Google DC. Section X discusses the results obtained. Finally, Section XI concludes this paper.

## II. Related work

Modeling energy consumption in DCs has been the focus of numerous studies. In their in-depth survey [3], Dayarathna et al. studied 200 models hierarchically organized into two

branches. The software-centric branch deals with Applications (e.g. MapReduce), OS and Virtualization, and Machine Learning, whereas the hardware-centric branch distinguishes between the power conditioning system, the cooling system, optical networks, network devices, and servers. A server consists of network interfaces, server storage (e.g. SSD, HD), memory, processors which may be single core or multi-core, and specialized (e.g. GPU) or not. They observed that energy consumption has been studied intensively at the lower levels but much less at the DC level. In addition, the most frequent energy model for servers is the linear model, where the energy consumed by the server varies linearly with its workload [4], [5], [6]. This model is also adopted in this paper.

Google made publicly available a set of traces collected in one of its DCs over a period of 29 days [7], [2]. This DC data set has been extensively studied by researchers [8], [9], [10], [11], [12]. 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. The ultimate goal is to improve DC management by means of machine learning. The two most widely used machine learning techniques are classification and prediction.

Classification has been used for machine configuration in [12], where a machine configuration is defined as a vector whose first component denotes the CPU capacity and the second one the memory capacity. Classification has also been applied to jobs. For instance, the K-Means method allows the authors of [11] to distinguish between short jobs which are the most frequent ones and the lowest resource consumers, medium jobs which are frequent ones and the largest memory consumers, and long jobs which are the least frequent ones and the largest CPU consumers.

Prediction allows DC managers to take their decisions based on the prediction of the future value of a time series. The smaller the prediction error, the more accurate the management decision. The simplest predictors are linear ones. Linear predictors estimate the parameters of a linear model using a set of observed values such that the prediction is computed as a linear combination of the previous observations and possibly their prediction errors. The most famous family of linear predictors is Auto Regressive Moving Average with Exogenous signal (ARMAX). Liu et al. [13] show that the Moving Average (MA), the Auto Regressive (AR), and the Weighted Moving Average (WMA) predictors give smaller prediction errors with lower complexity compared to Neural Networks. Prevost et al. [14] reach the same conclusion: AR is far superior to neural networks in predicting the load demand in DCs. ARMA predictors are also used to predict the number of job arrivals in the next 30 minutes, with a Mean Absolute Percentage Error (MAPE) close to 0.38 [15]. Several variants of ARMA predictors (e.g. integrated, fractional, seasonal) have been compared with an exponential smoothing predictor and a predictor based on singular spectrum analysis [16]. They are applied to predict CPU, RAM, and network consumptions for the next hour.

However, when data are generated by unknown models, it is preferable to use predictors that do not make any a priori assumptions on the model. Cao et al. [17] use random forests to predict the workload of servers in a DC and automatically detect overload. The nearest-neighbour method is a popular nonlinear approach [18], [19]. It is based on the observations of states and their subsequent evolution. To predict a new, not previously observed, state the method finds the most similar states in the observation data set and builds its prediction from them. Since it is unlikely to find the same state, the method finds some closest states and predicts the average of their behavior. Assumptions like smooth variation of the states, and enough amounts of available data are needed. Sometimes, instead of predicting the average of the $k$ nearest neighbours, a weighted average is computed where the observation weight decreases when its distance to the expected state increases. This weighting function is called the kernel [20]. In our approach, the concept of conditional probability is used as in [20], [21].

Several authors have investigated different solutions to improve energy efficiency in cloud computation. For instance, Wolke et al. in [22] showed that with typical workloads of transactional business applications, dynamic resource allocation based on live migration technology does not increase energy efficiency compared to static allocation of VMs (Virtual Machines) to servers. Here, we can note that DCs do not support only business applications. As a consequence, the workload is more heterogeneous and varies more over time. That is why we will focus on dynamic approaches based on prediction.

Delimitrou et al. proposed Quasar [23] to increase resource utilization in clusters, while providing high application performance. Quasar is in charge of monitoring workload performance to adjust resource allocation and assignment if required. Instead of expressing their resource requests, users only express performance constraints for each workload type (e.g. latency-critical, Hadoop framework, single node). By means of classification techniques, Quasar estimates the application performances when the number of servers, the server type, the amount of resources within a server, and the interference of other workloads vary. It uses the results of this classification to jointly perform resource allocation and assignment. The authors show that with Quasar, resource utilization is improved by 47% while the performance target of each workload type is met. It can be noticed that Quasar does not use load prediction to manage the cluster.

In [24], Zhang et al. propose to apply the Dynamic Capacity Provisioning (DCP) to clouds. DCP dynamically adjusts the number of active machines to reduce energy consumption, while maintaining short scheduling delays to increase user satisfaction. In their model, time is divided into intervals of equal duration, and control decision about the number of machines to turn on or off is made at the beginning of each time interval, based on the usage of resources (CPU and memory) predicted by ARIMA. All machines are assumed to have the same capacities in memory and CPU. The authors show that their solution reduces by up to 18.5% the energy consumption in the Google data center considered [7]. In our paper, we adopt a similar approach based on time intervals.

In [25], Zhang et al. extend the Dynamic Capacity Provi-

sioning (DCP) approach to clouds with heterogeneous workloads and heterogeneous machines. The heterogeneous workload is divided into multiple task classes with similar characteristics in terms of resource and performance objectives. First, tasks are classified according to their CPU and memory requests with k-Means. Since the task execution duration is known only when the task is finished, each task is first assumed to be short and if not, its duration is re-evaluated to long. DCP is formulated as an optimization problem that considers machine and workload heterogeneity as well as reconfiguration costs. Using predictions (ARIMA) of resource requests, an online control algorithm based on the Model Predictive Control framework that solves the optimization problem at runtime is used. The authors show that their solution, called Harmony, can reduce energy consumption by 28% with regard to solutions that do not take into account the heterogeneity of both the workload and the machines.

In [26], Dabbagh et al. propose an energy-efficient resource provisioning framework for cloud data centers. They first identify categories of requests by means of k-Means clustering, where all requests of the same category have similar characteristics. Then, with one Wiener filter per request category, they predict the number of requests per category that will arrive in the next time interval. The requested resources associated with each request of any given category are assumed to be close to the requested resources of the centroid of the category considered. These estimations are used to compute the number of machines that must be on to serve the requests. Finally, unneeded machines are put to sleep in order to improve energy-efficiency. The authors evaluate their framework on the same Google data set as that used in this paper. They show that, in the case of a perfect prediction, up to 100 megajoules per minute can be saved by turning off unused machines. However, real 50 MegaJoules per minute can be used by keeping on some amount of extra machines in the case of mismatch to reduce user dissatisfaction.

The last two studies, [25] and [26] have the same final goal as ours, i.e., to improve energy efficiency in DCs by turning off the unused machines. However, they differ in the way the goal is to be reached. Both [25] and [26] predict the amount of resource requests in the next time interval to decide the number of machines that will be needed. The diverse applications with different priorities, performance and resource requirements together with the heterogeneity of both machines and workloads are taken into account to decide the number and capabilities of machines to serve the demand. In [25] an ARIMA model structure is used to predict the amount of requests while in [26], a Wiener adaptive filter scheme is employed. In this paper, our approach is different, since i) it is the energy consumed which is predicted and not the resource requests, and ii) the number of machines is computed from optimal linear and nonlinear predictors in order to maximize the energy cost saving and not to minimize the prediction error as in [26] or a quadratic cost taking into account the error and the reconfiguration cost as in [25]. Thus, instead of keeping a threshold of extra machines on, we design predictors that optimize this margin based on maximizing energy savings.

As a conclusion, there is an abundant bibliography on DC management using proactive actions based on predicting user requests. Many studies have proposed different types of predictions for different horizons. However, to the best of our knowledge, there are no studies evaluating the upper bound of the energy cost savings that can be achieved using this technique. And this is the main contribution of this paper.

## III. Framework for DC modeling

### A. Concepts

The following concepts are used in the paper:

- *Job*: Each job consists of one or more tasks which are defined by the users. It enters the DC and leaves it after its execution.
- *Task*: Each task is run on a single physical machine, after having requested resources.
- *Resource Request*: In this paper, two types of resources are considered: CPU and memory. A resource request is composed of 1) the resource requested (i.e. CPU or memory), 2) the amount of each resource requested.
- *Job Scheduling*: After having requested resources, a task waits to be scheduled. The scheduler takes into account the resources requested by the task and their availability on the machine selected to run this task.
- *Job Execution*: Once scheduled, the job runs until completion, which can be normal (i.e. success) or abnormal (i.e. aborted by the user or the DC). Then, it leaves the DC.

### B. Notations and assumptions

Notations are summarized in Table I.

For the energy cost saving evaluation, the following assumptions are used:

- Assumptions with regard to jobs: Job requests have an additive deterministic component and a stochastic component. As regards the stochastic component, the following assumptions hold:
  - $A_0$ Job arrivals are random.
  - $A_1$ Amounts of requested resources are random.
  - $A_2$ Request durations are random.
- Assumptions with regard to energy:
  - $A_{11}$ When a job is waiting to be scheduled, it does not consume any energy.
  - $A_{12}$ CPU and memory resources are allocated for the job execution duration.
  - $A_{13}$ All machines belonging to the same configuration have the same CPU and memory capacities, as well as the same instruction execution time and energy consumption parameters.
- Assumptions with regard to machines and DC management:
  - $A_{21}$ DC management is in charge of maintaining the number of machines on in any configuration $c$ proportional to the amount of resources used or predicted on machines of this configuration. This is done by turning the necessary machines on or off according to the workload.

TABLE I
NOTATIONS.

| | Name | Meaning |
|---|---|---|
| DC mgt | $T$ (second) | DC management period also called sampling interval |
| Resource request $r(t_i, r_i, \tau_i)$ | $t_i$ | arrival time of the $i^{th}$ resource request |
| | $r_i$ | amount of resources requested by the $i^{th}$ resource request |
| | $\tau_i$ | duration for the $i^{th}$ resource request |
| Consumed Energy (Joule) | $y(k)$ | Energy consumed by the DC in the time interval $(k-1)T \leq t < kT$, computed at time $kT$, with $k \geq 1$ |
| | $y_c(k)$ | Same as $y(k)$, but applied to machine configuration $c$ only |
| Predicted energy (Joule) | $\hat{y}(k)$ | Prediction done one-step ahead at time $(k-1)T$ of $y(k)$ |
| | $\hat{y}_c(k)$ | Same as $\hat{y}(k)$, but applied to machine configuration $c$ only |
| Prediction error (Joule) | $e(k)$ | Error of prediction $\hat{y}(k)$ done for step $k$, known at time $kT$, $e(k) = y(k) - \hat{y}(k)$ |
| | $e_c(k)$ | Same as $e(k)$, but applied to machine configuration $c$ only |
| Machines | $c$ | a machine configuration |
| | $m_c(t)$ | number of machines with configuration $c$ that are on at time $t$ |
| | $r_c(t)$ | Amount of resources used on machines of configuration $c$ and not yet released at time $t \geq 0$ |
| Power (Watt) | $P(t)$ | Power consumed by the DC at time $t$ |
| | $P_c(t)$ | Power consumed by configuration $c$ at time $t$ |
| | $P_{idle,c}$ | Power consumed at null load by a machine of configuration $c$ |
| | $P_{peak,c}$ | Power consumed at 100% load by a machine of configuration $c$ |
| | $\eta_c$ | Power variation coefficient as a function of load for a machine of configuration $c$ |
| (dimensionless) | $\mu_c$ | Coefficient relating the number of machines of configuration $c$ to the resources used by configuration $c$ |
| Energy cost (dollar) | $c_p$ | Proactive cost of one Joule |
| | $c_r$ | Reactive cost of an underestimation of one Joule |
| | $J_{\beta,T}$ | Energy cost using proactive management |
| | $L_{\beta,T}$ | Energy cost using only reactive management |
| | $RES_{\beta,T}$ | Relative Energy Cost Saving |
| | $UB_{\beta,T}$ | Upper Bound of $RES_{\beta,T}$ |
| (dimensionless) | $\beta$ | $\beta = 2c_p/c_r - 1$, with $\beta \in [-1, 1]$ |

  - $A_{22}$ DC management is done periodically with period $T$.
  - $A_{23}$ The energy cost for proactively serving a request is less than or equal to the energy cost for serving it reactively, i.e. $c_p \leq c_r$.

These assumptions are valid in most DCs, and in particular in the Google data center whose data set is used in this paper (see Section IX).

### C. Methodology

Assuming that job requests are expressed as a multiple of resource allocation units, let $y(k)$ be the total amount of energy consumed during time interval $t \in [(k-1)T, kT)$, where $k$ is an integer $\geq 1$ and is called a step. The cost of this energy can be expressed as the sum of two components:

- The first component is the cost of the energy made available at step $k$ according to the prediction computed from past information until step $k-1$. This energy is denoted by $\hat{y}(k)$. Both the prediction and the corresponding proactive actions are done at time $(k-1)T$ for step $k$. Since this energy is available at step $k$, its cost is called a *proactive cost* and it is equal to $c_p\hat{y}(k)$, where $c_p$ is the proactive cost of one Joule.
- The second component is the cost of the energy for serving the prediction error $e(k) = y(k) - \hat{y}(k)$. We distinguish two cases. In the case of energy overestimation, some energy has been wasted and has already been paid for with a cost equal to the proactive cost. In the case of underestimation, a reactive action is carried out at time $kT$. It is important to note that this action is performed a posteriori with a cost greater than or equal to the proactive cost. Its cost is called a *reactive cost* and is equal to $c_r max(0, y(k) - \hat{y}(k))$, where $c_r$ is the reactive cost of an underestimation of one Joule. The reactive cost can include a penalty due to user dissatisfaction because he/she has to wait in the case of energy underestimation. When overestimation occurs (see Fig.1), the penalty is already included in the proactive cost paid with the overestimation and used in the interval. We then have $c_p \leq c_r$. Hence, the reactive action is applied only in the case of underestimation (see Fig.1). Therefore, the prediction error taken into account in the reactive cost is equal to $max(0, e(k)) = max(0, y(k) - \hat{y}(k))$.

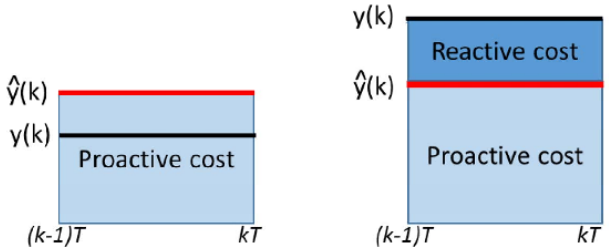


Fig. 1. Costs paid in the case of over-estimation (left part) and under-estimation (right part): in the case of overestimation, only the proactive cost is paid, whereas in the case of underestimation, both the proactive and the reactive costs are paid.

For example, let us assume that at step $k$ the energy actually consumed is $y(k) = n$, whereas the predicted value is $\hat{y}(k) = n + m$. Thus, the total cost for providing energy $y(k)$ is equal to the proactive action cost $c_p\hat{y}(k) = c_p(n + m)$ plus the reactive action cost $c_r|m|$ only if $m < 0$. Fig. 2 shows an example of the energy consumed in steps $k$ and $k + 1$, the predictions, the prediction errors, and the energy costs. The DC management proceeds as follows: the amount of energy needed at step $k$ is 32, whereas the prediction was 27. As an amount of energy of 27 was available at time $(k-1)T$ with proactive cost, an additional amount of 5 must be provided at time $kT$ with a reactive cost. The total cost at step $k$ is $27c_p + 5c_r$. At step $k + 1$, the energy required to serve the new requests is 40. The cost for the 5 additional amounts of energy due to the prediction error for the previous period has already been charged during the previous one. The additional amount of energy assigned proactively to the new period is available at time $kT$ for step $k + 1$. It is given according to the prediction of 42, paid with a proactive cost (even if only

40 instead of 42 are used). No reactive cost is paid in the case of overestimation (i.e. a negative prediction error, here equal to $40 - 42 = -2$).

The total cost assigned to this period is $42c_p$.

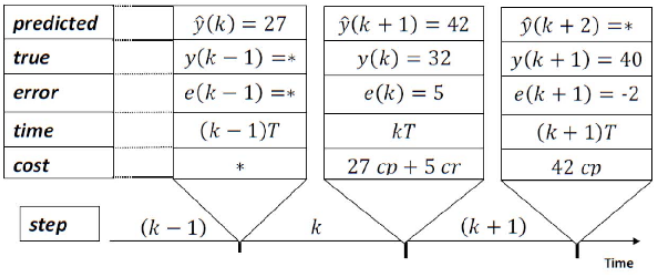


Fig. 2. Example of energy values and cost to serve requests using proactive and reactive actions.

Note that when the energy consumed remains constant from one step to the next, i.e. $y(k-1) = y(k)$, there is no proactive management. In this case, this is equivalent to considering that the prediction is just the last value, i.e. $\hat{y}(k) = y(k-1)$. Thus, in order to evaluate the energy saving with regard to the reactive management when using proactive actions, one must analyze the difference between the total cost achieved at step $k$ when using intelligent predictors and the one obtained by using the last value as a prediction.

Knowing achievable savings by proactive actions is extremely important, since it allows the DC manager to determine which is the most appropriate strategy as well as providing the best management guidelines for the data center. This paper sets out a methodology to analyze the energy cost saving bounds when using predictors.

Since in most DCs [2], machines of different generations and capacities coexist, our evaluation of the upper bound of energy cost saving takes into account machine heterogeneity. This evaluation is carried out in four steps:

- *Step 1: To simplify the modeling process, classify machines into configurations such as all machines of the same configuration have the same features (e.g. memory, CPU, instruction execution time, energy consumption parameters).*
- *Step 2: Compute $y_c(k)$ the energy consumed by machine configuration $c$ in each time interval $[(k-1)T, kT)$ in the recent past.* In this past time interval, we know the arrival time of each request served by machines with configuration $c$, the requested resources and, if finished, we also know its exact duration. Otherwise, we take the duration up to the end of time interval $T$. We then compute $y_c(k)$ the energy consumed by $c$ in the time interval $[(k-1)T, kT)$ as explained in Section IV-A. We proceed similarly for each time interval in the recent past. We iterate on each machine configuration existing in the DC, taking into account machine heterogeneity.
- *Step 3: Compute the Relative Energy cost Saving ($RES$) using proactive decisions, according to the definition given later.*
- *Step 4: Maximize $RES$ using recent past data for predictions $\hat{y}_c$ for each machine configuration $c$. Considering admissible predictors as predictors that give a value of $RES > 0$, compute the upper bound $UB$ for any admissible predictor.* Intuitively, $UB$ is the best $RES$ that can be obtained on any data set by admissible predictors.

We then apply this methodology to the data set [2] collected in an operational DC. For this DC, the cost savings obtained using these predictors are evaluated. More precisely, we:

- *Compute the optimal predictors that maximize the relative cost saving.* Among the optimal predictors, we distinguish linear predictors and nonlinear predictors. Linear predictors are the most well-known ones. Among them, the Auto Regressive Moving Average (ARMA) predictor is the most popular and the most often used with a parameter tuning based on the Recursive Least Squares (RLS) prediction error. It is worth noting that the ARMA model is equivalent to the ARIMA model. We use a procedure called Iterative Reweighted Least Squares (IRLS) together with RLS to minimize $RES$. The nonlinear predictor is based on the minimization of the conditional expected value of $y_c(k)$, given past samples $y_c(k-1), y_c(k-2), \ldots, y_c(k-N)$ for machine configuration $c$.
- *Compare the performance obtained by the optimal predictors with respect to the upper bound.* In this paper, we compare two optimal predictors: the linear optimal one and the nonlinear optimal one.

## IV. Proactive and Reactive Costs

### A. Computation of the energy consumed to serve requests

For the sake of simplicity, we first focus on the single-resource case. All requests concern the same resource which is either CPU or memory. Later in the paper, the result is extended to the case of multiple resources.

Let $r(t_i, r_i, \tau_i)$ denote any request that arrives in the DC at time $t_i$, requiring an amount of resource $r_i$ for a duration $\tau_i$. It can be represented as a rectangular pulse with amplitude $r_i$ of duration $\tau_i$ and zero out of this range, see Fig 3 for an illustrative scheme. The amount of resource consumed by $r(t_i, r_i, \tau_i)$ is equal to:

$$\begin{cases} r_i & \text{if } t_i \leq t \leq t_i + \tau_i \\ 0 & \text{otherwise} \end{cases}$$

The area under the pulse represents the energy consumed to serve the request. For example, in Fig 3, several requests arriving at various times are depicted. Notice that since the user does not know which machine will serve his request, he cannot express the amount of requested resource relatively to the capacity of this machine. Hence, $r_i$ denotes an absolute amount of resource, for example 3 Mbytes of memory.

Most DCs have heterogeneous machines from several generations. Each machine can be characterized by its configuration represented by the pair (CPU, memory) which denotes the CPU and memory capacities of the machine.

The amount of resource used at time $t$ on machines with configuration $c$ in the DC, denoted by $r_c(t)$, is given by the following equation (see also Fig 3 for an illustrative scheme):

$$\boxed{r_c(t) = \sum_i \alpha_c \, r_i} \quad (1)$$

where:

$$\alpha_c = \begin{cases} 1 & \text{if the request } r(t_i, r_i, \tau_i) \text{ is served at time } t \\ & \text{by a machine with configuration c,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Taking into account machine heterogeneity, we express $P_c(t)$ the power consumed at time $t \in [(k-1)T, kT)$ by any configuration $c$ as the sum of the power consumed by the machines with configuration $c$ that are on. For each machine that is on, we adopt the linear power model, which has been extensively used by many authors, [4], [5], [6], [30], where the power consumed by a machine at time $t$ is expressed as the sum of the power consumed at null load and the power proportional to the machine load. Taking into account that all machines of the same configuration have the same CPU and memory capacities, instruction execution time as well as energy consumption parameters (Assumption $A_{13}$), we get:

$$P_c(t) = P_{idle,c} m_c(t) + \eta_c \frac{r_c(t)}{Cap_c}. \tag{3}$$

where $P_{idle,c}$ denotes the power consumption at null load of a machine with configuration $c$; $m_c(t)$ is the number of machines with configuration $c$ that are on at time $t$; $\eta_c$ is its power variation coefficient as a function of the load for any machine with configuration $c$; $Cap_c$ is the capacity of the resource considered on any machine with configuration $c$. It is expressed in absolute; $r_c(t)$ denotes the amount of resources used at time $t$ on machines with configuration $c$.

Since the number of machines with configuration $c$ that are turned-on is proportional to the workload on configuration $c$, according to Assumption $A_{21}$, we can write $m_c(t) = \mu_c \frac{r_c(t)}{Cap_c}$, where $\mu_c$ is the proportionality coefficient relating the number of machines of configuration $c$ to the resource utilization factor. Hence,

$$P_c(t) = (P_{idle,c}\mu_c + \eta_c)\frac{r_c(t)}{Cap_c} \tag{4}$$

The energy consumed by configuration $c$ to serve the requests in the time interval $[(k-1)T, kT)$ is:

$$y_c(k) = \int_{(k-1)T}^{kT} P_c(t)dt \tag{5}$$

$$\boxed{y_c(k) = \left(P_{idle,c}\mu_c + \eta_c\right) \int_{(k-1)T}^{kT} \frac{r_c(t)}{Cap_c} dt.} \tag{6}$$

The total energy consumed by the DC in the time interval $[(k-1)T, kT)$ is expressed as:

$$\boxed{y(k) = \sum_{c \in Conf} y_c(k)} \tag{7}$$

where $Conf$ is the set of configurations in the DC.

### B. Extension to the multi-resource case

The results obtained so far apply to the scalar case of single-resource requests, such as either CPU or memory. The energy consumed given by Equation 6 is the sum of two terms: the energy consumed at null load by the machines that are on, plus the energy consumed by these machines as a function of their workload. This formula can be extended to the multi-resource case, with the following changes:

- The utilization of each resource consumes energy. We assume that this energy varies linearly with the resource utilization factor, as the CPU does. However, the coefficient $\eta$ varies according to the resource considered.
- In addition, the energy consumed by all resources is summed up over all the resources used.
- The number of machines that should be powered on in the interval $[(k-1)T, kT)$ is determined by the resource with the largest utilization factor $\frac{r_{b,c}(t)}{Cap_{b,c}}$ for $t \in [(k-1)T, kT)$. This resource is called the bottleneck resource as in [24] and denoted $b$.

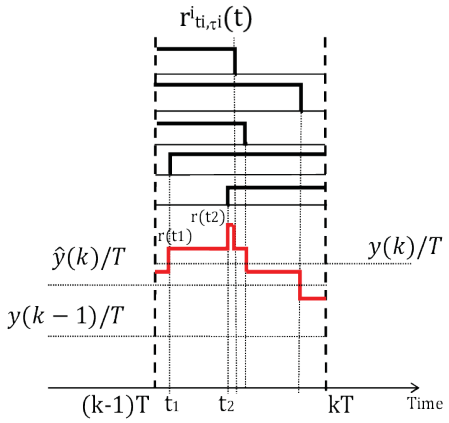


Fig. 3. Requests processed by any given configuration. $r(t_i, r_i, \tau_i)$ = Request arriving at random time $t_i$ with an amount of requested resources $r_i$ for a duration $\tau_i$ in black. $r(t)$ = Total amount of resources used in $[(k-1)T, kT)$, in red. $y(k)$ = Energy consumed in the interval $T$ to serve requests. $\hat{y}(k)$ = Energy predicted, and $y(k-1)$ = last value.

Finally, we get:

$$\boxed{\begin{aligned} y_c(k) = \left(P_{idle,c}\mu_c + \eta_{b,c}\right) & \int_{(k-1)T}^{kT} \frac{r_{b,c}(t)}{Cap_{b,c}} dt \\ + \sum_{u \neq b,\ u \in \mathcal{R}_c} & \eta_{u,c} \int_{(k-1)T}^{kT} \frac{r_{u,c}(t)}{Cap_{u,c}} dt \end{aligned}} \tag{8}$$

where $r_{b,c}(t)$ is the amount of the bottleneck resource $b$ used at time $t$ in configuration $c$; $Cap_{b,c}$ is the capacity of the bottleneck resource $b$ on any machine of configuration $c$; $\mathcal{R}_c$ denotes the set of resources present on machines of configuration $c$; $r_{u,c}(t)$ is the the amount of the resource $u$ used at time $t$ in configuration $c$; $Cap_{u,c}$ is the capacity of the resource $u$ on any machine of configuration $c$.

### C. Computation of the energy cost to serve requests

If the requested resource is available, the request is served immediately. Otherwise, it has to wait until a machine becomes available. If none becomes available in the current time interval, it needs to wait until the next time interval, in which the DC configuration will take into account the underestimation made in the previous time interval.

The cost of the energy consumed by configuration $c$ to serve the requests can be written as the sum of two components.

One is the energy cost of requests served proactively, $c_p\hat{y}_c(k)$, and the other is the cost of the requests served reactively, $c_r(y_c(k)-\hat{y}_c(k))$, when $y_c(k) > \hat{y}_c(k)$. The cost of the energy in the proactive service of requests is less than or equal to the reactive one ($c_p \leq c_r$). For instance, job migrations and server overload can be reduced, and the number of powered-on servers can be optimized. Then, by predicting the energy consumed by configuration $c$ to serve requests, we expect to be able to better manage a DC and effectively reduce the global energy cost. As a consequence, the cost saving is strongly related to the energy prediction accuracy because the energy cost to proactively serve the requests is lower than or equal to that of serving them reactively. Thus, the total cost for configuration $c$ is:

$$\boxed{\mathcal{C}_c(k) = c_p\hat{y}_c(k) + c_r \max(0, y_c(k) - \hat{y}_c(k))} \quad (9)$$

The total cost for the DC is:

$$\boxed{\mathcal{C}(k) = \sum_{c \in Conf} \mathcal{C}_c(k).} \quad (10)$$

As $\max(0, y_c(k) - \hat{y}_c(k))$ can be written $\frac{1}{2}\left(y_c(k) - \hat{y}_c(k) + |y_c(k) - \hat{y}_c(k)|\right)$, and defining $\beta = 2c_p/c_r - 1$ as a linear function of the ratio $c_p/c_r$ representing the relative impact of the proactive and reactive costs, the expected value of the total energy cost for configuration $c$ is:

$$\begin{aligned} C_c &= \mathcal{E}\left[c_p\hat{y}_c + c_r \max(0, y_c - \hat{y}_c)\right] && (11)\\ &= \mathcal{E}\left[c_p\hat{y}_c + \frac{c_r}{2}(y_c - \hat{y}_c) + \frac{c_r}{2}|y_c - \hat{y}_c|\right] && (12)\\ &= \frac{c_r}{2}\left(\beta\mathcal{E}[\hat{y}_c] + \mathcal{E}[y_c] + \mathcal{E}[|y_c - \hat{y}_c|]\right). && (13)\end{aligned}$$

We define $J_{\beta,T,c}$ by $C_c = \frac{c_r}{2}J_{\beta,T,c}$, leading to:

$$\boxed{J_{\beta,T,c} = \beta\mathcal{E}[\hat{y}_c] + \mathcal{E}[y_c] + \mathcal{E}[|y_c - \hat{y}_c|]} \quad (14)$$

$\mathcal{E}[y_c]$ and $\mathcal{E}[\hat{y}_c]$ stand for the expected values of random variables $y_c$ and $\hat{y}_c$, respectively. Since $c_r \geq c_p$, the coefficient $\beta \leq 1$. Since $c_p \geq 0$, $\beta \geq -1$. Therefore, $\beta \in [-1, 1]$.

Note that in the case of perfect prediction, $\hat{y}_c(k) = y_c(k)$, the energy cost of configuration is purely proactive. Then, taking into account that 1) the cost, $J_{\beta,T,c}$, can be reduced by using good predictions and 2) the cost, $L_{\beta,T,c}$, due to the last value prediction, $\hat{y}_c(k) = y_c(k-1)$ has a purely reactive cost, we are interested in evaluating the ratio between both as a useful index of cost reduction when using proactive management. We define the cost reduction index for configuration $c$ by:

$$\frac{\text{mean cost for } c \text{ with proactive management}}{\text{mean cost for } c \text{ without proactive management}} = \frac{J_{\beta,T,c}}{L_{\beta,T,c}} \quad (15)$$

$$\frac{J_{\beta,T,c}}{L_{\beta,T,c}} = \frac{\beta\mathcal{E}[\hat{y}_c] + \mathcal{E}[y_c] + \mathcal{E}[|y_c - \hat{y}_c|]}{(1+\beta)\mathcal{E}[y_c] + \mathcal{E}[|d_c|]} \quad (16)$$

where $d_c(k) = y_c(k) - y_c(k-1)$. Note that in the expression of this cost reduction index, neither $c_p$ nor $c_r$ needs to be known, but only $\beta$.

### D. Computation of the Relative Energy cost Saving

Let us define the Relative Energy cost Saving $RES_{\beta,T,c}$ for configuration $c$ using proactive action as

$$RES_{\beta,T,c} = \frac{L_{\beta,T,c} - J_{\beta,T,c}}{L_{\beta,T,c}}. \quad (17)$$

The greater $RES_{\beta,T,c}$ is, the higher the cost reduction is. By replacing the expressions of $L_{\beta,T,c}$ and $J_{\beta,T,c}$ we obtain:

$$\boxed{RES_{\beta,T,c} = \frac{\mathcal{E}[|d_c|] + \beta\mathcal{E}[e_c] - \mathcal{E}[|e_c|]}{(1+\beta)\mathcal{E}[y_c] + \mathcal{E}[|d_c|]}} \quad (18)$$

where $e_c(k) = y_c(k) - \hat{y}_c(k)$.

The total relative energy saving of the DC, denoted as $RES_{\beta,T}$ is given by:

$$RES_{\beta,T} = \frac{L_{\beta,T} - J_{\beta,T}}{L_{\beta,T}}. \quad (19)$$

Since $L_{\beta,T} = \sum L_{\beta,T,c}$ and $J_{\beta,T} = \sum J_{\beta,T,c}$, for all $c \in Conf$, we get:

$$\boxed{RES_{\beta,T} = \sum_{c \in Conf} \frac{L_{\beta,T,c}}{L_{\beta,T}}\ RES_{\beta,T,c}.} \quad (20)$$

Finally, we obtain:

$$RES_{\beta,T} = \sum_{c \in Conf} \frac{\mathcal{E}[|d_c|] + \beta\mathcal{E}[e_c] - \mathcal{E}[|e_c|]}{(1+\beta)\mathcal{E}(y) + \sum_{c \in Conf}\mathcal{E}(|d_c|)}. \quad (21)$$

### E. Optimal predictor for configuration $c$

An optimal predictor for configuration $c$ is a predictor that maximizes the Relative Energy cost Saving $RES_{\beta,T,c}$, or equivalently minimizes $J_{\beta,T,c}$ to achieve the best energy cost savings.

$$\boxed{Optimal\ predictor\ for\ config\ c = \arg\min_{\hat{y}_c}\left\{J_{\beta,T,c}\right\}} \quad (22)$$

Note that the optimal predictor for any given configuration $c$ is the one that minimizes the cost $J_{\beta,T,c}$ which is different from the cost minimizing the Mean Square Error $MSE$ as shown below:

$$\begin{aligned} J_{\beta,T,c} &\equiv \beta\mathcal{E}[\hat{y}_c] + \mathcal{E}[y_c] + \mathcal{E}[|y_c - \hat{y}_c|] && (23)\\ &= \beta\mathcal{E}[y_c] - \beta\mathcal{E}[e_c] + \mathcal{E}[y_c] + \mathcal{E}[|e_c|] && (24)\\ &= (1+\beta)\mathcal{E}[y_c] + \mathcal{E}[|e_c|] - \beta\mathcal{E}[e_c] && (25)\end{aligned}$$

Then, since $\mathcal{E}[|e_c|] \geq \beta\mathcal{E}[e_c]$, it follows:

$$\begin{aligned} \arg\min_{\hat{y}_c}\{J_{\beta,T,c}\} &= \arg\min_{\hat{y}_c}\{\mathcal{E}[|e_c|] - \beta\mathcal{E}[e_c]\} && (26)\\ &\neq \arg\min_{\hat{y}_c}\left\{\mathcal{E}[e_c^2]\right\} = MSE && (27)\end{aligned}$$

However, when $e_c = 0$ both coincide.

## V. Upper bound of the Relative Energy Cost Saving

We want to establish the upper bound on the relative energy cost savings that can be obtained, whatever the predictor considered. Notice, however, that this bound depends on the data set considered.

Let $y_c$ be a second order stationary random process [21]. Suppose we wish to predict the value of the sample $y_c(k)$ using a linear combination of infinite past samples $y_c(k-1), y_c(k-2), \ldots$. The optimal one-step-ahead prediction error (i.e. minimizing $\mathcal{E}[e_c^2]$) is an uncorrelated sequence with constant Power Spectrum Density (PSD). The minimum mean-squared error (MMSE) is given by the Szegö-Kolmogorov theorem which states that:

$$MMSE = \mathcal{E}[e_c^2] = exp\Big(\int_0^{2\pi} \ln \Phi(j\omega) d\omega)\Big), \tag{28}$$

where $j$ is the complex number meeting $j^2 = -1$, $\omega$ is the frequency in radians per second and $\Phi(j\omega)$ is the PSD of $y_c(k)$, [27].

The simplest upper bound for configuration $c$ is obtained when the prediction error is null. However, this property is never met by a realistic predictor. That is why we propose a realistic upper bound that can be reached by a predictor. To find this upper bound, we need to consider the maximum of $\beta\mathcal{E}[e_c] - \mathcal{E}[|e_c|]$ in Eq.(18). Since $\mathcal{E}[e_c] \leq \mathcal{E}[|e_c|]$, the maximum is obtained when $e_c(k)$ is the positive constant value for all $k$, see Fig. 4. Since $\mathcal{E}[e_c] = \mathcal{E}[|e_c|] = \mathcal{E}[e_c^2]^{1/2} = \gamma_c$, where $\gamma_c$ is MMSE given by Eq. 28, we get the following value for the upper bound $UB_{\beta,T,c}$ of $RES_{\beta,T,c}$:

$$\boxed{UB_{\beta,T,c} = \frac{\mathcal{E}[|d_c|] - (1-|\beta|)\gamma_c}{(1+\beta)\mathcal{E}[y_c] + \mathcal{E}[|d_c|]}} \tag{29}$$

The upper bound at the DC level, denoted as $UB_{\beta,T}$, can be written as the upper bound of $RES_{\beta,T}$:

$$\boxed{UB_{\beta,T} = \sum_{c \in Conf} \frac{L_{\beta,T,c}}{L_{\beta,T}} \; UB_{\beta,T,c}.} \tag{30}$$

Finally, we obtain:

$$UB_{\beta,T} = \sum_{c \in Conf} \frac{\mathcal{E}[|d_c|] - (1-|\beta|)\gamma_c}{(1+\beta)\mathcal{E}(y) + \sum_{c \in Conf} \mathcal{E}(|d_c|)}. \tag{31}$$

In the subsequent sections, we compare linear and nonlinear predictors that maximize $RES_{\beta,T,c}$ on any given configuration $c$.

A numerical comparison is carried out in Section VIII using experimental data collected from a Google DC that allow us to obtain the maximum levels of cost reduction in each case. Thus, the performance of different predictors can be quantified for the DC considered.

## VI. Linear predictors design

We first consider the optimal linear predictor that minimizes the cost $J_{\beta,T}$, given by Eq.14. The expected value is approximated using $N$ sufficiently large past sampled data as follows:

$$\mathcal{E}\big[\beta\hat{y} + |y - \hat{y}|\big] \approx \frac{1}{N}\sum_{i=1}^{N}\big[\beta\hat{y}(i) + |y(i) - \hat{y}(i)|\big] = J_{L_1} \tag{32}$$

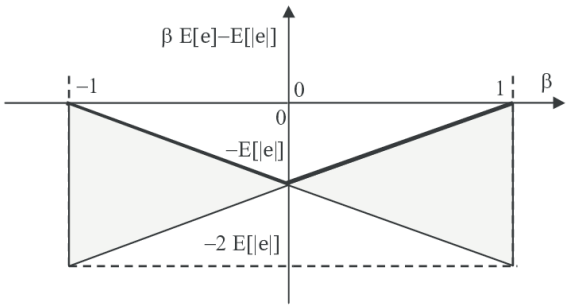


Fig. 4. Admissible values of $\beta\mathcal{E}[e] - \mathcal{E}[|e|]$ in the shaded area.

Since $\hat{y} > 0$ we can write the cost above using vector-norm notation,

$$J_{L_1} = \|R^{1/2}\hat{\mathbf{y}}\|_1 + \|Q^{1/2}(\mathbf{y} - \hat{\mathbf{y}})\|_1 \tag{33}$$

where $\|\mathbf{y}\|_p$ is the $L_p$-norm of vectors defined as follows:

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{N} |x(i)|^p\right)^{1/p}, \tag{34}$$

where $x(i)$ denotes the $i^{th}$ component of vector $\mathbf{x}$. $Q = \beta$ and $R = 1$, but in general they can be positive definite diagonal matrices that will be redefined later. The measured and predicted vectors are given by:

$$\mathbf{y} = \begin{pmatrix} y(k) \\ y(k-1) \\ \vdots \\ y(k-N) \end{pmatrix}; \quad \hat{\mathbf{y}} = \begin{pmatrix} \hat{y}(k) \\ \hat{y}(k-1) \\ \vdots \\ \hat{y}(k-N) \end{pmatrix}; \tag{35}$$

We are interested in obtaining the coefficients of the following $ARMA(na, nc)$ predictor, where $na$ and $nc$ denote the orders of the past values of $y(k)$ and $e(k)$, respectively:

$$\hat{y}(k) = \varphi(k)\theta \tag{36}$$

where $\theta$ is the vector of parameters to be found such that the prediction error is:

$$y(k) - \hat{y}(k) = e(k). \tag{37}$$

The regressor $\varphi(k)$ and the parameter vector $\theta$ have the following structures:

$$\varphi(k) = \begin{pmatrix} y(k-1) \\ \vdots \\ y(k-na) \\ e(k-1) \\ \vdots \\ e(k-nc) \end{pmatrix}^T; \quad \theta = \begin{pmatrix} a_1 \\ \vdots \\ a_{na} \\ c_1 \\ \vdots \\ c_{nc} \end{pmatrix}. \tag{38}$$

where $T$ means transposed. Both the prediction and the error vectors are given by $\hat{\mathbf{y}} = \Phi\theta$ and $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$, respectively, where

$$\Phi = \begin{pmatrix} \varphi(k) \\ \varphi(k-1) \\ \vdots \\ \varphi(k-N) \end{pmatrix}; \quad \mathbf{e} = \begin{pmatrix} e(k) \\ e(k-1) \\ \vdots \\ e(k-N) \end{pmatrix}. \tag{39}$$

The procedure for obtaining optimal $\theta$ that minimizes the quadratic cost $J_{L_2}$ is more often used than the procedure minimizing the cost $J_{L_1}$. The former is obtained using the Ordinary Least Squares (OLS) procedure. The quadratic cost leads to a convex minimization procedure ensuring the uniqueness of the solution with nice convergence properties. The cost $J_{L_2}$ can be written using matrix algebra as follows:

$$\begin{aligned} J_{L_2} &= \|R^{1/2}\hat{\mathbf{y}}\|_2^2 + \|Q^{1/2}(\mathbf{y}-\hat{\mathbf{y}})\|_2^2 \quad (40)\\ &= (\Phi\theta)^T R(\Phi\theta) + (\mathbf{y}-\Phi\theta)^T Q(\mathbf{y}-\Phi\theta)\\ &= \mathbf{y}^T Q\mathbf{y} - 2\mathbf{y}^T Q\Phi\theta + \theta^T\Phi^T(Q+R)\Phi\theta. \quad (41)\end{aligned}$$

For online computation, the optimal parameters $\theta$ that minimizes this cost are obtained by using the well known Recursive Least Squares method (RLS). It is also possible to use the RLS to solve the cost $J_{L_1}$. It consists in choosing appropriate matrices $Q$ and $R$ together with the RLS, and is called Iterative Reweighted Least Squares (IRLS) method, [28]. To this end, the following iteration is proposed:

$$\min_\theta \{J_{L_1}\} = \lim_{j\to\infty} \min_\theta \left\{ \|R_{j-1}^{1/2}\hat{\mathbf{y}}_j\|_2^2 + \|Q_{j-1}^{1/2}(\mathbf{y}-\hat{\mathbf{y}}_j)\|_2^2 \right\} \quad (42)$$

where

$$R_j = \text{diag}\,[\beta/(\hat{y}_j(k)),\ \ldots, \beta/(\hat{y}_j(k-N))], \quad (43)$$

$$Q_j = \text{diag}\,[1/|e_j(k)|,\ \ldots, 1/|e_j(k-N)|], \quad (44)$$

where $\mathbf{e}_j = \mathbf{y} - \hat{\mathbf{y}}_j$ obtained from RLS in the $j^{th}$ iteration. By iterating the procedure, at convergence we obtain $\mathbf{e}_j = \mathbf{e}_{j-1}$ which is the minimum of $J_{L_1}$ we are seeking. Convergence properties of the algorithm for $L_1$ are given in [28].

In the following, we use $J_{L_1}$ to find the optimal linear predictor.

## VII. Nonlinear predictors design

In this approach, instead of being constrained by a linear-predictor, we are interested in finding the optimal predictor $\hat{y}(k) = g(X)$ which is a nonlinear function of previous samples, included in the vector $X = [y(k-1),\ y(k-2),\ \ldots]$, such that the expected value of cost $\mathcal{E}[J]$, defined in (11), is minimum. We rewrite the cost conveniently as:

$$J(y, X) = \beta g(X) + \ |y - g(X)| \quad (45)$$

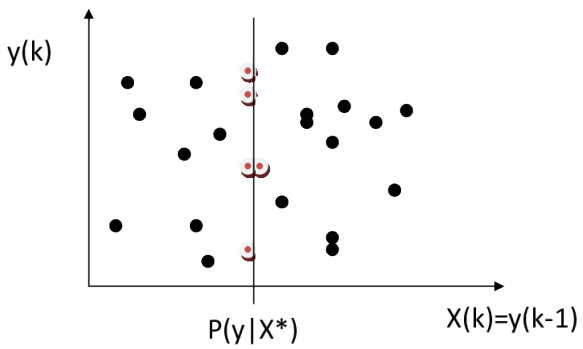


Fig. 5. Sampled $p(y|X)$ from the set of training data.

Taking into account that $\mathbf{y}$ and $\mathbf{X}$ are a stochastic variable and a stochastic vector, respectively, with joint density function $p(y, X)$, the expected value of the cost at each step $k$ can be written in terms of the conditional probabilities as follows, [21]:

$$\begin{aligned}\mathcal{E}[J] &= \int_{X=0}^{\infty}\int_{y=0}^{\infty} J(y,X)\ p(y,X)\ dy\ dX\\ &= \int_{X=0}^{\infty} p(X) \int_{y=0}^{\infty} J(y,X)\ p(y|X)\ dy\ dX,\end{aligned}$$

where $p(y|X)$ is the density of $y$, conditioned to a given $X$ and $p(X)$ is the density of $X$. Since $p(X)$ is always non-negative, therefore it is sufficient to minimize the conditional expectation given by the inner summation. By performing the derivative with respect to $g(X)$ and equating to zero, the following condition for the minimum holds:

$$\int_{y=0}^{\infty} (\beta - \ \text{sign}(y - g(X)))\ p(y|X)\ = 0 \quad (46)$$

where $g(X)$ is considered non negative for all $X$. Given a vector $X^* = [y^*(k-1), y^*(k-2), \ldots, y^*(k-n)]$, where $n$ is the number of previous samples considered, the conditional distribution $p(y|X^*)$ is obtained by finding in a set of training samples all the vectors $X = [y(k-1), y(k-2), \ldots, y(k-N)]$ that match $X^*$ and then forming a subset with the corresponding value $y(k)$ for each. We obtain the subset as shown in a simple example in Figure 5, where $X = y(k-1)$. Using the selected subset of $y$, the optimal predictor, $g(X)$, that fulfills (46) is:

$$\begin{aligned}&\beta \int_{y=0}^{\infty} p(y|X)dy = \int_{y=0}^{\infty} \text{sign}(y - g(X))\ p(y|X)dy\\ \Rightarrow\ &\beta = -\int_{y=0}^{g(X)} p(y|X)\ dy + \int_{y=g(X)}^{\infty} p(y|X)\ dy\\ &\beta = 1 - 2\int_{y=0}^{g(X)} p(y|X)\ dy.\end{aligned}$$

Given $N$ samples of $y$, with conditioned density $p(y|X^*)$, ordered from the smallest to the largest, the optimal nonlinear predictor $g(X^*)$ is the sample at position $N(1-\beta)/2$.

The nonlinear predictor is based on the estimation of the conditional probability density $p(y|X)$. The values of $y$ are associated to vectors $X$ which are obtained by selecting them in a previously stored database. As this process consumes calculation time, it is preferable to associate each value of $y$ to previously defined groups of vectors $X$. These groups are formed by vectors that are close to each other, for example, the vectors whose Euclidean norm is less than a certain pre-established value. This value will depend on the amount of data in the database. Thus, if the database has a large amount of available data, the smaller the bins will be and the greater will be the precision in determining $p(y|X)$.

## VIII. Application to a real DC

### A. Choice of possible values for $T$

The parameter $T$ is the DC reconfiguration period. It is the length of the time interval upon which the prediction is

made. It is chosen by the manager of the DC. Many approaches which use proactive actions, for instance [25] and [26], define such a time interval, as seen in the state-of-the-art presented in Section II. In our analysis we are interested in evaluating how this reconfiguration time affects the energy cost savings when proactive actions are used.

The value of $T$ is a trade-off: small values of $T$ make the DC management more adaptive to workload changes, whereas large values of $T$ decrease the management overhead. However, switching a machine off and on has a cost. Hence, the minimum value of $T$, denoted $Tmin$ can be obtained as in [26] by considering the smallest time interval for which switching off a machine and then switching it on in the next time interval has a null energy cost. In other words, the energy cost saving obtained by switching off the machine is equal to the switching costs (i.e. one switching off followed by one switching-on) for $T = Tmin$.

Too large a value of $T$ is not desirable either, as the delay to serve the requests may become far too large. If the predicted value is underestimated, some users might have to wait for too long until the next configuration time. If the predicted value is overestimated, extra energy is consumed for too long. Hence, the maximum value of $T$, denoted $Tmax$, is determined by the maximum waiting time acceptable by users.

### B. Choice of possible values for $c_p$, $c_r$ and $\beta$

The total energy cost arises from the management of the DC, and it is equal to the sum of the energy cost associated with each configuration. We recall that a configuration groups all machines in the DC having the same features (i.e. CPU, memory, instruction execution time, energy consumption parameters). We assume that the energy cost associated with each configuration in the DC is known by means of a detailed billing by the energy provider. Fig 2 in Section I depicts an example for a given machine configuration.

The energy cost of any configuration $c$ at step $k$ is equal to $C_c(k)$, and it has two parts. The proactive part of the cost is $c_p\hat{y}_c(k)$ and the reactive part is $c_r \max\{0, y_c(k) - \hat{y}_c(k)\}$, where $c_p$ and $c_r$ are coefficients $\geq 0$. The maximum is considered since in the case of overestimation there is no reactive cost. The energy cost of $c$ at step $k$ is the sum of both components. We then have $C_c(k) = c_p\hat{y}_c(k) + c_r \max\{0, y_c(k) - \hat{y}_c(k)\}$. Note that there is always a proactive cost, no matter which predictor is used. For example, for a purely reactive action the proactive cost is $c_p y_c(k-1)$. Since $e_c(k) = y_c(k) - \hat{y}_c(k)$, the total energy cost $C$ is:

$$C = c_p \sum_{c \in Conf} \mathcal{E}[\hat{y}_c] + \frac{c_r}{2} \sum_{c \in Conf} (\mathcal{E}[|e_c| + \mathcal{E}[e_c]) \qquad (47)$$

The value of the coefficients $c_p$ and $c_r$ can be obtained from historical records, since the true and estimated energies, and also the total energy cost in each time interval $T$ are known. For $N$ consecutive time intervals in the recent past, the following formulation holds:

$$\begin{bmatrix} C(1) \\ \vdots \\ C(N) \end{bmatrix} = c_p \begin{bmatrix} \hat{y}(1) \\ \vdots \\ \hat{y}(N) \end{bmatrix} + \frac{c_r}{2} \begin{bmatrix} |e(1)| - e(1) \\ \vdots \\ |e(N)| - e(N) \end{bmatrix} \qquad (48)$$

where $C(k)$ denotes the total energy cost for time interval $[(k-1)T, kT)$.

In this system of equations, the only unknowns are the coefficients $c_p$ and $c_r$. Then, the coefficients are computed using the least squares method.

The coefficients $c_p$ and $c_r$ should be chosen appropriately according to the case.

*1) Energy saving based on proactive strategy:* In this case, the energy cost per Joule (i.e. price in US dollars for one Joule) has the same value for both the prediction and the error. Coefficients $c_p$ and $c_r$ are equal or equivalently $\beta = 2c_p/c_r - 1 = 1$. The total energy cost is given by:

$$C = \frac{1}{2} \sum_{c \in Conf} (\mathcal{E}(\hat{y_c}) + \mathcal{E}(|e_c|) + \mathcal{E}(e_c)) \qquad (49)$$

*2) Quality-of-service-aware energy saving:* In the previous case, the only way that user satisfaction is taken into account is by means of the maximum acceptable value of $T$. Here, the energy cost is artificially increased by some penalties that represent the cost induced by user dissatisfaction caused by long waiting times.

Since the delay in serving the jobs is increased by the prediction error in the case of underestimation, it is reasonable to represent a better quality of service by considering $c_r > c_p$. For example, jobs with short scheduling delays are charged more expensively than the delayed ones. $\beta$ can be considered as a DC management parameter that can be tuned to maximize profit by using proactive actions.

Moreover, value $c_p/c_r$ (or $\beta$) can be different during night, day, priority, etc. This can be taken into account by the DC management to minimize the energy cost.

## IX. Application to the Google Data Set

The results reported in this paper have been obtained using the Google data set described in [2] and collected in an operational data center over a period of 29 days.

### A. Machine configurations

According to our methodology, we first classify machines into configurations, where machines of the same configuration have the same features. This classification is given by Table II. Notice that the CPU capacity is normalized with regard to the most powerful CPU, whereas the memory capacity is normalized with regard to the largest memory.

TABLE II
THE MACHINE CONFIGURATIONS IN THE GOOGLE DC.

| Config | CPU capacity | Memory capacity | Percentage of machines |
|---|---|---|---|
| **1** | **0.5** | 0.5 | **53.45%** |
| **2** | **0.5** | 0.25 | **30.73%** |
| **3** | **0.5** | 0.75 | **7.97%** |
| 4 | 1 | 1 | 6.31% |
| 5 | 0.25 | 0.25 | 0.98% |
| **6** | **0.5** | 0.125 | **0.43%** |
| **7** | **0.5** | 0.03 | **0.039%** |
| **8** | **0.5** | 1 | **0.031%** |
| 9 | 1 | 0.5 | 0.023% |
| **10** | **0.5** | 0.06 | **0.007%** |

Notice that the first six configurations correspond to more than 99.87% of machines in the DC. Furthermore, in this Section, we focus on a single resource: the CPU. Table II shows that 92.65% of machines have the same CPU capacity of 0.5 (see the configurations in bold). That is why in the following, we consider a single configuration with a CPU capacity of 0.5.

### B. Energy parameters used

The energy parameters used for the Google DC are those given in [26], where the power consumed by a machine in configuration $c$ increases linearly from $P_{idle,c} = 300$ W to $P_{peak,c} = 600$ W, when its workload increases from 0% to 100%. The linear increase with the workload has already been observed in [30]. We define $E_{o,c,T}$ as the energy consumed by any machine of configuration $c$ at its peak load during $T$. We then have:

$$E_{o,c,T} = P_{peak,c} T Cap_c \tag{50}$$

By applying Eq. 6 to a single machine of configuration $c$ with a workload equal to its capacity during $T$ and consuming a power $P_{peak,c}$, we deduce:

$$P_{peak,c} = \frac{P_{idle,c}\mu_c + \eta_c}{Cap_c} \tag{51}$$

$$\boxed{y_c(k) = P_{peak,c} \int_{(k-1)T}^{kT} r_c(t)dt.} \tag{52}$$

### C. Acceptable range for T

From the Google data set, we identified that an acceptable value for $T$ is in the range $Tmin = 60$ seconds to $Tmax = 3600$ seconds (i.e. 1 hour). The exact computation of $Tmin$ done in [26] gives 47 seconds. As [26], a conservative value of $Tmin = 60$ seconds is taken for this study. For the maximum value, we take $Tmax = 3600s$, which is a great value of $T$, if we want to minimize user dissatisfaction. However, such a value is useful to study the evolution of the relative energy cost savings when $T$ increases.

### D. Computation of the energy consumed in the past time intervals and its prediction

We compute the energy consumed in the past time intervals of length $T$, as explained in Section IV-A.

We apply the proposed methodology to evaluate the improvements of using proactive management compared to reactive management. Our methodology allows us to know what the benefits would be of using the proposed predictors as well as the upper bound of these benefits.

We select a linear predictor represented by an ARMA model. Two parameters, $na$ the order of the AutoRegressive model and $nc$ the order of the Moving Average model, have to be identified. Starting with $na = 1$ and $nc = 0$, the $F$-test [29] based on the Mean Squared Error (MSE) is applied to decide whether the order should be increased. The recursive algorithm used by the linear predictor works on a sliding window of $na$ past samples. We also select a nonlinear predictor based on the conditional probability. For the nonlinear predictor, the prediction for step $k+1$, done at the end of step $k$, uses the entire set of past samples.

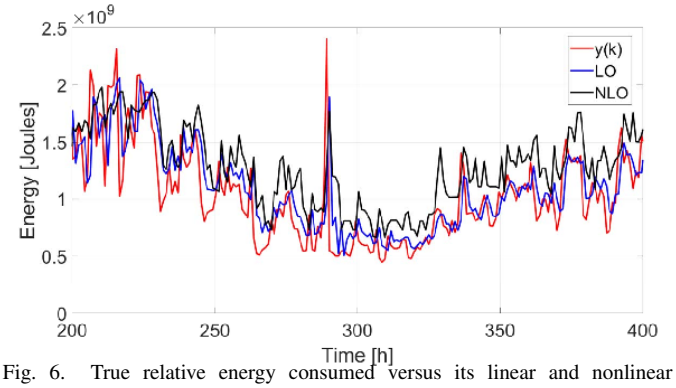


Fig. 6. True relative energy consumed versus its linear and nonlinear predictions for $T = 3600$s and $\beta = -0.8$.

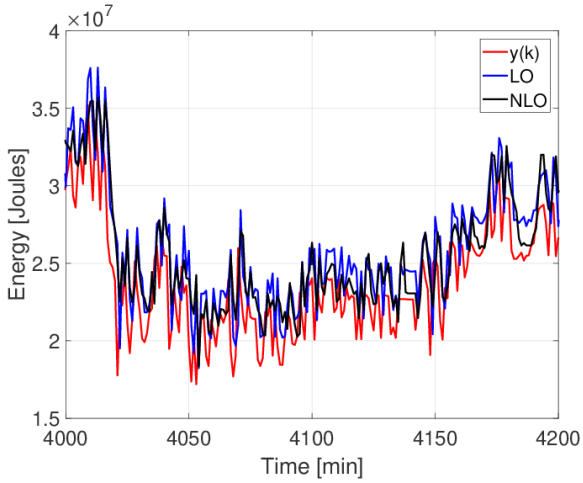


Fig. 7. True relative energy consumed versus its linear and nonlinear predictions for $T = 60$s and $\beta = -0.8$.

Fig. 6 and Fig. 7 depict the relative energy consumed by the DC, its prediction by the Linear Optimal predictor and the NonLinear Optimal predictor for $T = 3600$ s and $T = 60$ s. Notice that the choice of the optimal predictor depends on the value of $\beta$.

For $T = 3600$ s, the nonlinear predictor, which is the best predictor in this case, tends to estimate a value higher than the true energy to avoid reactive cost. This is similar to the safety margin of [26]. However, in our case, this overestimation is optimized to minimize the resulting cost.

For $T = 60$ s, both predictors behave similarly.

### E. Evaluation of the upper bound

The evaluation of the upper bound $UB$ is performed according to Equation 29, for different sets of admissible $\beta \in [-1, 1]$ and $T \in [60, 3600]$ seconds, using the energy needed to serve the CPU requests. The results are depicted in Figure 8. The possible savings are between 1% and 85% depending on the values of $\beta$ and $T$. It can be seen that the highest values are found for decreasing values of $\beta$ and increasing values of $T$. This is due to the fact that when $\beta \leq -0.5$, the proactive cost becomes very small compared to the reactive cost; hence, the upper bound increases considerably. For $\beta = 0$, we can observe in the upper bound of $RES_{\beta,T}$ (see Eq. 29) that the

prediction error $e(k)$ is similar to the difference $d(k)$, therefore the relative energy cost saving $RES_{\beta,T}$ is low in the case of the Google data set studied. However, for $\beta \neq 0$, it is possible to obtain benefits by using biased predictors, as shown by the term $\beta\mathcal{E}[e]$ in Eq. (18).

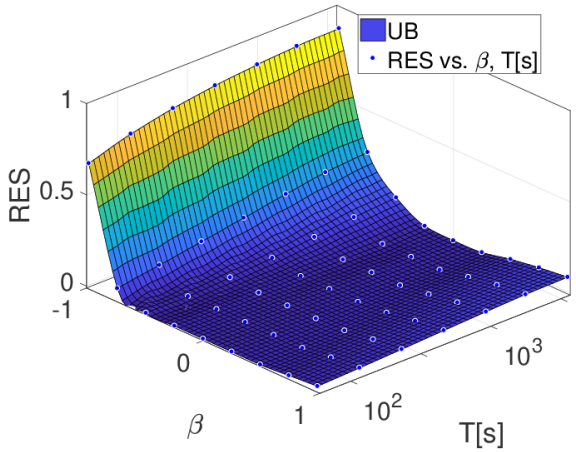


Fig. 8. Upper bound in the interval $\beta = [-1, 1]$ and $T \in [60, 3600]$ seconds.

### F. Computation of the relative energy cost saving

The evaluation of $RES_{\beta,T}$ is performed for different sets of admissible $\beta$ and $T$ using the energy needed to serve the CPU requests, leading to the results depicted in Fig. 9 for the Linear Optimal predictor and Fig. 10 for the NonLinear Optimal predictors. With both predictors, energy cost savings are obtained only for values of $\beta \leq -0.5$. Notice also that the NonLinear Optimal predictor gets much closer to the upper bound than the Linear Optimal predictor.

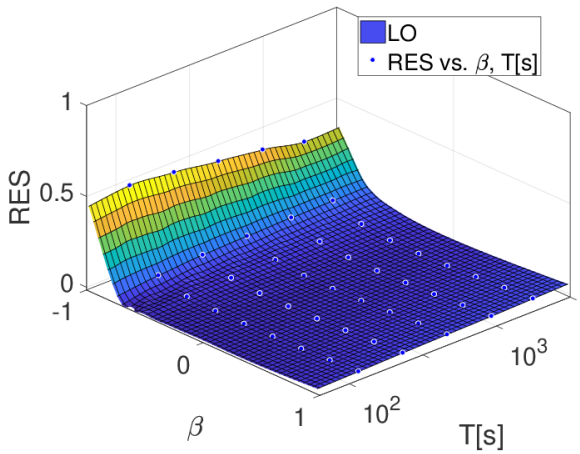


Fig. 9. RES for CPU requests using linear $LO$ optimal predictor in the interval $\beta = [-1, 1]$ and $T \in [60, 3600]$ seconds.

Small values of $T \in [60, 3600]$ give lesser values of $RES_{\beta,T}$, but the relative energy cost savings strongly depend on the value of $\beta$. For values of $\beta$ varying from $-0.4$ to $-1$, $RES_{\beta,T}$ increases from 0 to 0.85. Although $RES_{\beta,T}$ increases with larger values of $T$, we consider no value larger than 3600 seconds to minimize user dissatisfaction caused by a long waiting time before job execution in the case of underestimation.

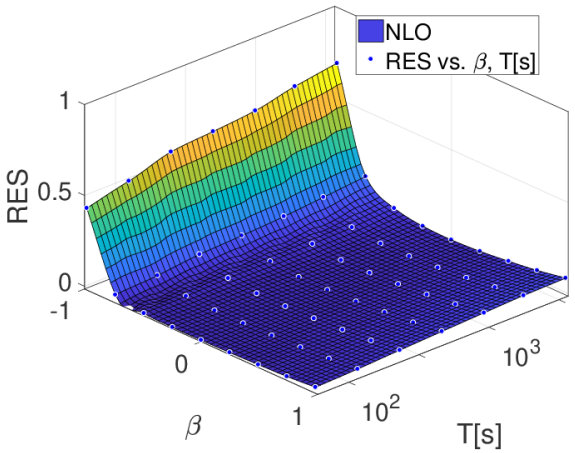


Fig. 10. RES for CPU requests using nonlinear optimal $NLO$ predictor in the interval $\beta = [-1, 1]$ and $T \in [60, 3600]$ seconds.

We first consider the case $T = 60$ s and make $\beta$ vary in the interval $[-1, 1]$. Fig. 11 depicts the performance of the Linear and NonLinear predictors together with the upper bound. Relative energy cost savings are achieved for values of $\beta > 0.5$ or $< -0.5$. For values of $\beta > 0.5$, there is a benefit of less than $5\%$. For values of $\beta < -0.5$, the benefit is higher and increases when $\beta$ is getting closer to $-1$ to reach $58\%$ for the Linear predictor and only $50\%$ for the NonLinear predictor. In both cases, the Linear predictor used performs better than the NonLinear one.

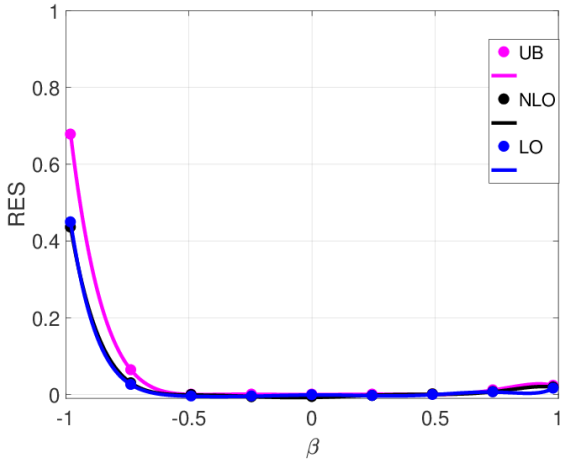


Fig. 11. $RES_{\beta,T}$ for nonlinear optimal $NLO$, linear optimal $LO$ and Upper bound $UB$ in the interval $\beta \in [-1, 1]$ and $T = 60$ seconds.

We now consider the case $T = 3600$ s and $\beta \in [-1, 1]$. The same performance is depicted in Fig. 12. Again, we notice that there is an interval for which there is no benefit over last value prediction: this is when $\beta$ belongs to $[-0.2, 0.2]$. Unlike the previous case, the NonLinear predictor obtains better energy cost savings. For $\beta = 1$, it reaches $8\%$, whereas the Linear predictor we used does not get better than $3\%$. For $\beta = -1$, the NonLinear predictor reaches $73\%$, whereas the Linear predictor reaches only $19\%$.

### G. Conclusion on the Google dataset

This example shows that by applying the methodology proposed, it is possible to improve costs achieved by Google

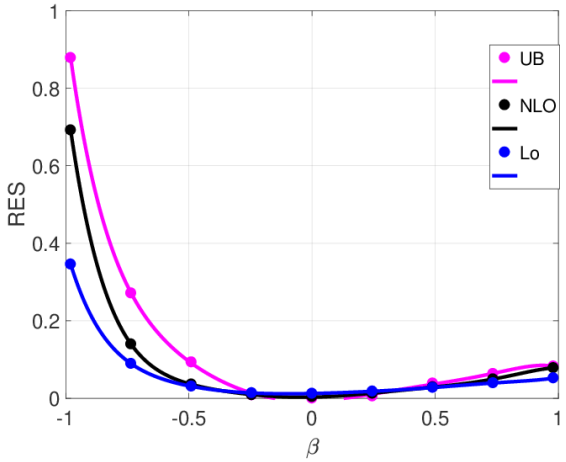


Fig. 12. $RES_{\beta,T}$ for nonlinear optimal $NLO$, linear optimal $LO$ and Upper bound $UB$ in the interval $\beta \in [-1, 1]$ and $T = 3600$ seconds.

by up to 85%, strongly depending on the value of $\beta$. This example highlights the importance of knowing the upper bound, because it helps the DC manager to position the predictor with regard to the upper bound. If the current predictor is near the upper bound, it is suitable for DC management.

## X. Discussion

### A. Impact of $\beta$ and $T$

The previous section showed that the value of $\beta$ has a strong impact on the relative energy cost savings one can get. Our results show that for $\beta \in [-0.2, 0.2]$, there are no energy cost savings with respect to the last value. However, the value of $\beta$ is computed from external data (energy cost, user satisfaction, etc.) and cannot be changed.

In this data set, a strong dependency between $\beta = 2c_p/c_r - 1$ and the benefits of the proactive action has been observed. More generally, there exists a trade-off between energy savings and quality of service (QoS). If one tries to improve the QoS, more energy is spent and vice-versa. In this example, we show that when $T$ increases, the energy cost savings will be higher due to the proactive action, but the QoS gets worse because some jobs will take much longer, and vice versa. If short response times are sought, small values of $T$ are needed. Even though, the value of $T$ has a small impact on the relative energy cost savings, it should be carefully chosen by the DC manager. To summarize, the value chosen for $T$ is obtained by successive refinements. First, it should belong to the interval $[Tmin, Tmax]$, as said in Section IX.C. Second, as previously pointed out, it is a trade-off between energy saving and QoS. Third, for a given value of $\beta$, the value of $T$ chosen by the DC manager maximizes the value of $RES_{\beta,T}$ for the predictor considered. Hence, according to the results of this study (Fig. 9 and Fig 10), this value optimizes the energy saving for the quality of service required.

### B. Dynamic capacity provisioning based on energy prediction

The basic principle of energy saving in a DC lies in turning off machines which are not being used during a time greater than or equal to $Tmin$. More generally, how to provide dynamic capacity provisioning [25] when only the energy that should be consumed in the next time interval is predicted. If the predicted energy is higher than the energy consumed in the previous time interval, some machines may have to be turned on. On the other hand, if the predicted energy is less than the energy consumed in the previous time interval, some machines may have to be turned off. We define $m_c(k)$, the minimum number of machines on in configuration $c$ to consume the energy $\hat{y}_c(k)$. In the single resource case, we have:

$$m_c(k) = \left\lceil \frac{\hat{y}_c(k)}{E_{o,c,T}} \right\rceil. \tag{53}$$

According to Eq. 53, the number of machines that have to be switched on or off for step $k$ is equal to $|m_c(k) - \mathbf{m_c}(k-1)|$, where $\mathbf{m_c}(k-1)$ is the number of machines on in step $k-1$. If $m_c(k) - \mathbf{m_c}(k-1) > 0$, these machines have to be turned on. If $m_c(k) - \mathbf{m_c}(k-1) = 0$, no change is needed. If $\mathbf{m_c}(k-1) - m_c(k) > 0$, these machines have to be turned off.

In the case of multiple resources, the energy consumed by each resource present in configuration $c$ is predicted. More precisely, we predict for each resource the part of the energy consumed which is proportional to the utilization factor of this resource: the second term of the sum in Eq 3. We deduce which resource is the bottleneck resource. We then compute the number of machines needed to provide this energy.

### C. Mitigation of the impact of underestimation

If the prediction leads to an underestimation, user requests have to wait until a machine becomes available. This is a large factor of user dissatisfaction. In this paper, we assume that the corrective action is done at the beginning of the next time interval.

To mitigate the effect of underestimation, Dabbagh et al. [26] propose two solutions. The first one tries to avoid underestimation, by adding a safety margin to the prediction of the number of machines. This solution may lead to overestimation which is not energy efficient. To limit overestimation, the safety margin is dynamically tuned according to the prediction error.

The goal of the second solution is to limit the DC latency to recover from this underestimation. To react more promptly (e.g. every 10s for a time interval T=60s), every T/10 for instance, an underestimation check is done. If some underestimation is detected, some machines are turned on to reduce user waiting time. In our case, the cost minimization performed during the search of the optimal predictor within a given predictor family (e.g. Linear, NonLinear) determines the optimal value to have the greatest energy saving. This is an important difference of our approach compared with [26].

## XI. Conclusion

The purpose of this paper is to evaluate the energy cost savings that can be achieved by a proactive DC management. Such a management consists in periodically configuring the DC by turning some servers on or off to provide an amount of energy equal to the sum of i) the predicted energy to

serve the user requests that will arrive in the next period, and ii) the energy needed to correct the error on the prediction made for the previous period. The predicted energy has a proactive cost, whereas the corrective energy has a reactive cost. Two predictors are proposed: a linear one based on the ARMA model and a nonlinear one using the conditional probability density function. These predictors maximize the relative energy cost saving.

We determined the tight upper bound of the relative energy cost savings valid in all cases. Knowing the upper bound makes it possible to decide on the suitability or not of using proactive action. If a choice must be made between different possible proactive strategies, both costs and savings change. Therefore, this paper proposes a methodology that helps to choose the best strategy (i.e. that maximizes the savings) between possible proactive strategies.

The methodology proposed is generic and can be applied to any DC that meets the assumptions made in this paper. These assumptions are realistic and can be met by many data centers. As an example, we have used the data set collected over 29 days in an operational data center of Google. By applying this methodology on the Google data, an improvement up to 85% can be obtained, leaving room for multiple optimizations.

The relative energy cost saving that depends on the DC considered have been computed. Using the predictors analyzed with this data set, we find that the relative saving of the proactive action increases for $\beta$ near to $-1$. This would make it possible to decide between possible proactive action strategies in this Google DC.

## REFERENCES

[1] A. Patrizio, "10 predictions for the data center and the cloud in 2019", Network World Journal, December 2018. https://www.networkworld.com/article/3324050/10-predictions-for-the-data-center-and-the-cloud-in-2019.html

[2] J. Wilkes, "More Google cluster data", Google research blog, Nov. 2011. http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html

[3] M. Dayarathna, Y. Wen, R. Fan, "Data center energy consumption modeling: a survey", IEEE Communications Surveys & Tutorials, Vol.18 No 1, First Quarter 2016.

[4] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Y. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems", CoRR, 2010.

[5] 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. 4th ACM/SPEC ICPE, 2013, pp. 295-306.

[6] 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.

[7] C. Reiss, J. Wilkes, J. L. Hellerstein, "Google cluster-usage traces: format + schema", Google Inc. Technical Report, Mountain View, CA, Nov 2011. https://github.com/google/cluster-data

[8] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis", ACM Symposium on Cloud Computing (SoCC), San Jose, CA, Oct. 2012,

[9] S. Di, D. Kondo, W. Cirne, "Host Load Prediction in a Google Compute Cloud with a Bayesian Model", Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC'12, Salt Lake City, Utah, IEEE Computer Society Press, 2012.

[10] O. Beaumont, L. Eyraud-Dubois, J.A. Lorenzo-Del-Castillo, "Analyzing real cluster data for formulating allocation algorithms in Cloud platforms", 2nd International Symposium on Computer architecture and High Performance Computing (SBAC-PAD), Paris, France, Oct. 2014.

[11] M. Alam, K. A. Shakil, S. Sethi, "Analysis and Clustering of Workload in Google Cluster Trace based on Resource Usage", Jan. 2015,

[12] P. Minet, E. Renault, I. Khoufi, S. Boumerdassi, "Analyzing traces from a Google data center", 14th International Wireless Communications and Mobile Computing Conference (IWCMC 2018), Limassol, Cyprus, June 2018.

[13] B. Liu, Yinan Lin and Y. Chen, "Quantitative workload analysis and prediction using Google cluster traces," 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), San Francisco, CA, 2016, pp. 935-940.

[14] J. Prevost, K. Nagothu, B. Kelley and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," 2011 6th International Conference on System of Systems Engineering, Albuquerque, NM, 2011, pp. 276-281.

[15] M.S. Yoon, A.E. Kamal, Z. Zhu (2017) "Adaptive data center activation with user request prediction", Computer Networks 122:191-204.

[16] S. Mazumdar and A.S. Kumar, "Forecasting Data Center Resource Usage: An Experimental Comparison with Time-Series Methods", Springer International Publishing AG 2018 A. Abraham et al. (eds.), Proceedings of the Eighth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2016), Advances in Intelligent Systems and Computing 614, DOI 10.1007/978-3-319-60618-7 16.

[17] R. Cao, Z. Yu, T. Marbach, J. Li, G. Wang, X. Liu, "Load Prediction for Data Centers Based on Database Service", 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), Tokyo, Japan, July 2018.

[18] Hastie, T., R. Tibshirani, and J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. (2009) Springer.

[19] H. Kantz, and T. Schreiber (2004). Nonlinear Time Series Analysis. Cambridge, England: Cambridge University Press, 2nd edn.

[20] S. Caires, and J. A. Ferreira (2005). On the Non-parametric Prediction of Conditionally Stationary Sequences. Statistical Inference for Stochastic Processes, 8: 151-184. (2005).

[21] A. Papoulis, Probability, Random Variables and Stochastic Process, McGraw-Hills Book Company, USA, 1965.

[22] A. Wolke, M. Bichler, and T. Setzer. "Planning vs. dynamic control: Resource allocation in corporate clouds". IEEE Transactions on Cloud Computing, 7161(99):1-14, 2014.

[23] C. Delimitrou and C. Kozyrakis. "Quasar: Resource-efficient and QoS-aware Cluster Management". In Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS'14, pages 127-144, 2014.

[24] Q. Zhang, M. Faten Zhani, S. Zhang, Q. Zhu, R. Boutaba, and J. L. Hellerstein. 2012. "Dynamic energy-aware capacity provisioning for cloud computing environments". In Proceedings of the 9th international conference on Autonomic computing (ICAC '12). ACM, New York, NY, USA, 145-154, 2012.

[25] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein. "Dynamic Heterogeneity-Aware Resource Provisioning in the Cloud". IEEE Transactions on Cloud Computing, 2(1):14-28, 2014.

[26] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes. Energy-Efficient Resource Allocation and Provisioning Framework for Cloud Data Centers. IEEE Transactions on Network and Service Management, 12(3):377-391, Sep 2015.

[27] P. J. Brockwell and R. A. Davis, Time Series: Theory and Methods. New York, USA: Springer-Verlag, 1986.

[28] C. S. Burrus, J. A. Barreto, and I. W. Selesnick, "Iterative reweighted least squares design of filters", IEEE Transactions on Signal Processing, 42(11):2926-2936, November 1994.

[29] T. Söderström, P. Stoica. System Identification. Prentica Hall International, 1989.

[30] I. Sarji, C. Ghali, A. Chehab, A. Kayssi, "CloudESE: Energy Efficiency Model for Cloud Computing Environments". International Conference on Energy Aware Computing (ICEAC), 2011.

**Ruben Milocco** is research scientist at the Grupo de Control Automático y Sistemas (GCAyS) and consulting professor at the Department of the Electrical Engineering both of the Universidad Nacional del Comahue (UNCo). He received a Doctoral degree at the Universidad Nacional de La Plata, (UNLP), where he also served as professor. He is member of the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), all from Argentina. His research interests include filtering, estimation and identification with applications to wireless and mobile networks and efficient energy storage management systems.

**Pascale Minet** is a senior researcher at Inria, Paris in the EVA team. She got her qualification in advising PhD students (HDR) in 1998 from the University of Versailles. Previously she got her PhD diploma in Computer Science, in 1982 from the University of Toulouse and her Engineer diploma in Computer Science in 1980 from ENSEEIHT. 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. More recently, she is interested in 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 supervised 15 defended PhD Theses. She is co-author of the OLSR routing protocol standardized at IETF. She published more than 200 papers in international conferences, journals and book chapters.

**Éric Renault** is Full Professor at ESIEE Paris and a member of LIGM (UMR CNRS 8049) at Université Gustave Eiffel, France. He received a MSc in Computer Engineering (diplôme d'ingénieur) from ISTY and a MSc in Computer Science (DEA) from UVSQ in 1995, a PhD in Computer Science from UVSQ in 2000 and the qualification in advising PhD students (HDR) from UPMC in 2011. His research interests include high-performance computing and messaging, compilation, virtualization, positioning and lightweight security for mobile, sensor and vehicular networks. He worked on several European projects and served as an expert for the evaluation of French national projects (ANR), private company research works (MESRI) and Eiffel Excellence Scholarship Program applications (Campus France). He authored more than 100 articles in international journals and conferences.

**Selma Boumerdassi** is an Associate Professor at CNAM, and a Research Associate at Inria Paris. She received a MSc in Computer Engineering (diplôme d'ingénieur) from ESI, Algeria, and a MSc in Computer Science and a PhD in Computer Science from UVSQ, France, where she also served as an Assistant Professor. 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. Selma Boumerdassi 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 authored more than 100 articles and served as a TPC member for various international journals and conferences.