



Assessing and forecasting energy efficiency on Cloud computing platforms



Josep Subirats*, Jordi Guitart

Barcelona Supercomputing Centre (BSC), Jordi Girona 31, 08034 Barcelona, Spain
 Universitat Politècnica de Catalunya (UPC), Jordi Girona 1-3, 08034 Barcelona, Spain

HIGHLIGHTS

- Assess and forecast energy/ecological efficiency for multiple levels in real time.
- Assess and forecast energy/ecological efficiency for potential actions.
- Estimate the future CPU utilisation of a VM.

ARTICLE INFO

Article history:

Received 18 October 2013
 Received in revised form
 10 November 2014
 Accepted 10 November 2014
 Available online 18 November 2014

Keywords:

Cloud computing
 Energy efficiency
 Ecological efficiency
 Forecasting
 Green computing
 IaaS provider

ABSTRACT

IaaS providers have become interested in optimising their infrastructure energy efficiency. To do so, their VM placement algorithms need to know the current and future energy efficiency at different levels (Virtual Machine, node, infrastructure and service levels) and for potential actions such as service deployment or VM deployment, migration or cancellation. This publication provides a mathematical formulation for the previous aspects, as well as the design of a CPU utilisation estimator used to calculate the aforementioned forecasts. The correct adjustment of the estimators' configuration parameters has been proved to lead to considerable precision improvements. When running Web workloads, estimators focused on noise filtering provide the best precision even if they react slowly to changes, whereas reactive predictors are desirable for batch workloads. Furthermore, the precision when running batch workloads partially depends on each execution. Finally, it has been observed that the forecasts precision degradation as such forecasts are performed for a longer time period in the future is smaller when running web workloads.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

Over the last years, Cloud computing has become a consolidated computing paradigm that allows sharing different kinds of resources over the network in an automated way. Infrastructure as a Service (IaaS) providers provide low level computing resources on which the user can run its own software, typically in the form of Virtual Machines (VMs) and Virtual Local Area Networks (VLANs) established between them. The use of VMs allows the creation of services that can ask for more or less resources depending on their instantaneous demand variation (scalability). For instance, by simply adding a new VM replica to an already existing service, it can process the requests of a bigger number of users of that service. Another advantage of using VMs is that they offer the possibility to

be migrated from one physical host to another without stopping its execution, transparently to the user.

Back in 2006, the annual energy consumption of the datacenters in the United States was already of about 61 Billion kWh, which is equivalent to the energy consumed by 5.8 million average US households [1]. From that year, this amount of energy has significantly increased. What is more, given that most of the energy produced in US and around the world comes from burning coal and natural gas [2], the datacenter's energy consumption has a direct impact on climate change [3].

An individual IaaS provider typically owns a very large number of physical hosts, where the different users' VMs are deployed. Altogether, they consume a massive amount of energy which represents about 30% of their datacenter's operating expenses [4]. Moreover, IaaS providers have to pay fines if their carbon footprint exceeds the limits imposed by international regulations such as the Kyoto Protocol [5]. As a consequence, IaaS providers have a great interest in reducing the total power consumption of their physical hosts.

Each server has a power consumption that is not proportional to its resource utilisation. Even though the power consumption

* Corresponding author at: Barcelona Supercomputing Centre (BSC), Jordi Girona 31, 08034 Barcelona, Spain.

E-mail addresses: josep.subirats@bsc.es (J. Subirats), jordi.guitart@bsc.es (J. Guitart).

grows linearly with regards to its utilisation [6], if the host is not used it still has an associated idle power consumption. Using VM migration, multiple VMs can be consolidated into a single physical server and the remaining idle servers can be turned off, avoiding their idle power consumption and therefore reducing the overall datacenter's energy consumption.

Until recently, IaaS providers only cared about providing the required performance to their users, regardless of the incurred energy consumption. Green Cloud computing is envisioned to achieve efficient processing while minimising the energy consumption [7]. Energy efficiency is defined as the ratio between the useful output of a process and the energy input to this process [8]. It is a generic term which can be applied in multiple fields, but when talking about Green Cloud computing, improving the energy efficiency refers to using less energy to produce the same amount of services or useful output. Therefore, the ultimate target of Green Cloud computing is maximising the energy efficiency.

IaaS providers need tools to measure and predict the energy efficiency of their infrastructure in real time (this includes the performance and the power being delivered), as well as the generated carbon emissions. The current available tools are in fact benchmarks which are able to evaluate the energy efficiency of a node or set of nodes in offline mode, but they are not able to evaluate it in real time while executing general applications, or to predict it for a given amount of time in the future. Some publications use real-time metrics affecting the energy efficiency to drive their VM placement algorithms, but not the energy efficiency itself. What is more, IaaS providers use VM placement algorithms which need to know what will be the impact of performing a VM deployment/migration/cancellation on the infrastructure's energy efficiency in order to optimise it. Even though some of these calculations have been addressed by the related literature, they have been only evaluated in simulated environments, but no tools have been provided to IaaS providers to gather such information.

In this context and taking into account the limitations of the current available tools, the present publication tackles the following innovations:

- Develop mechanisms to assess and forecast the energy efficiency and the ecological efficiency of a Cloud IaaS provider's infrastructure in real time. Different levels will be considered: infrastructure level, node level, service level and VM level.
- Develop mechanisms to predict the energy and ecological efficiency of a Cloud IaaS provider infrastructure for potential actions, including: service deployment, VM deployment, VM migration and VM cancellation.
- Develop a mechanism to estimate the future CPU utilisation of a VM.

The present publication is structured as follows. Section 2 provides the necessary background and exposes the related work. Section 3 contains all the mathematical models developed to evaluate and forecast the energy efficiency at different levels for the current status of the provider, as well as to forecast it for potential actions. In Section 4, the description of the CPU estimator indirectly used when predicting the energy efficiency can be found. The developed tool is evaluated in Section 5 using two kinds of realistic workload: web and batch. The configuration parameters of the CPU estimator are also tuned in order to obtain a good precision when forecasting the energy efficiency of the infrastructure. Finally, in Section 6, the conclusions drawn from this work and future research lines are presented.

2. Background and state of art

2.1. Background

Along with the outburst of Cloud computing came a surge of interest in Green Computing. The latter is a field where the engi-

neers and developers try to optimise the energy efficiency of various computing processes either by intervening into the hardware or the software architecture.

A tool to measure the energy and ecological efficiency (see Section 3) has been developed as a component called the Ecoefficiency Tool, as part of the OPTIMIS project [9] (Optimized Infrastructure Services, FP7-ICT-2009-5 program). A preliminary version was introduced in [10] and has been extended and improved in this publication. If a service (a set of VMs performing a collaborative task) is accepted by an IaaS provider, an OPTIMIS component called Virtual Machine Manager (VMM) evaluates what is the best node to deploy each of the VMs of the service, considering the Business Level Objective (BLO) of the provider. If the BLO seeks to maximise the energy/ecological efficiency of the infrastructure, the VMM will choose the destination node that maximises the infrastructure's energy/ecological efficiency if the VM is deployed in it. When the node to host a particular VM has been chosen, the VMM issues the VM creation by sending its VM description to the virtualisation middleware.

The Ecoefficiency Tool continuously monitors the current and future energy/ecological efficiency of the infrastructure, and notifies higher-level components if a potential threshold violation is expected to happen, so corrective actions (such as VM migration) can be taken in advance. The performed predictions need to be as accurate as possible in order to avoid unnecessary and incorrect notifications. This is the reason why the conducted experimentation has been focused on trying to maximise the precision of the tool when performing these forecasts. Additional actions like VM deployment or cancellation can also be considered for service scalability. Its impact in the infrastructure's energy/ecological efficiency can be calculated using the methods described in Section 3.2.

The Ecoefficiency Tool calculates/predicts the energy/ecological efficiency using three kinds of data types: performance, CPU utilisation and power data. We have adopted the MWIPS metric to measure the performance, which refers to the Millions of Whetstone Instructions per Second performed with respect to those performed by a SPARCstation 20-61 with 128 MB RAM, a SPARC Storage Array, and Solaris 2.3. The MWIPS of each physical node have been measured using the UnixBench benchmark [11]. The CPU utilisation data is captured in real time using the *virt-top* utility [12]. The power data is obtained in real time from a power sensor using the SNMP protocol.

The PUE or Power Usage Effectiveness is defined in [13] as the ratio between the total power consumed in a datacenter (P_{DC}) and the power delivered to the computing infrastructure (P_{IT}). As exposed in [14], P_{IT} actually depends, among others, on the temperature of the cold air supplied by the air conditioning to the infrastructure room (T_5), which also influences the PUE. For a fixed T_5 , the total power consumption of the datacenter grows linearly by a factor of PUE with regards to the IT power consumption P_{IT} . Even though T_5 could be modified and thus the PUE would vary, our assessments and forecasts are made for much smaller periods than the ones in which T_5 varies. In addition, as exposed in [13], different datacenters normally have different PUE values. However, we address the calculation of the energy efficiency at different levels for a single datacenter of an IaaS provider. For these reasons, the PUE can be considered as a constant in all the formulas.

The mathematical models described in this paper will be validated with different workload types. In particular, we differentiate between interactive and batch workloads:

- Interactive workloads have a variable amount of work to perform, which consists of a number of client transactions to process in a given instant. Therefore, resource utilisation for interactive workloads is highly variable over time, as it depends on the number of simultaneous clients interacting with the sys-

tem and on the particular interaction that each client performs. In our experimentation, we have used an interactive workload emulating the arrival of client's requests at a Web server over several days.

- Batch workloads consist of a collection of tasks that are submitted to a task scheduler, that decides when/where to execute them and keeps them in a queue in the meanwhile. These tasks have a predefined amount of work to be performed, which is known since the beginning of their execution. Once this work has been processed, the task ends. Normally, a task is processed as fast as the machine executing it can cope with. For computing-intensive tasks such as the ones from HPC, the utilisation of each CPU executing a task tends to be around 100% during its execution. So, even though the global resource utilisation will vary over time (as new tasks begin their execution and old tasks finish), the utilisation of the CPUs executing a particular task will be quite stable during its execution. In our system validation, we have used an HPC batch workload executing computing-intensive tasks whose arrival times follow a Poisson distribution.

As part of the experimentation, we needed a way to compare the precision of different CPU utilisation predictors under several configuration parameter values. We also needed a way to evaluate the degree of agreement between the performed infrastructure energy efficiency forecasts for a given time and the assessment value at that same time, to see if the previously made forecasts had been accurate. Several metrics were initially considered to this end: the Mean Absolute Error (or MAE), the Coefficient of Determination (or r^2) and the Concordance Correlation Coefficient (or CCC).

The MAE can be used to measure the average numerical closeness between assessments and forecasts. The MAE between two vectors of length N can be calculated as in Eq. (1). However, as discussed in [15], these measures are not standardised nor bounded and depend on data unit and scale. As it will be seen, we have compared the precision of the predictors with two different workloads that cause different resource utilisation levels. A difference of 5% of CPU utilisation on workloads whose mean value is around 100% would be much more relevant if the mean value was around 50%. So, as we need a metric that is comparable across workloads with different utilisation levels, the MAE was discarded.

$$\text{MAE} = \frac{1}{N} \cdot \sum_{n=1}^N |x_n - y_n| = \frac{1}{N} \cdot \sum_{n=1}^N |e_n|. \quad (1)$$

The r^2 measures the proportion of the total variation of variable y explained by variable x (see Eq. (2)). In this case, r^2 is a bounded metric between 0 and 1 (a value of 1 meaning perfect correlation), and would be appropriate to compare between workloads with different utilisation levels, contrarily to MAE. However, also as exposed in [15], it is not considered appropriate to measure data agreement, since it fails to measure the actual difference between the two datasets. For instance, datasets $A = \{1,2,3\}$ and $B = \{1,2,3\}$ would have a r^2 of 1, but also datasets A and $C = \{110,210,310\}$ (being $C = 100 \cdot A + 10$). We need to use a metric that actually measures how identical two datasets are, in terms of agreement and difference between the predictions and the real values. So, r^2 was also discarded.

$$r^2 = \frac{\sum_{n=1}^N (x_n - \bar{x})(y_n - \bar{y})}{\sqrt{\sum_{n=1}^N (x_n - \bar{x})^2 \sum_{n=1}^N (y_n - \bar{y})^2}}. \quad (2)$$

The CCC described in [16] also measures the agreement between two variables and can be used to evaluate the degree of agreement among raters. Its formula is provided in Eq. (3). A

value of the CCC of 1.0 indicates perfect agreement, -1.0 indicates perfect reverse agreement and a value of 0 corresponds to no agreement. As with the MAE, it is sensible to differences between the two variables being compared, but as the r^2 , it is a bounded metric which does not depend on data unit or scale. For example, when comparing the previous A and B datasets, CCC has a value of 1, whereas when comparing datasets A and C it is 0.0137. So, the CCC has both the advantages of the MAE and the r^2 and meets our requirements: measures agreement and difference between the predictions simultaneously. Therefore, we have chosen this metric to evaluate precision in the conducted experiments.

$$\rho_c = \frac{2 \cdot s_{xy}}{s_x^2 + s_y^2 + (\bar{x} - \bar{y})^2}, \quad \text{where:} \quad (3)$$

$$\bar{x} = \frac{1}{N} \cdot \sum_{n=1}^N x_n$$

$$s_x^2 = \frac{1}{N} \cdot \sum_{n=1}^N (x_n - \bar{x})^2$$

$$s_y^2 = \frac{1}{N} \cdot \sum_{n=1}^N (y_n - \bar{y})^2$$

$$s_{xy} = \frac{1}{N} \cdot \sum_{n=1}^N (x_n - \bar{x}) \cdot (y_n - \bar{y}).$$

2.2. State of the art

In the last years there have been several efforts that introduced the concept of energy efficiency into the Cloud computing field and proposed related solutions and concepts.

In [17] different hardware configurations' impact on energy consumption and performance for typical applications are evaluated, and it is concluded that different workloads need different hardware configurations to achieve performance and energy efficiency. What is interesting of this publication is that in order to obtain energy efficiency assessments which can be compared across different workloads, a metric of the form iops/J (input/output operations per Joule) is derived. Similarly, in order to be able to compare the computing performance delivered across nodes with different architectures and for any workload type, a metric of the form MWIPS (see Section 2.1) will be used in this publication. While in [17] the evaluated energy efficiency was in the form of read/write operations by energy unit, this publication assesses the amount of computation power delivered per unit of real power consumed by the node.

The Storage Performance Council (SPC) also developed a couple of benchmarks which also evaluate the energy efficiency of a storage subsystem [18]. They report the energy efficiency with a metric of the form $\frac{T}{s \cdot W}$, where T are the I/O operations per time unit s and per electricity consumed W in Watts. In these benchmarks, the energy efficiency is evaluated at a limited number of intermediate load levels. In this publication, the energy efficiency is evaluated for all possible load levels.

The Transaction Processing Performance Council (TPC) released a benchmark which allowed to measure the energy consumption corresponding to the work performed by some of their own performance benchmarks, and to merge both performance and energy information to produce a metric of the form $\frac{W \cdot s}{T}$, where the electricity (W) per time unit (s) is divided by the total amount of transactions (T) [18]. So, in this case the energy efficiency definition is presented inverted with regards to the formulations of the previously referred publications.

Finally, the Standard Performance Evaluation Corporation (SPEC) has also released benchmarks to measure the energy efficiency of a system executing a server side Java application and of a system emulating a web server serving different kinds of web workloads [18]. These benchmarks also evaluate the consumed energy and performance at multiple load levels. The provided metric is of the form $\frac{T}{W}$, where T can represent server side Java operations or web transactions, depending on the benchmark. Of all the related work, this metric for energy efficiency is the most similar to the one studied in this publication.

So far, only benchmarks to measure the “offline energy efficiency” of a given application have been considered in the current literature, but there are no tools to measure the real time energy efficiency of a system. Moreover, what is interesting from an IaaS provider point of view is to be able to maximise the energy efficiency of its infrastructure, and to do that it needs tools to predict the future energy efficiency that it will have. This is also not addressed by the analysed benchmarks. This is actually one of the main innovations of the present publication: being able to assess and predict the real time energy efficiency of the infrastructure.

Energy efficiency assessments and forecasts can be used by placement algorithms in order to maximise the energy efficiency of the infrastructure. In [19] two task consolidation algorithms targeted for batch applications take into account resource utilisations and power consumptions when deciding in which node a task must be run. The provided energy efficiency metric in this publication can be used by VM placement algorithms independently of the underlying workload type (web, batch, etc.). Moreover, the resource utilisation incurred by a given VM (or task) is not the same if it runs in nodes with different characteristics. Therefore, we consider performance as the universal metric which can be translated for each specific node into resource utilisation and in turn into power consumption. Furthermore, the algorithms in [19] perform placement optimisation when a task has to be scheduled to a particular node, but not during its execution. In Cloud computing environments, tasks can be encapsulated in VMs, which can be migrated while running. So, the impact of migrating a VM on the infrastructure's energy efficiency has to be studied in order to allow placement algorithms to take it into consideration. This way, energy efficiency can be optimised not only during VM deployment but also during VM execution.

The authors in [7] present a resource management technique for Cloud datacenters that leverages from energy efficiency measurements. The solution proposed mainly focuses in minimising the energy consumption by migrating and consolidating VMs while keeping the QoS level as high as possible. The findings of that experiment showed that there is great potential in reducing the energy consumption by optimising the resource management in a Cloud infrastructure.

While the previously described benchmarks in [18] are used to measure the offline energy efficiency of a node or set of nodes, the algorithm described in [20] considers real-time metrics, but not the energy efficiency itself. It actually considers metrics which potentially affect the energy efficiency. In this publication, the energy efficiency is directly considered as both the delivered performance and power are measured using real-time metrics and power-measuring devices.

Multiple power models describing the power consumption of a server by using different metrics and relationships between them can be found in literature. For instance, in [21] a linear power model relating CPU utilisation to power is used. Authors in [6] also use a power model that predicts the total system power based on the CPU utilisation. They compare a simple linear model like the one presented in [21] with their own proposed non-linear empirical one, which is based on a calibration parameter that minimises the squared error of the power measurements vs. the predicted ones. Even though they achieve a high precision using this non-

linear model, the experimentation only shows results for dual-core processors. Similarly, non-linear models have been demonstrated in [22] to provide high accuracy (better than linear models) for system power prediction, even for multi-processor systems.

According to this, we plan to extend our power model in order to be non-linear. Whereas this will enhance the accuracy of the model, it will also increase the computation cost and the complexity of the formulation. Most of the exposed formulas in Section 3 that calculate the power consumption at different levels consider the power model as a black box. These formulas can directly use a non-linear model without any adaptation. However, the formulation of Sections 3.2.2, 3.2.3 and 3.2.6 assume a linear model based on CPU utilisation, since this allows for a simple and quick estimation of the energy efficiency for situations in which the destination nodes of the VMs being deployed/migrated are not known beforehand. The rationale of these formulas should be reconsidered if a non-linear model is used.

Authors in [23] argue that even though the CPU has been considered the most power-consuming component of a system, memory power consumption is likely to be equally or even more important in the future. They also highlight that more than 30%–40% of the power is spent on the rest of the system components, including the disk and the network, among others. The power model they present is more complex than the two previously exposed in the sense that it uses additional system metrics to derive it, including CPU utilisation, off-chip memory access count, hard disk I/O rate, and network I/O rate. This approach of considering the power consumption of other system components apart from the CPU fits better with the heterogeneous nature of the workloads executed in current datacenters, and for this reason, we will consider such a model in our future work. Note that this change will imply a higher system complexity because we need to keep track of a greater amount of metrics of the system and the VMs being executed, and we must be able to predict each of them in order to use the model to predict the power consumption.

It also has to be observed that both in [19,7] the evaluation of the models is performed by means of simulation, and not with real experimentation. In [20], they actually conduct their experimentation on a state of the art testbed. The experimentation of this publication has also been conducted on real physical servers, using real CPU utilisation and power data.

So, tools to evaluate the energy efficiency at real time during VM operation and to be able to consider the possibility of performing VM deployments/undeployments or migrations are needed by Cloud providers in order to optimise the energy efficiency of their infrastructure. Taking into consideration the exposed requirements, additional effort has been put in the present work to allow the Ecoefficiency Tool to forecast the infrastructure's energy efficiency for a variety of potential actions, which can then be used by VM placement algorithms to maximise the energy efficiency in real time.

3. Assessment and forecast of the eco-efficiency for infrastructure providers

As stated in Sections 1 and 2, VM placement algorithms need to know the energy efficiency at which a node or a set of nodes are running. Some service contracts might also establish constraints regarding the minimum energy efficiency that a VM or the whole service should accomplish. What is more, these algorithms also need to know what will be the effect of performing certain actions (such as VM deployment, migration, and cancellation or service deployment) on the energy efficiency of the whole infrastructure.

Energy efficiency is defined as the ratio between the useful work performed and the energy consumed to do it [8]. Note that this definition is equivalent to the ratio between the amount of work

performed each second (performance) and the power delivered during that second (see Eq. (4)).

$$EnEf = \frac{\text{Useful work}}{\text{Energy [J]}} = \frac{\frac{\text{Useful work}}{s}}{\frac{\text{Energy [J]}}{s}} = \frac{\text{Performance}}{\text{Power [W]}}. \quad (4)$$

Following this last definition, our aim is to obtain a formula which evaluates this relationship at physical node level in the first place (Section 3.1.1). The obtained formula will then be extended to obtain similar definitions for the remaining levels of our interest: infrastructure level (Section 3.1.2), virtual machine level (Section 3.1.3) and service level (Section 3.1.4).

An IaaS provider can be interested in having energy efficiency information for all of the aforementioned levels, as it can be used by its placement algorithms. These algorithms will make decisions in order to optimise, among other things, the energy efficiency of the whole infrastructure while at the same time fulfilling the SLA agreements with the provider's clients. However, a SaaS is only interested in the energy efficiency achieved by a given service, and not about the individual energy efficiency of the VMs comprising it or the nodes where they are being executed. Note that the performance units mentioned in Eq. (4) do not need to be the same when considered from an IaaS or a SaaS perspective. This aspect will be further discussed in Section 3.1.4.

The energy efficiency gives an idea of the performance delivered per unit of power being consumed, but it does not reflect the environmental impact caused to deliver that performance. The ecological efficiency tackles this aspect, as it evaluates the ratio between the useful work performed and the carbon dioxide emitted to do it, which is equivalent to the amount of work performed each second divided by the carbon dioxide emission rate (in $\frac{gr}{s}$) during that second. This definition is very similar to the one of energy efficiency. In fact, it can be demonstrated (refer to [24] or [25]) that the ecological efficiency can be calculated out of the energy efficiency by using a conversion factor which depends on the amount of brown energy used by the provider, the origin of such energy mix (coal, gas, etc.) and the associated emissions to each energy source. Therefore, in this publication we will focus our study on the energy efficiency, because the ecological efficiency can be easily calculated out of it.

This section contains the description of the main research body conducted in this paper. In Section 3.1, the definition of energy efficiency is extended for each of the levels of interest of an IaaS provider: node, infrastructure, VM and service levels. Finally, Section 3.2 describes how to calculate the impact on the infrastructure energy efficiency if a VM is deployed, migrated or cancelled, or if a service is deployed. A summary of all the symbols used in this section of the paper can be found in Table 1.

3.1. Energy efficiency assessments and predictions

In the following subsections, different considerations concerning energy efficiency assessment calculations and predictions are discussed. As commented before, only the energy efficiency formulas and explanations will be tackled, as the ecological efficiency can be easily calculated with a conversion out of the energy efficiency. Section 3.1.1 introduces the methodology followed to evaluate and predict the energy efficiency of a single physical node of an IaaS, while Section 3.1.2 uses these results to do it at an IaaS level. In Section 3.1.3 these calculations are discussed when performed at VM level, while in Section 3.1.4 these results are used to assess and predict the energy efficiency of a service running on several VMs.

3.1.1. Energy efficiency at node level

An IaaS is not interested on the kind of work being performed by the applications running in their infrastructure, but on the

Table 1
Summary of symbols used in Section 3.

$EnEf$	Energy efficiency
$Perf_{node_i}$	Current performance of node i in Computing Units (CU)
VMs	Set of virtual machines, index j
VMs_i	Subset of virtual machines being executed in node i , index j
U_{VM_j}	CPU utilisation of VM j in node i
$Perf_{max_i}$	Maximum performance of node i in Computing Units (CU)
$Power_{node_i}$	Current power consumption of node i
R_i	Real power consumption of node i
PUE	Power Usage Effectiveness
$EnEf_{node_i}$	Energy efficiency assessment of node i
$\widehat{Perf}_{node_i}$	Forecasted performance of node i in Computing Units (CU)
$\widehat{U}_{VM_j}$	Forecasted CPU utilisation of VM j in node i
$\widehat{Power}_{node_i}$	Forecasted power consumption of node i
$\widehat{R}_i$	Forecasted real power consumption of node i
$\widehat{U}_0$	Forecasted CPU utilisation of Domain-0 of node i
$\widehat{EnEf}_{node_i}$	Forecasted energy efficiency value of node i
$EnEf_{infr}$	Energy efficiency assessment of the whole IaaS provider infrastructure
$nodes$	Set of currently active nodes (when assessing) or potentially active nodes (when forecasting), index i
$\widehat{EnEf}_{infr}$	Forecasted energy efficiency value of the whole IaaS provider infrastructure
$Perf_{VM_j}$	Current performance of VM j in Computing Units (CU)
$Power_{VM_j}$	Current power consumption of VM j
U_0^i	CPU utilisation of Domain-0 of node i
n_i	Number of VMs being executed in node i
$EnEf_{VM_j}$	Energy efficiency assessment of VM j
$\widehat{Perf}_{VM_j}$	Forecasted performance of VM j in Computing Units (CU)
$\widehat{Power}_{VM_j}$	Forecasted power consumption of VM j
$\widehat{EnEf}_{VM_j}$	Forecasted energy efficiency of VM j
$EnEf_{serv_k}$	Energy efficiency assessment of service k
$services$	Set of services, index k
$\widehat{EnEf}_{serv_k}$	Forecasted energy efficiency of service k
B	Set of VMs executing batch workloads
CPU_{VM_j}	Number of CPUs required by VM j
CPU_{node_i}	Total number of CPUs in node i
I	Set of VMs executing interactive workloads
$Perf_{VM_{l,hist.}}$	Historical performance recorded for VMs with similar characteristics as VM l
$\overline{Perf}_{CPU}$	Average performance per CPU in the whole IaaS provider infrastructure
N	Number of nodes of the whole IaaS provider infrastructure
$Power_{min_i}$	Minimum power consumption of node i (when $U = 0\%$)
CPU_{alloc_i}	Total number of CPUs allocated to services in node i
CPU_{depl}	Additional number of CPUs as part of the new deployment (either of VM or service)
$\overline{Power}_{incr}$	Average incremental power consumption per CPU in the whole IaaS provider infrastructure
$Power_{incr_i}$	Incremental power consumption per CPU in node i
CPU_{serv_k}	Number of CPUs required by service k
$\delta_{cert.}$	Boolean variable: equals to 1 if the provider owns all the certifications required in the Service Manifest

resource utilisation they incur in the system. So, from an IaaS point of view, the useful work per second carried out by a server running several VMs can correspond to the amount of computing performance delivered to them during that second, based on the assumption that the applications being run on the VMs are computing-intensive.

A first approach could be to evaluate the delivered computing performance per second as the amount of CPU time which has been assigned to the VMs running in the node during that second (amount of CPU utilisation). However, it is obvious that a 1990 computer with 50% of CPU utilisation will not give the same performance as a 2010 computer with the same CPU utilisation.

Possible metrics to measure the delivered performance per second could be MIPS or MFLOPS in this sense, given that they are independent of the underlying hardware and allows to compare the delivered performance per second in heterogeneous clusters. In particular, the MWIPS metric (Millions of Whetstone Instructions Per Second) is currently being used by the Eco-Efficiency Tool as it measures both integer and floating-point arithmetic performance. However, all these possible metrics will be referred in a general way as Computing Units (CU) throughout the rest of this paper. Therefore, a CU will relate to the amount of delivered performance per second.

The maximum computing power (performance per second) which can be delivered by a node i is represented as $Perf_{max_i}$ and can be measured as the number of CUs delivered by the node when its processors are running at full speed with 100% of CPU utilisation. By multiplying $Perf_{max_i}$ by the sum of CPU utilisations (normalised to 1) incurred by the VMs running in the node, we can obtain the amount of computing power delivered by the node to the VMs (see Eq. (5)), assuming that the distribution of computing power among the VMs is proportional to their CPU utilisation. These assumptions are valid when running computing-intensive applications.

Note that the CPU utilisation of the privileged domain (named Domain-0 in Xen Hypervisor nomenclature) has not been included in this sum, since it is not considered to be useful work. In fact, it can be considered as the overhead needed to run them. Also observe that the CPU utilisation values range from 0 to 100 independently of the number of processors present in the node.

$$Perf_{node_i} = \sum_{j \in VMs_i} \frac{U_{VM_j^i}}{100} \cdot Perf_{max_i}. \quad (5)$$

On the other hand, the amount of power consumed by the node to execute these VMs along with the privileged domain corresponds to the real power and not to the apparent power, given that the real power reflects the energy really consumed and charged for by utilities [26]. It also needs to be taken into account the power overhead needed to run the node (cooling, lighting and power delivery losses). Following the definition of PUE exposed in Section 2.1, the total power consumption will be the real power consumed by the node multiplied by the PUE of the datacenter. Therefore, the total power consumption incurred by the node, including overheads, is depicted in Eq. (6).

$$Pwr_{node_i} = R_i \cdot PUE. \quad (6)$$

Taking into consideration Eqs. (5) and (6), the energy efficiency of a node from an IaaS point of view can be calculated using Eq. (7).

$$EnEf_{node_i} = \frac{Perf_{node_i}}{Pwr_{node_i}}. \quad (7)$$

One way to forecast the node energy efficiency for a given time in the future would be to keep track of the previous assessment results and extrapolate them using a linear regression. However, this approach does not take into account that the VMs running in a particular host can have completely independent behaviours. In order to overcome this problem, each of the variable terms of Eqs. (5) and (6) are predicted separately. Then, the energy efficiency is forecasted using the same relationship as in Eq. (7).

On the one hand, the CPU utilisations of each VM and the Domain-0 are forecasted using an estimator which combines four prediction methods: moving average, linear regression, exponential smoothing and double exponential smoothing (see Section 4). Each CPU estimator uses as an input the previous CPU utilisation

samples of the VM and outputs the best CPU utilisation forecast based on the last errors obtained by each of the prediction methods. Note that each VM/Domain-0 uses an independent CPU estimator. The node's maximum computing performance $Perf_{max_i}$ will remain constant, as it is inherent to the node. Therefore, the performance forecast of a node can be calculated as exposed in Eq. (8).

$$\widehat{Perf}_{node_i} = \sum_{j \in VMs_i} \frac{\widehat{U}_{VM_j^i}}{100} \cdot Perf_{max_i}. \quad (8)$$

On the other hand, the node's real power consumption can be estimated by using an offline power characterisation of the node, like in [6], and evaluating it for the total forecasted CPU load. This characterisation is depicted as function $\widehat{R}_i(U_i)$, which gives as an output the estimated real power consumption in Watts of node i for a given CPU utilisation in the range of 0–100, independently of its number of cores (refer to the power model in [24] for more details). The PUE is inherent to the datacenter and can be assumed to be invariant for short time elapses (see Section 2.1). Considering these two exposed factors, the forecasted total power consumption incurred by the node, including overheads, can be calculated as in Eq. (9). Note that the Domain-0 forecasted CPU utilisation ($\widehat{U}_0^i$) also has to be considered in the estimation of the real power consumption as it is part of the CPU utilisation of the physical node.

$$\widehat{Pwr}_{node_i} = \widehat{R}_i \left(\widehat{U}_0^i + \sum_{j \in VMs_i} \widehat{U}_{VM_j^i} \right) \cdot PUE. \quad (9)$$

Finally, the predicted node energy efficiency can be calculated using Eq. (10). This approach accounts for the different behaviour of the VMs running in a particular host.

$$\widehat{EnEf}_{node_i} = \frac{\widehat{Perf}_{node_i}}{\widehat{Pwr}_{node_i}}. \quad (10)$$

3.1.2. Energy efficiency at infrastructure level

In order to describe the energy efficiency at infrastructure level following the general definition in Eq. (4), the total performance and power delivered by all the nodes of the infrastructure must be calculated. Using Eqs. (5) and (6), this calculation is very simple and the resulting energy efficiency formula at infrastructure level results as depicted in Eq. (11).

$$EnEf_{infr} = \frac{\sum_{i \in nodes} Perf_{node_i}}{\sum_{i \in nodes} Pwr_{node_i}}. \quad (11)$$

Following the same reasoning as in Eq. (11), the forecasted energy efficiency at infrastructure level can be calculated by forecasting each of the terms of Eq. (11), which in fact means using Eqs. (8) and (9) to build the final formula, resulting in Eq. (12).

When issuing the calculation of this forecast, though, the placement algorithm might consider the possibility of switching on new nodes or even switching off unused ones. In this case, the set *nodes* will consist of the *potentially* active nodes for the forecast, that is, the subset of nodes which will be operating at the time for which the forecast is done.

$$\widehat{EnEf}_{infr} = \frac{\sum_{i \in nodes} \widehat{Perf}_{node_i}}{\sum_{i \in nodes} \widehat{Pwr}_{node_i}}. \quad (12)$$

3.1.3. Energy efficiency at VM level

The useful work carried out by a VM from an IaaS perspective corresponds to the fraction (normalised $U_{VM_j^i}$) of the maximum

performance of the node where it is running ($Perf_{max_i}$) consumed by this VM, as depicted in Eq. (13).

$$Perf_{VM_l} | l \in VMs_i = \frac{U_{VM_l}^i}{100} \cdot Perf_{max_i}. \quad (13)$$

In order to determine the approximate power consumption incurred by a VM, its proportional part of the total power consumption R_i of the node where it is being executed is derived based on the CPU utilisation of the VM, including overheads. This proportional distribution with regards to the CPU utilisation is accomplished with the second term of Eq. (14). As in [26], a linear relationship between a node's real power consumption and its total CPU utilisation is assumed. Since Domain-0's CPU utilisation is required to run all of the n_i VMs being executed in node i , it is considered as an overhead which is uniformly distributed between them. Therefore, the real power consumption of a VM corresponds to the first two terms of Eq. (14), which is in turn multiplied by the datacenter's PUE in order to determine the total amount of power needed to run the VM in the provider's premises.

$$Pwr_{VM_l} | l \in VMs_i = R_i \cdot \frac{U_{VM_l}^i + \frac{U_0^i}{n_i}}{U_0^i + \sum_{j \in VMs_i} U_{VM_j}^i} \cdot PUE. \quad (14)$$

The energy efficiency of a VM can be calculated, using the results of Eqs. (13) and (14), as in Eq. (15).

$$EnEf_{VM_l} | l \in VMs_i = \frac{Perf_{VM_l} | l \in VMs_i}{Pwr_{VM_l} | l \in VMs_i}. \quad (15)$$

As in Section 3.1.1, the forecasted VM energy efficiency can be computed by estimating both the performance and the power consumption for that particular time in the future. The performance delivered in the future only depends on the future CPU utilisation that this VM will cause in the node where it runs and on the maximum performance which this node can deliver, which is inherent to the node and remains constant, as shown in Eq. (16).

$$\widehat{Perf}_{VM_l} | l \in VMs_i = \frac{\widehat{U}_{VM_l}^i}{100} \cdot Perf_{max_i}. \quad (16)$$

The future power consumption of a VM corresponds to the proportional part of the future real power consumption of the node where the VM is running. As in Eq. (9), the future real power consumption of the node is obtained from the offline power model described in [24], using as input parameter the sum of all the forecasted CPU utilisations of all the VMs running in the node, as well as Domain-0's one (estimated as previously described in Section 3.1.1). The proportional factor which describes the part of this power consumed by the VM is calculated as in Eq. (14), by forecasting all of its terms. The resulting consumed power is multiplied by the datacenter's PUE (can be considered constant for short time elapses, see Section 2.1) to account for the extra power needed to sustain the infrastructure, finally resulting in Eq. (17).

$$\widehat{Pwr}_{VM_l} | l \in VMs_i = \widehat{R}_i \left(\widehat{U}_0^i + \sum_{j \in VMs_i} \widehat{U}_{VM_j}^i \right) \cdot \frac{\widehat{U}_{VM_l}^i + \frac{\widehat{U}_0^i}{n_i}}{\widehat{U}_0^i + \sum_{j \in VMs_i} \widehat{U}_{VM_j}^i} \cdot PUE. \quad (17)$$

Using the results of Eqs. (16) and (17), the expected energy efficiency of a VM can be calculated as in Eq. (18).

$$\widehat{EnEf}_{VM_l} | l \in VMs_i = \frac{\widehat{Perf}_{VM_l} | l \in VMs_i}{\widehat{Pwr}_{VM_l} | l \in VMs_i}. \quad (18)$$

3.1.4. Energy efficiency at service level

A service is a set of components which can interact between them in order to provide a complete functionality to a client. Usually, these components are encapsulated in a set of VMs which can run in different hosts. Therefore, the energy efficiency of a service can be calculated as the sum of the performance delivered to all the VMs composing the service divided by the sum of their power consumption. Eq. (19) summarises this idea.

$$EnEf_{serv_k} = \frac{\sum_{j \in serv_k} Perf_{VM_j}}{\sum_{j \in serv_k} Pwr_{VM_j}}. \quad (19)$$

In order to calculate the energy efficiency of a service for a future time, each of the delivered performance and power terms present in Eq. (19) must be predicted, resulting in Eq. (20).

$$\widehat{EnEf}_{serv_k} = \frac{\sum_{j \in serv_k} \widehat{Perf}_{VM_j}}{\sum_{j \in serv_k} \widehat{Pwr}_{VM_j}}. \quad (20)$$

Note that a SaaS provider can select multiple IaaS providers to run different partitions of a given service. Locally, each IaaS provider will assess and forecast the energy efficiency of each service partition as described in Eqs. (19) and (20), given that an IaaS provider is not necessarily aware that it is running just a partition of a wider service. The SaaS provider will use the same formulas, but it will have to separately ask each IaaS provider for the total performance and power being delivered to the service partition (a subset of VMs of the whole service) deployed in that particular IaaS provider, and aggregate all the performance and power deliveries separately in order to calculate the overall service's energy efficiency.

The approach described in this subsection can be useful for both IaaS and SaaS providers during service operation because it gives an idea of the computing power delivered to the service by Watt consumed. However, SaaS providers can be also interested in the performance delivered by the application in terms of application metrics like response time or throughput instead of the CUs it consumes. In this case, in order for a SaaS to assess and forecast the energy efficiency of a given service during its operation, it needs to know the power consumed by the VMs comprising it (or its estimation when forecasting the energy efficiency), so it can come up with metrics like throughput per Watt consumed, or session rate per Watt consumed.

3.2. Energy efficiency predictions for potential actions

In the following subsections, the mathematical models used to predict the energy efficiency impact on the infrastructure of an IaaS provider when performing actions such as VM or service deployment, VM cancellation and VM migration are discussed. Different variants of the VM deployment and migration, for unknown and known destination nodes, are presented. Depending on how the IaaS provider operates and of each particular use case, it may use one version or the other, or even both. They might be possibly used in a different order from the one used to expose them in this publication. However, we have chosen this order since later section formulations are based on those presented in the previous sections.

We envision several use cases in which the formulations presented in the following subsections can be used:

- If a SaaS provider issues the deployment of a new service, it will require knowing what the delivered energy efficiency to this service will be, while the IaaS provider will need to know how this service deployment will impact in its infrastructure energy efficiency. Section 3.2.3 covers these aspects.

- When a service is already running, the SaaS provider might consider replicating some of its VMs to be able to cope with extra concurrent users. When requested to create a new VM instance, the IaaS provider can perform a rough evaluation on the energy efficiency impact of performing such an action by using the calculations of Section 3.2.2. If the new VM is finally accepted, the IaaS provider's VM placement algorithm can decide at which node should it be deployed to maximise the energy efficiency using the calculations provided in Section 3.2.1.
- When a VM is found to be idle, the IaaS provider can consider turning it off (if this action still fulfils the service SLAs). The IaaS provider can study the impact on the infrastructure energy efficiency of cancelling that VM using the calculations of Section 3.2.4.
- During its operation, an IaaS provider might reach a situation in which several nodes are used far below its maximum capacity. In such a context, it might consider migrating the VMs of one node to another so it can turn off the first one. It can perform a rough but quick estimation of the impact that migration will have on the infrastructure energy efficiency by using the calculations of Section 3.2.6, without still deciding to which particular node this VM will be migrated. If the IaaS provider finally decides to migrate the VM, its VM placement algorithm can use the calculations provided in Section 3.2.5 to decide at which destination node should the VM be migrated to in order to maximise the infrastructure energy efficiency.

3.2.1. VM deployment (known placement)

A VM is the minimal deployable unit in an IaaS provider infrastructure. A VM can be deployed as part of a new service deployment or as part of a running service. During a given service operation, its virtual machines suffer variations in the resource demand depending on the users' workload or the amount of parallel tasks waiting to be executed. In some situations, these VMs become overloaded and may need to be replicated to cope with the extra load.

If a VM deployment takes place the global power consumption and thereby energy associated costs will increase. One could aim for the energy efficiency of the infrastructure by using consolidation, but if the replicated VMs cannot fit in the currently active physical hosts, new physical hosts will have to be turned on and then the global energy efficiency will decrease. So, the impact of deploying a VM in terms of energy efficiency of the whole infrastructure needs to be predicted. In this section, this aspect is calculated for a context where the destination node is known.

The total computing performance delivered by the infrastructure will correspond to the one which would be delivered if no changes were made plus the extra performance delivered to the newly created VM (Eq. (21)). This second term depends on the workload type that will be executed inside the newly deployed VM (interactive or batch workloads, refer to Section 2.1).

$$\widehat{Perf}_{infr} | VM_l \text{ depl}_{dst} = \sum_{i \in nodes} \widehat{Perf}_{node_i} + \widehat{Perf}_{VM_l} | \text{depl}_{dst}. \quad (21)$$

VMs executing batch tasks such as HPC ones normally use as many resources as they are provided. A batch VM will use the received CPUs at full utilisation. So, if a node i has a maximum performance of $Perf_{max_i}$ when using all of its CPU_{node_i} CPUs, a batch VM of CPU_{VM_l} CPUs will receive the performance depicted in Eq. (22).

$$\widehat{Perf}_{VM_l} | \text{depl}_{dst} \wedge VM_l \in B = Perf_{max_{dst}} \cdot \frac{CPU_{VM_l}}{CPU_{node_{dst}}}. \quad (22)$$

A similar reasoning is applied when the VM to be deployed executes an interactive workload, as displayed in Eq. (23). In this case, however, it must be considered that interactive workloads do not necessarily saturate the assigned CPUs. In fact, their resource utilisation is proportional to the amount of concurrent clients accessing their hosted application(s), which typically depends

on the time and day of the week. In this scenario, the best estimation of the performance that the VM will require should be based on previous observed performance measurements of previously deployed VMs executing similar interactive workloads ($Perf_{VM_l, hist.}$). This measurement should be consistent with the time and the day of the week for which the prediction is being made. Alternatively, this information could be also provided by the owner of the service. It must also be taken into account that in certain nodes, the required performance might exceed the maximum one that the node can provide to a given number of CPUs. The VM will never receive more than this maximum performance. This aspect is addressed with the $\min()$ function within the formula.

$$\begin{aligned} \widehat{Perf}_{VM_l} | \text{depl}_{dst} \wedge VM_l \in I \\ = \min \left(Perf_{VM_l, hist.}, Perf_{max_{dst}} \cdot \frac{CPU_{VM_l}}{CPU_{node_{dst}}} \right). \end{aligned} \quad (23)$$

The power consumption of all the nodes of the infrastructure excluding the destination node will follow their natural trend, independently of the VM deployment. Only the trend in the power consumption of the destination node will be affected. The offline power model of [24] is used to predict the power of the destination node (other models relying in different metrics could be used as well). In this case, the input to the model will consist of the future CPU utilisation of the privileged domain (Domain-0) and those of the already running VMs (which will follow their original trend), plus the additional CPU utilisation caused by the new VM ($100 \cdot (\widehat{Perf}_{VM_l} | \text{depl}_{dst} / Perf_{max_{dst}})$), where $\widehat{Perf}_{VM_l} | \text{depl}_{dst}$ is calculated as in Eqs. (22) or (23), depending on the workload type.

$$\begin{aligned} \widehat{Pwr}_{infr} | VM_l \text{ depl}_{dst} = \sum_{\substack{i \in nodes \\ i \neq dst}} \widehat{Pwr}_{node_i} + \widehat{R}_{dst} \\ \times \left(\widehat{U}_0^{dst} + \sum_{j \in VMs_{dst}} \widehat{U}_{VM_j}^{dst} + 100 \cdot \frac{\widehat{Perf}_{VM_l} | \text{depl}_{dst}}{Perf_{max_{dst}}} \right) \cdot PUE. \end{aligned} \quad (24)$$

Finally, the energy efficiency of the provider's infrastructure if a VM l is deployed into a given destination node can be calculated as in Eq. (25).

$$\widehat{EnEf}_{infr} | VM_l \text{ depl}_{dst} = \frac{\widehat{Perf}_{infr} | VM_l \text{ depl}_{dst}}{\widehat{Pwr}_{infr} | VM_l \text{ depl}_{dst}}. \quad (25)$$

3.2.2. VM deployment (unknown placement)

In this subsection, the forecasted energy efficiency of the infrastructure if a new VM is deployed to a yet unknown physical host is evaluated. This calculation is actually a rough estimate of the future energy efficiency. If the placement algorithm then determines that a new VM deployment might lead to a situation which better satisfies the provider's BLOs, it will proceed by evaluating to which destination node should it be deployed (using the calculations described in Section 3.2.1). This second forecast will be much more accurate, although it will require a greater amount of calculations. Therefore, the utility of the forecast described in this section is to provide a quick estimation which, in case of being potentially favourable, will trigger the calculation of the future infrastructure energy efficiency when deploying the VM to a particular node, as in Section 3.2.1.

The total computing performance which will be delivered by the infrastructure corresponds to the one which would be delivered if the VM was not deployed (first term of Eq. (26)) plus the extra performance delivered to the newly created VM (second term of the same equation, calculation details below).

$$\widehat{Perf}_{infr} | VM_l \text{ depl.} = \sum_{i \in nodes} \widehat{Perf}_{node_i} + \widehat{Perf}_{VM_l} | VM_l \text{ depl}_{unk}. \quad (26)$$

An approximate estimation of the performance which will be delivered to a new VM whose placement is unknown can be computed by aggregating the mean performance delivered to each of the CPUs that it will occupy. As the VM placement is not known in this scenario, the maximum performance of the future destination node is neither known. So, we will consider that the future destination node will deliver a maximum performance per CPU calculated as the averaged maximum performance per CPU of all the *potentially* active nodes of the datacenter, as shown in Eq. (27). Of course, there will be nodes that will be able to provide more performance while there will be others that will become saturated without being able to deliver this amount of performance. Nevertheless, the fact of considering such an average value provides us a rough but quick way to evaluate if the deployment of a new VM in the datacenter might or might not be worth it. If the VM is finally deployed, the VM placement algorithm will place it in such a fashion that guarantees that the required SLAs of the service where this VM belongs are met, thus avoiding saturating nodes and/or providing the minimum required performance to the VM.

$$\overline{Perf}_{CPU} = \frac{\sum_{i \in nodes} \frac{Perf_{max_i}}{CPU_{node_i}}}{N}. \quad (27)$$

On the one hand, a VM that executes batch tasks tends to saturate its assigned CPUs, wherever the VM is located. So, we can consider that the performance delivered to a batch VM whose future placement is unknown will be its number of CPUs multiplied by the average maximum performance per CPU of the datacenter, as depicted in Eq. (28).

$$\widehat{Perf}_{VM_I} \mid VM_I \text{ depl}_{unk} \wedge VM_I \in B = CPU_{VM_I} \cdot \overline{Perf}_{CPU}. \quad (28)$$

On the other hand, an interactive VM does not necessarily saturate its assigned CPUs. In fact, its resource utilisation is proportional to the amount of concurrent clients accessing it, which typically depends on the time and day of the week. As discussed in Section 3.2.1, performance calculation in this scenario will be based on historically observed performance measurements of previously deployed VMs executing similar interactive workloads ($Perf_{VM_I, hist.}$), but limiting this value according to the average performance per CPU of the datacenter, as shown in Eq. (29).

$$\begin{aligned} \widehat{Perf}_{VM_I} \mid VM_I \text{ depl}_{unk} \wedge VM_I \in I \\ = \min \left(Perf_{VM_I, hist.}, CPU_{VM_I} \cdot \overline{Perf}_{CPU} \right). \end{aligned} \quad (29)$$

As in Section 3.2.1, only the power consumption trend of the destination node will be modified, while the rest of the nodes will follow their natural trend. Nevertheless, given that the destination node is not known yet, in this case we will separately calculate a rough estimate of the additional power consumption caused by the VM and add it to the natural trend of all the nodes of the infrastructure. This rough estimate is the result of multiplying the number of CPUs of the VM by the mean power consumption that each of them will cause. The mean power consumption per CPU, detailed below, is calculated by averaging it for the *potentially* active nodes in which the VM can be deployed. Note that the discussed approach is only valid when using a power model that assumes a linear relationship between CPU utilisation and power, as in [24]. This model has been proved not to be very accurate for modern multicore systems in the related literature. However, its simplicity allows us to get easily an initial rough estimation of the power consumption per CPU.

As discussed in [24], the power consumption of a single node can be characterised as an idle power Pwr_{mini} , which is independent of the node load, and a variable power which follows

a linear trend with regards to the CPU utilisation. Therefore, the power consumption of each CPU can be modelled as an idle part and a variable part which depends on the CPU utilisation.

On the one hand, the idle part is a “necessary power consumption” for a node to be able to run, it can be considered as an overhead. If only one CPU of a node is active all this overhead is attributed to this CPU, while if n CPUs are active within this node, each one can be considered to consume $\frac{Pwr_{mini}}{n}$ Watts of idle power. This idea is extrapolated when calculating the idle power consumption associated to a single CPU when considering the whole list of *potentially* active nodes. In this case, the idle power associated to a single CPU corresponds to the total idle power of all the *potentially* active nodes divided by the total number of CPUs already allocated to VMs running in the nodes plus the additional CPUs which will be used in the deployment to where this VM belongs. That is, if only a single VM is deployed as part of an elasticity process, the additional CPUs will be $CPU_{depl.} = CPU_{VM_I}$, whereas if this VM is part of a service deployment we will use $CPU_{depl.} = CPU_{serv_k}$. In this section, thus, we will use $CPU_{depl.} = CPU_{VM_I}$. This corresponds to the first part of Eq. (30).

On the other hand, the variable part corresponds to the second part of Eq. (30). The incremental power Pwr_{incr_i} is the extra power consumption of a node when one of its CPUs is occupied at 100% (refer to the power model in [24]). Because the destination node is not known, we consider the mean incremental power of all the *potentially* active nodes of the infrastructure, which can be calculated as in Eq. (31). By multiplying the mean incremental power by the normalised utilisation of the CPUs of the VM, we are able to calculate the variable power per CPU of the VM. As we are considering that each CPU of the *potentially* active nodes provides at most an average of $\overline{Perf}_{CPU}$ CUs, if the VM receives a performance per CPU of $\frac{Perf_{VM_I} \mid VM_I \text{ depl}_{unk}}{CPU_{VM_I}}$ (where $\widehat{Perf}_{VM_I} \mid VM_I \text{ depl}_{unk}$ is calculated as in Eqs. (28) or (29), depending on the workload type the VM executes), the utilisation of each of its CPUs will be $\frac{Perf_{VM_I} \mid VM_I \text{ depl}_{unk}}{CPU_{VM_I} \cdot \overline{Perf}_{CPU}}$.

$$\begin{aligned} \widehat{Pwr}_{VM_I} \mid VM_I \text{ depl}_{unk} = CPU_{VM_I} \cdot \left(\frac{\sum_{i \in nodes} Pwr_{mini}}{\sum_{i \in nodes} CPU_{alloc_i} + CPU_{depl.}} \right. \\ \left. + \frac{\widehat{Perf}_{VM_I} \mid VM_I \text{ depl}_{unk}}{CPU_{VM_I} \cdot \overline{Perf}_{CPU}} \right) \cdot PUE \end{aligned} \quad (30)$$

$$\overline{Pwr}_{incr} = \frac{\sum_{i \in nodes} Pwr_{incr_i}}{N}. \quad (31)$$

The overall impact on the infrastructure power consumption of performing a VM deployment can thus be expressed as the original power consumption trend of the *potentially* active nodes of the infrastructure plus the additional power consumption incurred by the VM, as depicted in Eq. (32).

$$\begin{aligned} \widehat{Pwr}_{infr} \mid VM_I \text{ depl}_{unk} \\ = \sum_{i \in nodes} \widehat{Pwr}_{node_i} + \widehat{Pwr}_{VM_I} \mid VM_I \text{ depl}_{unk}. \end{aligned} \quad (32)$$

Finally, the potential energy efficiency of the infrastructure upon a VM deployment to a yet unknown destination physical host can be calculated using the previously obtained formulas and combining them as shown in Eq. (33).

$$\widehat{EnEf}_{infr} \mid VM_I \text{ depl}_{unk} = \frac{\widehat{Perf}_{infr} \mid VM_I \text{ depl}_{unk}}{\widehat{Pwr}_{infr} \mid VM_I \text{ depl}_{unk}}. \quad (33)$$

3.2.3. Service deployment

When a new service deployment is issued to a SaaS provider, it must decide to which IaaS provider it will forward the service creation request in order to issue the creation. In some cases, the owner of the service may have specified a minimum desired eco-efficiency level (either energy and/or ecological efficiency) for that service (when evaluated at service level, see Section 3.1.4). Moreover, the service owner may also have specified a given set of environmental-aware certifications which the IaaS provider where the service will be deployed must possess (i.e. LEED, BREEAM, CASBEE, ISO14000, etc.). In this case, the SaaS provider needs to determine whether the minimum required eco-efficiency for that service can be guaranteed by the IaaS provider where the service will be deployed, as well as if it meets the specified certification requirements.

To predict the energy efficiency at service level, the performance this service will require and the power consumption that it will incur need to be determined.

Given that the future location of the VMs composing the service is not known beforehand, we can calculate the overall performance delivered to a service whose VMs have not been assigned to any destination node as the sum of the performances delivered to its VMs using the calculation of $\widehat{Perf}_{VM_l} | VM_l \text{ depl}_{unk}$ explained in Section 3.2.2, as showed in Eq. (34).

$$\widehat{Perf}_{serv_k} | serv_k \text{ depl.} = \sum_{j \in serv_k} \widehat{Perf}_{VM_j} | VM_j \text{ depl}_{unk}. \quad (34)$$

The power consumed by a new service can be calculated as in Eq. (35), by doing the sum of the power consumption of each VM composing it (using Eq. (30)). Consider that in this case, we will use $CPU_{depl.} = CPU_{serv_k}$, because the idle power consumption will be shared between the CPUs of the VMs already running in the infrastructure and the CPUs of all the VMs composing the new service. Note that as in Section 3.2.2, this calculation is made by considering a linear power model based on the CPU utilisation.

$$\widehat{Pwr}_{serv_k} | serv_k \text{ depl.} = \sum_{j \in serv_k} \widehat{Pwr}_{VM_j} | VM_j \text{ depl}_{unk}. \quad (35)$$

Taking into account the explanations in this subsection, the energy efficiency of a service upon its deployment can be calculated as in Eq. (36). The term $\delta_{cert.}$ is a boolean variable which is equal to 1 if all the required environmental-aware certifications specified in the service manifest (service contract specified by the service owner) are in possession of the IaaS provider where the service is to be deployed, and 0 otherwise. Therefore, if the IaaS provider has the required certifications, the forecasted energy efficiency of the service will be different from 0 and the service might be deployed in that provider (if the specified minimum energy efficiency requirements are met). In case the IaaS provider does not have any of the required certifications, the returned energy efficiency forecast will be 0 and that particular IaaS provider will not be selected for the service deployment.

$$\widehat{EnEf}_{serv_k} | serv_k \text{ depl.} = \delta_{cert.} \cdot \frac{\widehat{Perf}_{serv_k} | serv_k \text{ depl.}}{\widehat{Pwr}_{serv_k} | serv_k \text{ depl.}}. \quad (36)$$

Similarly, from an IaaS provider point of view, it is required to know which will be the impact of deploying a service in the infrastructure. Depending on the forecasted impact and taking into account the BLOs of the IaaS provider, the service will be accepted or rejected by the IaaS provider. The forecasted energy efficiency of the whole infrastructure of an IaaS provider upon a service deployment is depicted in Eq. (37). The only difference with regards to Eq. (12) is that the extra performance being delivered to the new service as well as the power it will consume are now included in the nominator and the denominator of the formula, respectively. Note that as in Eq. (36), the boolean term $\delta_{cert.}$ is used to consider the fact that if the IaaS provider does not possess all

the required environmental-aware certifications specified in the service manifest, no extra performance or power will be delivered to the service, because it will be actually rejected.

$$\widehat{EnEf}_{infr} | serv_k \text{ depl.}$$

$$= \frac{\sum_{i \in nodes} Perf_{node_i} + \delta_{cert.} \cdot \widehat{Perf}_{serv_k} | serv_k \text{ depl.}}{\sum_{i \in nodes} Pwr_{node_i} + \delta_{cert.} \cdot \widehat{Pwr}_{serv_k} | serv_k \text{ depl.}}. \quad (37)$$

3.2.4. VM cancellation

In this subsection, our aim is to determine the potential energy efficiency at infrastructure level for the hypothetical case in which a VM l that is being executed at node src is cancelled (shut down).

When a VM l is cancelled, it will cease to ask for CPU time to the node it is being run (src). So, the forecasted useful performance delivered by node src will have to exclude the CPU utilisation of VM l from the original calculation in Eq. (8). The rest of nodes of the infrastructure will not suffer any modification in this sense, as the number of VMs they host will remain constant. This reasoning is summarised in Eq. (38).

$$\begin{aligned} \widehat{Perf}_{infr} | VM_l \text{ canc.} &= \sum_{\substack{i \in nodes \\ i \neq src}} \widehat{Perf}_{node_i} \\ &+ \sum_{\substack{j \in VMs_{src} \\ j \neq l}} \frac{\widehat{U}_{VM_j^{src}}}{100} \cdot Perf_{max_{src}}. \end{aligned} \quad (38)$$

Similarly, the expected power consumption of the nodes of the infrastructure will follow Eq. (9) except for node src . For this particular node, its expected power consumption can be induced using the same reasoning as in Eq. (9), but excluding the CPU utilisation of VM l from the calculation.

$$\begin{aligned} \widehat{Pwr}_{infr} | VM_l \text{ canc.} &= \sum_{\substack{i \in nodes \\ i \neq src}} \widehat{Pwr}_{node_i} \\ &+ \widehat{R}_{src} \left(\widehat{U}_0^{src} + \sum_{\substack{j \in VMs_{src} \\ j \neq l}} \widehat{U}_{VM_j^{src}} \right) \cdot PUE. \end{aligned} \quad (39)$$

Taking into consideration the previous results, the expected energy efficiency of the infrastructure if a VM l is cancelled can be calculated as in Eq. (40).

$$\widehat{EnEf}_{infr} | VM_l \text{ canc.} = \frac{\widehat{Perf}_{infr} | VM_l \text{ canc.}}{\widehat{Pwr}_{infr} | VM_l \text{ canc.}}. \quad (40)$$

3.2.5. VM Migration (known placement)

Once the placement algorithm of an IaaS provider determines that performing a VM migration to an undefined node can help meeting the business level objectives of that provider (see Section 3.2.6), it will then evaluate in detail to which specific node this VM should be migrated.

When a VM is migrated, the impact on the performance it receives can be studied depending on the workload type it executes (interactive or batch workloads, refer to Section 2.1).

A VM executing HPC batch tasks tends to use as many resources as it is provided. In fact, it can be understood as a VM which always requires more performance than it is given, no matter in which node it is executed. This situation is depicted in Fig. 1. A VM executed in Node X is delivered $\frac{Perf_{max_x}}{CPU_{node_x}}$ CUs of performance to each of its CPUs, as it is the maximum that the node can provide. Nevertheless, given that HPC batch tasks are always “hungry” of more resources, this same VM would take more of them were they provided. So, if the VM was migrated to Node Y (situation A1), it would

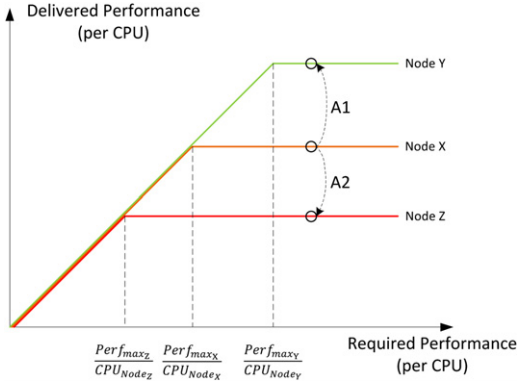


Fig. 1. Performance variation due to different migration scenarios (batch workloads).

receive $\frac{Perf_{max_Y}}{CPU_{node_Y}}$ CUs, as it is the maximum that the destination node can provide (saturation point). On the other hand, if the VM was migrated to node Z (situation A2), it would only receive $\frac{Perf_{max_Z}}{CPU_{node_Z}}$. In this situation, the VM will receive less performance than in the current node, which will lead to a longer execution time of the task because less work will be executed per unit of time. This issue must be considered by the VM placement algorithm in order to check if the required SLAs of the task will still be met.

To generalise, the performance that a VM executing HPC batch tasks receives once migrated is always limited by the maximum performance per CPU that the destination node can deliver. So, the delivered performance once migrated to node dst will be as in Eq. (41).

$$\widehat{Perf}_{VM_I} | migr_{src}^{dst} \wedge VM_I \in B = \frac{Perf_{max_{dst}}}{CPU_{node_{dst}}} \cdot CPU_{VM_I}. \quad (41)$$

A VM executing interactive workloads such as web serving uses an amount of resources that is proportional to the number of simultaneous clients performing transactions in the VM, as well as the individual transactions that each client does. So, differently from the case of batch VMs, interactive VMs do not always saturate the node in which they are executed. This leads to multiple migration scenarios, as displayed in Fig. 2. In scenario A, either if the VM is migrated from Node X to Y or Z, as the required performance per CPU is lower than the maximum that both of them can provide, the delivered performance will remain unmodified because none of the destination nodes saturate. Nevertheless, in scenarios B2, C2 and D2, the destination node cannot provide as much performance per CPU as the one requested by the VM, and the delivered performance per CPU will be limited to $\frac{Perf_{max_Z}}{CPU_{node_Z}}$. In scenario B1, the destination node (Node Y) can still provide the required performance, and thus it remains unmodified throughout the migration. In scenario C1, the VM is actually saturating the CPUs in Node X (it would make use of more performance were it provided). When migrated to Node Y, given that it can provide that extra required performance, the delivered performance increases to match the required one. Finally, in scenario D1 the VM is also saturating the current node, but even when migrated to Node Y it will still require more performance than the one that Node Y can provide, thus saturating it. So, in Scenario D1 the delivered performance per CPU will be $\frac{Perf_{max_Y}}{CPU_{node_Y}}$. Given that in scenarios C1 and D1 Node X is saturated (utilisation of the CPUs of the VM at 100%), it is impossible to determine if we are in scenario C1 or D1. However, we can assure that the delivered performance per CPU once migrated will be in the range $\frac{Perf_{max_X}}{CPU_{node_X}} \leq \widehat{Perf}_{VM_I} | l \in VMs_Y \leq \frac{Perf_{max_Y}}{CPU_{node_Y}}$.

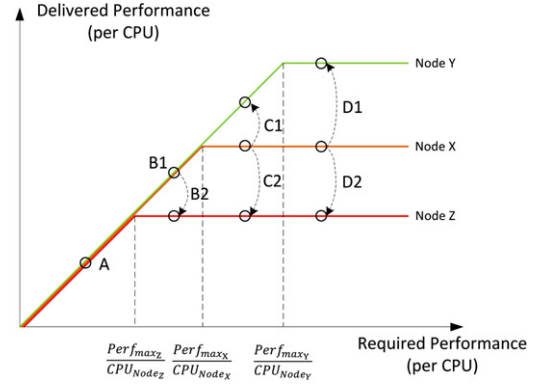


Fig. 2. Performance variation due to different migration scenarios (interactive workloads).

To generalise, the performance that a VM executing interactive workloads receives once migrated from node src to node dst can be summarised as in Eq. (42). That is, if the VM is migrated to a node with less performance per CPU or if the source node is not saturated, the future received performance will be the minimum between the current received performance and the maximum that the destination node can offer to the VM. Otherwise (and consequently, if the VM is migrated to a node with more performance per CPU and the VM is currently saturating the source node), the received performance will be in the range comprised between the maximum performance that the source and destination node can provide.

$$\widehat{Perf}_{VM_I} | migr_{src}^{dst} \wedge VM_I \in I = \begin{cases} \text{if } \left[\frac{Perf_{max_{src}}}{CPU_{node_{src}}} \geq \frac{Perf_{max_{dst}}}{CPU_{node_{dst}}} \right] \\ \quad \vee \left[\widehat{Perf}_{VM_I} | l \in VMs_{src} \leq \frac{Perf_{max_{src}}}{CPU_{node_{src}}} \cdot CPU_{VM_I} \right] : \\ \quad \min \left(\widehat{Perf}_{VM_I} | l \in VMs_{src}, \frac{Perf_{max_{dst}}}{CPU_{node_{dst}}} \cdot CPU_{VM_I} \right) \\ \text{else:} \\ \quad \frac{Perf_{max_{src}}}{CPU_{node_{src}}} \cdot CPU_{VM_I} \leq \widehat{Perf}_{VM_I} | l \in VMs_{dst} \\ \quad \leq \frac{Perf_{max_{dst}}}{CPU_{node_{dst}}} \cdot CPU_{VM_I}. \end{cases} \quad (42)$$

The impact of migrating VM I with regards to the overall performance of the infrastructure can be calculated as in Eq. (43). The source node will experiment a reduction of the delivered performance, as VM_I will cease causing CPU utilisation in it (last term of the equation). Obviously, the destination node also experiments a variation of the delivered performance, but this is accounted by the second term of the equation. This term is calculated using the previously exposed formulas (Eqs. (41) and (42)), by taking into account the workload type executed inside the VM.

$$\widehat{Perf}_{infr} | VM_I migr_{src}^{dst} = \sum_{\substack{i \in nodes \\ i \neq src}} \widehat{Perf}_{node_i} + \widehat{Perf}_{VM_I} | migr_{src}^{dst} \\ + \sum_{\substack{j \in VMs_{src} \\ j \neq I}} \frac{\widehat{U}_{VM_j^{src}}}{100} \cdot Perf_{max_{src}}. \quad (43)$$

A VM migration can be understood as a VM which ceases its execution in its original node src (VM cancellation) and starts being executed in another node (VM deployment), in terms of power consumption. This reasoning is followed in Eq. (44), where the

two first terms are the same as in Eq. (39) (representing a VM cancellation in its source node), and the third term represents the additional energy consumption incurred by the VM once migrated. This last term can be calculated very similarly to the last term of Eq. (24), representing a VM deployment to a known destination node, but in this case the performance delivered to the VM in that node will be $\widehat{Perf}_{VM_l} | migr_{src}^{dst}$ (calculated as in Eqs. (41) or (42)).

$$\begin{aligned} \widehat{Pwr}_{infr} | VM_l migr_{src}^{dst} &= \sum_{\substack{i \in nodes \\ i \neq src}} \widehat{Pwr}_{node_i} + \left(\widehat{R}_{src} \left(\widehat{U}_0^{src} + \sum_{\substack{j \in VMs_{src} \\ j \neq l}} \widehat{U}_{VM_j^{src}} \right) \right. \\ &+ \left. \widehat{R}_{dst} \left(\widehat{U}_0^{dst} + \sum_{j \in VMs_{dst}} \widehat{U}_{VM_j^{dst}} + 100 \cdot \frac{\widehat{Perf}_{VM_l} | migr_{src}^{dst}}{\widehat{Perf}_{max_{dst}}} \right) \right) \\ &\cdot PUE. \end{aligned} \quad (44)$$

Eq. (45) evaluates the forecasted energy efficiency at infrastructure level for a potential VM l migration to a specific destination node.

$$\widehat{EnEf}_{infr} | VM_l migr_{src}^{dst} = \frac{\widehat{Perf}_{infr} | VM_l migr_{src}^{dst}}{\widehat{Pwr}_{infr} | VM_l migr_{src}^{dst}}. \quad (45)$$

3.2.6. VM Migration (unknown placement)

Energy efficiency forecasts at infrastructure level when VMs are migrated between nodes are needed by placement optimisation scheduling policies. For instance, the placement algorithm might be interested in consolidating a VM which is being executed alone in a server. If that VM is migrated, the source node can be turned off in order to avoid consuming unnecessary idle power. Another possible situation can be when a VM is demanding more resources than the ones which its hosting node can provide. In this case, it becomes necessary to migrate this VM to a node with more resources.

Initially, a rough estimation of the impact of this migration can be done by estimating it without selecting a specific destination node for the VM, as it will be studied in this subsection. Based on this initial evaluation, if the placement algorithm considers that it is worth performing this migration, it will then study in detail which is the best node to receive the migrated VM by performing the calculations described in Section 3.2.5.

Similarly to Section 3.2.2, given that the destination node is not known, we consider that the maximum received performance per CPU of the migrated VM will be the averaged performance per CPU of the datacenter, that is, $\overline{Perf}_{CPU}$ from Eq. (27). Thus, Eqs. (41) and (42) can be rewritten as Eqs. (46) and (47) respectively, and express a rough estimate of the performance that the VM will receive once migrated if they execute batch or interactive workloads.

$$\widehat{Perf}_{VM_l} | migr_{src}^{unk} \wedge VM_l \in B = \overline{Perf}_{CPU} \cdot CPU_{VM_l} \quad (46)$$

$$\widehat{Perf}_{VM_l} | migr_{src}^{unk} \wedge VM_l \in I = \begin{cases} \text{if } \left[\frac{\widehat{Perf}_{max_{src}}}{CPU_{node_{src}}} \geq \overline{Perf}_{CPU} \right] \\ \quad \vee \left[\widehat{Perf}_{VM_l} | l \in VMs_{src} \leq \frac{\widehat{Perf}_{max_{src}}}{CPU_{node_{src}}} \cdot CPU_{VM_l} \right] : \\ \quad \min \left(\widehat{Perf}_{VM_l} | l \in VMs_{src}, \overline{Perf}_{CPU} \cdot CPU_{VM_l} \right) \\ \text{else:} \\ \quad \frac{\widehat{Perf}_{max_{src}}}{CPU_{node_{src}}} \cdot CPU_{VM_l} \leq \widehat{Perf}_{VM_l} | l \in VMs_{dst} \\ \quad \leq \overline{Perf}_{CPU} \cdot CPU_{VM_l}. \end{cases} \quad (47)$$

Using these equations, the impact on the infrastructure delivered performance can be expressed as in Eq. (48).

$$\begin{aligned} \widehat{Perf}_{infr} | VM_l migr_{src}^{unk} &= \sum_{\substack{i \in nodes \\ i \neq src}} \widehat{Perf}_{node_i} + \widehat{Perf}_{VM_l} | migr_{src}^{unk} \\ &+ \sum_{\substack{j \in VMs_{src} \\ j \neq l}} \frac{\widehat{U}_{VM_j^{src}}}{100} \cdot \widehat{Perf}_{max_{src}}. \end{aligned} \quad (48)$$

We follow a similar reasoning as in Section 3.2.5 to calculate the impact on the infrastructure power consumption for a VM migration whose destination node is unknown. The complete formula is provided in Eq. (49). A VM migration can be understood as a VM cancellation (first two terms of the formula) followed by a VM deployment (last term). In this case, as the destination node for the VM being migrated is not known, we use the calculations exposed in Section 3.2.2 to calculate the extra power incurred by the VM once deployed to an unknown destination node. Note that in this case, the term $CPU_{depl.} = 0$, since no additional CPUs are used because of the migration. Also observe that this calculation is valid when a linear power to CPU utilisation model is used.

$$\begin{aligned} \widehat{Pwr}_{infr} | VM_l migr_{src}^{unk} &= \sum_{\substack{i \in nodes \\ i \neq src}} \widehat{Pwr}_{node_i} + \left(\widehat{R}_{src} \left(\widehat{U}_0^{src} + \sum_{\substack{j \in VMs_{src} \\ j \neq l}} \widehat{U}_{VM_j^{src}} \right) \right. \\ &+ \left. CPU_{VM_l} \cdot \left(\frac{\sum_{i \in nodes} \widehat{Pwr}_{min_i}}{\sum_{i \in nodes} CPU_{alloc_i}} + \overline{Pwr}_{incr} \cdot \frac{\widehat{Perf}_{VM_l} | migr_{src}^{unk}}{CPU_{VM_l} \cdot \widehat{Perf}_{CPU}} \right) \right) \\ &\cdot PUE. \end{aligned} \quad (49)$$

Following the general formula of energy efficiency described in Eq. (4) and taking into consideration the aspects described in this subsection, the forecasted energy efficiency at infrastructure level for a potential VM l migration is presented in Eq. (50).

$$\widehat{EnEf}_{infr} | VM_l migr_{src}^{unk} = \frac{\widehat{Perf}_{infr} | VM_l migr_{src}^{unk}}{\widehat{Pwr}_{infr} | VM_l migr_{src}^{unk}}. \quad (50)$$

4. CPU estimator

To forecast the eco-efficiency at the different levels described in this document, the CPU utilisations of the VMs running in the infrastructure need to be forecasted separately, as well as that of the privileged domain of each host (Domain-0 in the Xen nomenclature). To this end, a general variable estimator has been developed. It relies on forecasts provided by four different predictors and selects one of them appropriately to obtain the “best forecast” of a given variable. The overall design of the variable estimator is presented in Fig. 3.

As it can be observed in the figure, the variable estimator consists of four different predictors: the moving average, the exponential smoothing, the linear regression, and the double exponential smoothing predictors. A very brief explanation on how these predictors perform the forecasts is provided in Sections 4.1–4.4. Further detailed information about these predictors can be found in [27]. All of them individually take the past and present observed values of the variable to predict (from time $t - n$ until t) in its input and provide a forecast of its expected value in time $t + \tau$ in its output, where τ is the time difference between the current time and the future instant to predict. Note that each estimator can be configured with one or more configuration parameters. In the experimentation subsection, these parameters will be appropriately tuned in order to obtain a better precision when forecasting the CPU utilisation of a given virtual machine.

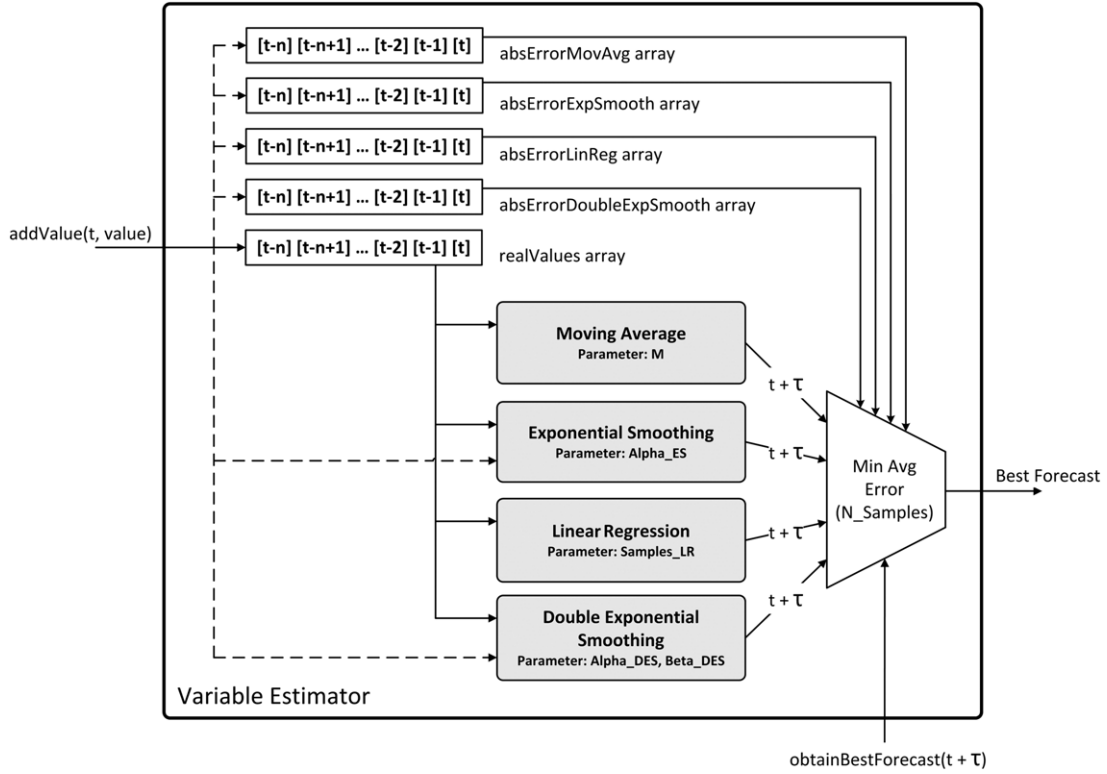


Fig. 3. Variable estimator overall design.

The variable estimator provides two interfaces: the *addValue(t, value)* and the *obtainBestForecast(t + tau)* methods. When the *addValue(t, value)* method is invoked, a new sample of the variable is inserted at the last position of the *realValues* array, corresponding to time *t*. The already existing samples are shifted to the left. If the *realValues* array's size was already *n* (a discussion of the value of *n* is provided in short), the oldest sample is simply discarded.

In addition, *addValue* also updates the absolute error arrays (also of size *n*) of the different estimators. In particular, it calculates the absolute error that each estimator would have if the sample at the current time (time = *t*) was predicted using the samples of the *realValues* array until time = *t* - 1. This calculated value is inserted in the last position of each absolute error array, displacing the previous absolute errors to the left and discarding the oldest one if the array's size was already *n*.

Finally, *addValue* also updates the state of the exponential and the double exponential smoothing predictors taking into consideration the newly inserted value.

The *obtainBestForecast(t + tau)* method firstly evaluates, for all the predictors, which has had the minimum mean absolute error during the last *N_Samples* predictions. To do this, it takes all the absolute error arrays of all the predictors and calculates its average values over the last *N_Samples* samples for each of them. Depending on the result, it selects the forecast provided by the predictor with the smallest mean average absolute error over the last *N_Samples* as the "Best Forecast" for time *t + tau*.

The value of *N_Samples* is configurable and might affect to the precision of the predictor. In the experimentation subsection, the value of *N_Samples* which maximises the precision of the variable estimator when predicting the CPU utilisation of a VM will be determined.

All the absolute error and real values arrays have a size of *n* samples. The value of *n* has to be at least *Samples_LR* for the Linear Regression predictor to be able to work correctly, and it also has to be at least equal to *N_Samples* so the last *N_Samples* samples of the

absolute error can be averaged when selecting the predictor with the smallest mean average error. Therefore, *n* should be at least $\max(\text{Samples_LR}, N_Samples)$. To simplify the implementation of the variable estimator, the same size *n* has been chosen both for the real values and the absolute error arrays.

4.1. Moving average predictor

The Moving Average predictor estimates the future value of a variable by averaging its last *M* observations, as it assumes that the mean value of the variable changes through time:

$$\hat{y}_{t+\tau} = \frac{\sum_{i=t-M+1}^t y_i}{M}. \quad (51)$$

4.2. Exponential smoothing predictor

The exponential smoothing predictor calculates the forecasted value of a variable by performing a weighted average of its last observed value and the previous forecast. So, the *Alpha_ES* parameter determines how much the predictor relies on the last observed sample or in the previous forecast to do the prediction.

$$\hat{y}_{t+\tau} = \text{Alpha_ES} y_t + (1 - \text{Alpha_ES}) \hat{y}_{t-1}. \quad (52)$$

4.3. Linear regression predictor

The linear regression predictor calculates the line which minimises the sum of squared distances between a set of *Samples_LR* previously observed samples of the variable and the line. Using this line and assuming a model comprised of a linear trend plus a constant, it is able to predict future samples of the variable:

$$\hat{y}_{t+\tau} = M\tau + C, \quad \text{where } \tau \geq 0 \quad (53)$$

and

$$M = \frac{m \sum_{i=t-m+1}^t x_i y_i - \sum_{i=t-m+1}^t x_i \sum_{i=t-m+1}^t y_i}{m \sum_{i=t-m+1}^t x_i^2 - \left(\sum_{i=t-m+1}^t x_i \right)^2}$$

$$C = \frac{\sum_{i=t-m+1}^t y_i}{m} - M \frac{\sum_{i=t-m+1}^t x_i}{m}.$$

To ease visualisation, the parameter m has been used. However, given that it could be confused with the M parameter of the Moving Average predictor, we use *Samples_LR* to refer to m . Also note that in this model, the variable x is actually the value of the time stamp at the moment that each sample was taken.

4.4. Double exponential smoothing predictor

The double exponential smoothing predictor is based on a model consisting of a linear and a constant trend. In this case, the parameters are calculated by taking into consideration the last observed value of the variable, as well as the previous value of the parameters themselves:

$$\hat{y}_{t+\tau} = \hat{a}_t + \hat{b}_t \tau, \quad \text{where } \tau \geq 0 \quad (54)$$

and

$$\begin{aligned} \hat{a}_t &= \text{Alpha_DES} y_t + (1 - \text{Alpha_DES})(\hat{a}_{t-1} + \hat{b}_{t-1}) \\ \hat{b}_t &= \text{Beta_DES}(\hat{a}_t - \hat{a}_{t-1}) + (1 - \text{Beta_DES})\hat{b}_{t-1}. \end{aligned}$$

The double exponential smoothing is configured by means of two parameters: *Alpha_DES* and *Beta_DES*. For simplicity, the different parameter values have been calculated using Eq. (55) (as in [28]), by increasing the value of δ from 0.0 to 1.0 in steps of 0.1. Note that these parameters do not necessarily need to be related in this way.

$$\text{Alpha_DES} = 1 - (1 - \delta)^2 \quad \text{Beta_DES} = \frac{\delta^2}{1 - (1 - \delta)^2}. \quad (55)$$

5. Experimentation and validation

Many mathematical models have been deduced in the context of this work. Of all of them, it was considered that the assessment and forecast of the energy efficiency of the whole infrastructure (see Section 3.1.2) was the most important one. On the one hand, it is the factor that IaaS providers want to optimise. On the other hand, note that this calculation intrinsically validates the assessment and forecast at node level, as it internally uses the same formulas, as well as VM and consequently service levels, because they derive from the node-level formulations. Also consider that by evaluating the forecasts at infrastructure level we are validating the capability of our system to perform accurate forecasts of the CPU utilisation of VMs, which is used to calculate the performance and power consumption for all the levels. Potential forecasts are also based on the forecasts of CPU utilisation and at infrastructure level. According to this, good accuracy in making those forecasts will ensure that potential forecasts can be also predicted accurately, if we assume that the presented formulation represents adequately the corresponding actions.

Consequently, the conducted experimentation aims to evaluate and to improve the precision of the forecasts of the energy efficiency of the infrastructure when running different workload types. To this end, the parameters of the variable estimators used to forecast the CPU utilisation of all the VMs running in the infrastructure were appropriately tuned for each workload type. Refer to Sections 4.1–4.4 for a quick overview of such parameters.

As commented in Section 2.1, this publication extends and improves the work regarding energy efficiency assessment and

prediction presented in [10]. During the experimentation of this section, we have used the same testbed as in the aforementioned article. The reader will be able to find there the details regarding the monitoring infrastructure used to capture the information about CPU utilisation and energy consumption. As shown in [10] (see Fig. 7), the Energy collector (which obtains the power measurements from the power sensor) only causes a 0.40% of CPU utilisation, while the Virtual collector (responsible of capturing the CPU utilisation metrics) incurs less than 0.05% of CPU utilisation. Therefore, the intrusiveness and associated power consumption of the monitoring mechanism can be considered as negligible. The response time of the Energy collector was measured to be around 3 s. Since the power values do not change very fast and considering that corrective actions can be taken by the VM placement algorithm in soft real-time manner in this time window, we consider this delay to be sufficient for our needs. In order to avoid causing interferences in the VMs, the Ecoefficiency Tool was placed in an external node, as well as the database from where it retrieved the CPU utilisation and energy information. The node where these were placed has not been considered in the infrastructure energy efficiency calculations.

In Sections 5.1 and 5.2, the respective procedures followed to generate realistic web and batch workloads are explained in detail. Then, in Sections 5.3 and 5.4, the Ecoefficiency Tool precision is evaluated against these different workloads, its predictor parameters are appropriately tuned and the obtained precision results are presented.

5.1. Methodology to generate web workload

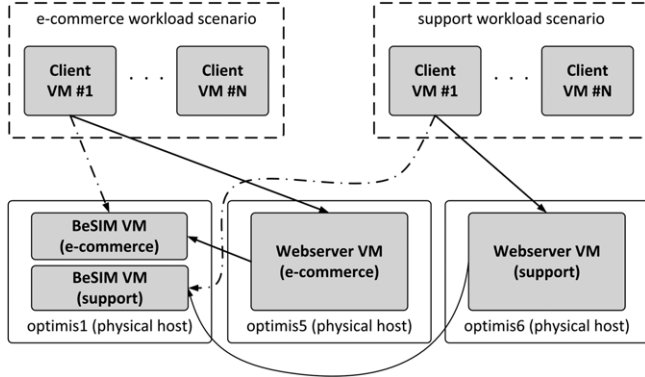
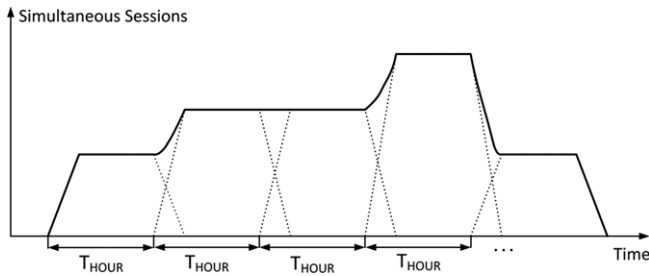
A realistic web workload has been generated based on the SPECweb2005 benchmark [29]. SPECweb2005 is a benchmark developed by the Standard Performance Evaluation Corporation (SPEC), widely used to evaluate the performance of a system acting as a web server servicing static and dynamic page requests. It mimics the behaviour of real Internet users under different scenarios: a banking site, an e-commerce site and a support site, based on the client's behaviour when accessing different web pages stored in a web server and the responses it gets. In the conducted experiments, both the e-commerce as well as the support sites have been used.

The SPECweb2005 benchmark components have been encapsulated in three different Virtual Machine types: *Webserver VM*, *BeSIM VM* and *Client VM*. Each of them encapsulates different SPECweb2005 components. Refer to [24] for more details. In a nutshell, the *Webserver VM* acts as a web server, the *BeSIM VM* emulates the logic of an application and database server and the *Client VM* emulates multiple clients accessing the web server. The number of simultaneous sessions that each *Client VM* generates can be independently established, and the Web server and BeSIM VM web servers have been configured following the tune up recommendations published in [30]. Note that the web server attending the clients (inside the *Webserver VM*) needs to be encapsulated in a VM in order to be considered as “useful work” by the infrastructure provider.

One *Webserver VM* and one *BeSIM VM* are hosted in the Cloud provider per each deployed service. That is, a *Webserver VM* and a *BeSIM VM* provide the required service to access the e-commerce website, while another pair provide access to the support website. These four VMs cause a realistic load in the infrastructure provider resources, which is measured in order to assess the energy efficiency of the provider's infrastructure. The *Client VMs* for each scenario are deployed in an external set of nodes (not accounted in the energy efficiency calculation), and will be the ones generating load to the system.

Table 2
Node characteristics.

	Optimis 1	Optimis 2	Optimis 5	Optimis 6
Number of processors	4	2	4	4
Processor type	Intel Xeon X3430	Intel Pentium G6950	Intel Xeon X3330	Intel Xeon X3330
Processor frequency (GHz)	2.40	2.80	2.66	2.66
RAM Memory (GB)	10	8	7	7
Operating System	Debian 6.0.7	Debian 6.0.7	Debian 6.0.7	Debian 6.0.7
Hypervisor	Xen 4.0.1	Xen 4.0.1	Xen 4.0.1	Xen 4.0.1
Pwr_{min} (W)	60.119	57.249	75.777	70.67
Pwr_{max} (W)	125.569	57.249	151.947	144.51
$Perf_{max}$ (MWIPS)	2171.6	1215.4	2173.96	2179.84

**Fig. 4.** Virtual Machine distribution (web workload experimentation).**Fig. 5.** Variable web workload generation by overlapping individual SPECweb2005 benchmark executions.

The VM distribution of the conducted experiment is presented in Fig. 4. The solid arrows stand for direct communication throughout the experiment, while the dashed ones depict the messages sent during the configuration stage of each benchmark execution.

The physical nodes where the aforementioned VMs were deployed have the characteristics described in Table 2.

The number of simultaneous sessions is the main factor influencing in a web server's resource utilisation. By default, this parameter is constant during an execution of the SPECweb2005 benchmark. However, to really mimic the workload of real Internet sites, this parameter needs to be modified along the execution of the experiment. To achieve this, the SPECweb2005 benchmark is executed multiple times (for each of the scenarios), in a fashion as the one displayed in Fig. 5. As it can be observed, an independent SPECweb2005 benchmark is launched in each iteration and its execution starts precisely when the previous benchmark run starts reducing the number of client threads connected to the web server. Consequently, the global number of simultaneous sessions changes smoothly from one value to the other across different iterations.

The web traces of the Barcelona School of Informatics (FIB) web page¹ have been used as a reference to set the number of

simultaneous sessions at a given time during the experiments (corresponding to the different hours captured during Monday 5th and Tuesday 6th April 2010 in the web traces). Note that the simulated time is different from the execution time of the experiment. A “real-world” hour is simulated during T_{HOUR} seconds ($T_{\text{HOUR}} = 480 \text{ s} = 8 \text{ min}$ in the conducted experimentation). After T_{HOUR} seconds, the experiment changes the number of simultaneous sessions based on the next value of received requests per hour of the FIB web traces. Also observe that if a forecast is performed (either CPU or energy efficiency forecast) for 1 min (of the experiment execution time) ahead in the future, this actually represents $\frac{60}{8} = 7.5 \text{ min}$ of the simulated time.

In the conducted experiments, it was determined that a maximum of 200 simultaneous sessions could be handled safely (without becoming overloaded) by a *Client VM*. So, in each iteration of the experiment, more than one *Client VM* may need to be created in order to generate the desired simultaneous sessions for that period.

5.2. Methodology to generate batch workload

To generate a realistic batch workload, the COMP Superscalar (COMPSS) Programming Model [31,32] has been used. COMPSS is a programming model which facilitates the development of applications that run on top of distributed infrastructures such as Clusters, Grids and Clouds. In a nutshell, COMPSS parallelises an application sequential code by splitting it into parts (named tasks) which can have data dependencies among them. The COMPSS runtime system exploits the inherent parallelism of such tasks at execution time and schedules their execution to the available remote resources.

In a COMPSS environment, two main entities are present: a master and several workers. So, in order to be considered as “useful work” by the infrastructure provider, the master and the workers have been encapsulated in two different VM kinds: *Master VM* and multiple *Worker VMs*.

Although the COMPSS PM was not originally conceived to mimic a batch workload, an application which generates job arrivals (consisting on multiple tasks) according to a Poisson process has been developed and makes it work as such. The batch workload generator application and the main part of the COMPSS environment components are placed in the *Master VM*, while the *Worker VMs* are in charge of executing the tasks issued in the *Master VM* and assigned to them. The *Worker VMs* are the ones actually causing a high resource utilisation in the Cloud infrastructure during the experimentation, given that the tasks they execute are CPU intensive.

The deployed scenario consists of a single *Master VM* which is in charge of controlling and scheduling tasks to 6 *Worker VMs*, as displayed in Fig. 6 (the node characteristics are the same as in Table 2):

When a new job is launched by the application in the *Master VM*, several tasks are submitted to the COMPSS PM, which assigns its execution to the different slots present in the *Worker VMs*. In general, each *Worker VM* has p processors that can execute p

¹ <http://www.fib.upc.edu>.

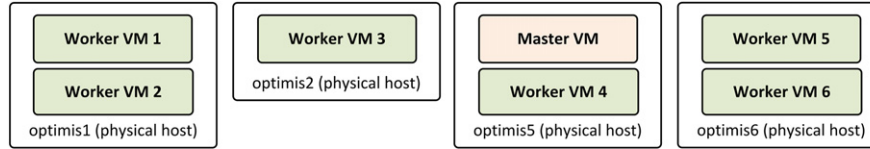


Fig. 6. Virtual Machine distribution (batch workload experimentation).

tasks simultaneously and independently. When a job arrives at the COMPSs PM runtime its tasks are automatically assigned to Worker VMs with at least one empty slot, if there is any. However, if all the slots are already occupied, the unassigned tasks are retained in a *waiting task set*, until one of the slots becomes free and the COMPSs PM decides to execute one of these waiting tasks in it. Note that the waiting tasks are not necessarily picked up in FIFO order from the waiting task set. They are picked one by one and therefore executed in a batch fashion. However, for the stability analysis of this subsection, the waiting task set can be understood as a queue.

The application generates job arrivals according to a Poisson process with rate λ_{JOB} . Note that since each job consists of k tasks, $\lambda_{TASK} = k \cdot \lambda_{JOB}$. Assuming that the service times of each slot of the Worker VMs have an exponential distribution with parameter μ_{TASK} , the whole system can be modelled as a M/M/c system, following the Kendall notation [33].

The system has been analysed using basic queue theory. Details on this analysis can be found in [24]. It has been determined that the maximum job submission rate follows the formula $\lambda_{JOB} < \frac{p \cdot w}{k} \cdot \mu_{TASK}$, where p are the number of processors per Worker VM and w the number of Worker VMs.

After measuring the execution time of a job when the system was empty, it was determined that the serving rate of a task was of 0.1397 servings/min. With this information and using the previous equation, the maximum job arrival rate when 6 Worker VMs of 2 processors each were used to execute jobs consisting of 3 tasks each ($w = 6, p = 2, k = 3$) was of 0.5588 arrivals/min.

Having a constant job arrival date throughout a given day is not a realistic situation: users tend to submit more jobs while they are working in their offices, whereas fewer jobs are submitted during the night. The FIB web traces were used as a reference to mimic this situation. Taking into account the number of sessions of the web traces during these time periods during 5th and 6th April 2010 and considering the number of sessions during that week, the job arrival rate through the experiment in Section 5.4 was established. It has been scaled in such a way that its mean value corresponds with the 90% of the maximum theoretical arrival rate, to guarantee the system stability.

The obtained waveform is the one which has been used to establish the arrival rate at each “hour period”. Remember that T_{HOURL} seconds of the experiment execution represents an hour of the simulated experiment (refer to Section 5.1 for more details). Note that although during some periods the arrival rate will be greater than the theoretical maximum, this aspect will be compensated during night hours. As long as the mean arrival rate is below the theoretical maximum (the 90% reduction factor establishes an additional safety margin), the system will not become saturated.

5.3. Model validation with web workload

In the following subsections, the experiments performed to validate and improve the accuracy of the forecasts performed by the Eco-efficiency Tool when running a realistic web workload are presented. In particular, the target of this experimentation has been to improve the accuracy of the forecasts at the infrastructure level, which are the ones used by the placement algorithms of the infrastructure provider to make optimisation decisions.

The two scenarios (e-commerce and support) were deployed as displayed in Fig. 4. The conducted experiment consisted in assessing the energy efficiency of the whole infrastructure and

forecasting it one minute ahead in the future. Then, after one minute the energy efficiency was evaluated again and the obtained energy efficiency assessment could be compared with the forecast which had been made in the previous step. At the same time, the CPU utilisation of each VM was also monitored, as well as the forecasted values provided by the predictors of the variable estimators (see Section 4) associated to each of them. This capturing procedure was repeated during all the experiment, and finally the obtained set of assessment and forecast samples are compared by means of the CCC (refer to Section 2.1). Note that, in fact, each assessment has been compared with the forecast which was performed in the previous iteration of the experiment. The objective of the experimentation is to obtain a CCC as close of 1.0 as possible when comparing the energy efficiency assessments and forecasts at the infrastructure level.

To sum up, two kinds of assessments–forecasts pairs were captured throughout the experiment: these of the overall energy efficiency of the infrastructure and these of the CPU utilisation of each VM. For the CPU utilisation of a given VM, each assessment was compared with the forecasts provided by each of the internal predictors (moving average, exponential smoothing, linear regression and double exponential smoothing) as well as the “best forecast” provided by its associated variable estimator (refer to Section 4 for more details).

Note that in all the experiments, the energy efficiency of the infrastructure is assessed and forecasted by the Ecoefficiency Tool installed in the infrastructure provider. Nevertheless, in order to be able to determine the CPU forecasts made for each VM, by each internal predictor of its associated variable estimator (as well as the resulting “best forecast”) and for each possible configuration parameter of each predictor, an external probe was installed in the same system. This external probe received the values of the current CPU utilisations of all the running VMs, each time they were measured by the Ecoefficiency Tool. It also received a notification each time the Ecoefficiency Tool issued the forecast of the CPU utilisation of a given VM for a given time stamp in the future, therefore allowing the external probe to perform the same forecast for all the predictors of that VM, using multiple configuration parameters in each of them.

5.3.1. Experiment 1: predictors’ parameter tuning

In this first experiment, the workload described in Section 5.1 was used. The accuracy of each internal predictor of each variable estimator (see Section 4) was separately evaluated for different configuration parameter values (refer to Sections 4.1–4.4 for a quick overview of them), in order to obtain the value which provides the highest precision when the CPU utilisation of a VM is predicted when running a pure web workload. The CCC was used as a criterion to compare the precision provided by the different parameter values: the one providing the greatest CCC was considered to be the best one.

The CPU utilisations of the *Webserver* VMs of the e-commerce and the support scenarios were evaluated and forecasted continuously, and the CCC between the expected CPU utilisations and the real values was calculated for each of them, for each parameter value of each internal predictor. The *BeSIM* VMs were not considered in this calculation, given that they were support VMs not causing a significant resource utilisation.

The obtained results when the CPU utilisation is predicted by each of the internal predictors (moving average, exponential

smoothing, linear regression and double exponential smoothing predictors) can be found in Figs. 7–10, respectively. Note that these results are the averaged results of those obtained in the e-commerce and support Webserver VMs.

When the CPU utilisation is predicted using the moving average predictor (Fig. 7), a value of $M = 2$ maximises the mean obtained CCC.

On the other hand, when the forecasts are made using the exponential smoothing predictor (Fig. 8), it is maximised using a value of $\text{Alpha_ES} = 0.8$, although 0.7 almost provides the same result.

For the linear regression predictor (Fig. 9), a value of $\text{Samples_LR} = 11$ is the best option. Considering the energy efficiency assessments displayed in Fig. 11 below, it can be observed that its waveform is actually very noisy. So, by using such a high value, the linear regression predictor actually filters the noise of the last 11 observed CPU samples in order to predict the next one, which is consistent with the used web workload.

As displayed in Fig. 10, a value of $\text{Alpha_DES} = 0.36$ (with a corresponding Beta_DES of 0.1111) maximises the obtained CCC.

During this experiment the energy efficiency was also internally assessed and forecasted by the Ecoefficiency Tool, which was configured with the following initial parameter values for its CPU estimators: $M = 3$, $\text{Alpha_ES} = 0.2$, $\text{Samples_LR} = 10$, $\text{Alpha_DES} = 0.36$ and $\text{Beta_DES} = 0.1111$. The obtained CCC was of 82.30%. Fig. 11 displays how the energy efficiency forecasts closely follow the trend of the assessments over time quite well.

5.3.2. Experiment 2: last samples error tuning

The “best forecast” (see Section 4) of the CPU utilisation of the VMs running in the infrastructure is the one used by the Ecoefficiency Tool to calculate, among other things, the infrastructure's energy efficiency. So, a better precision when determining it may lead to an improvement in the overall precision of the infrastructure energy efficiency forecasts. The parameter which has direct influence when determining which internal predictor is chosen to provide the “best forecast” is $N_Samples$.

The experiment described in the previous subsection was repeated, but configuring the CPU estimators of the Ecoefficiency Tool so they used the optimal parameters for each predictor (determined in Experiment 1: $M = 2$, $\text{Alpha_ES} = 0.8$, $\text{Samples_LR} = 11$, $\text{Alpha_DES} = 0.36$ and $\text{Beta_DES} = 0.1111$). In this second experiment, the impact of obtaining the “best forecast” by taking into consideration a value for $N_Samples$ which ranged from 1 to 10 (both inclusive) was evaluated. The obtained results can be observed in Fig. 12. The results display the fact that the obtained precision is not highly affected by the value of $N_Samples$. Nevertheless, the one providing the highest CCC corresponds to $N_Samples = 8$, which will be the one used in the next experiment.

In this experiment the infrastructure energy efficiency was also assessed and forecasted internally by the Ecoefficiency Tool, which was now configured with the optimal predictor configuration values obtained in Experiment 1. This configuration modification was reflected in the obtained precision: the resulting CCC was of 84.95%. So, the use of the optimal parameters obtained in Experiment 1 led to a better overall precision. In this experiment, the optimal $N_Samples$ value had not been determined yet and it was set to 1.

Fig. 13 displays how the energy efficiency forecasts follow the evolution of the assessments closer than in the previous experiment. In particular, it can be observed how the peaks are predicted more accurately.

One interesting outcome of Experiment 2 was that, in fact, the precision of the exponential smoothing estimator with $\text{Alpha_ES} = 0.8$ happened to be better than the “best forecast” prediction, both in the support (CCC: 71.51%) and in the e-commerce (CCC: 83.05%) scenarios, for all the possible values of $N_Samples$. This fact suggests that the way in which the individual forecasts of the predictors of the CPU estimators are combined might not be the

optimal one. Further study on this aspect will be conducted as future work.

5.3.3. Experiment 3: final results

Finally, the experiment was repeated again with the same parameters as in the previous one but setting a value of $N_Samples = 8$ in the configuration of the CPU estimators of the Ecoefficiency Tool. Nevertheless, given that a low variation in the precision was observed in the last experiment when $N_Samples$ was modified, the external probe was configured to still perform the “best forecasts” for a value of $N_Samples$ ranging from 1 to 20, both inclusive.

Fig. 14 demonstrates that there is not a clear value for $N_Samples$ which provides the best precision, as the CCC is maximised for $N_Samples = 2$ and $N_Samples = 20$ in this experiment, while in the previous experiment it was maximised for $N_Samples = 8$. Therefore, as discussed at the end of Experiment 2, the way the forecasts of the individual predictors inside of the CPU estimator are combined need to be studied further.

However, with $N_Samples = 8$ the quality of the energy efficiency forecasts performed by the Ecoefficiency Tool clearly increased when compared to Experiment 2. In this experiment, the obtained CCC was of 90.35%. Although there is not a clear variation of the precision when modifying $N_Samples$, in both experiments it was observed that with $N_Samples = 1$, the precision was worse than when selecting $N_Samples = 2, 8$ or 20. This partly justifies that in this experiment, given that $N_Samples$ was not equal to 1 (which was used in Experiment 2), the precision increased. On the other hand, in this experiment the variance of the energy efficiency assessments was lower than in the previous experiments (see Fig. 15), which allowed the predictions performing better.

Finally, the precision of the Ecoefficiency Tool when performing forecasts for different time intervals in the future was measured. Note that these time intervals represent the experiment execution time and not the simulated time (refer to Section 5.1 for more details). The results can be visualised in Fig. 16. Clearly, the CCC degrades when the prediction is performed for more than one minute ahead in the future. In fact, it can be concluded that for web workload the precision remains stable for predictions up to 7 min in the future and degrades from 8 min predictions onwards. Note that because of the time scaling ($T_{\text{HOURL}} = 8$ min) performed when conducting this experiment, 7 min in the simulation corresponds to 52.5 min in the “real world”.

5.4. Model validation with batch workload

The Eco-efficiency Tool was also validated using a batch workload which was generated following the steps described in Section 5.2. The set of conducted experiments were the same ones as in Section 5.3.

The individual predictor's configuration parameters (see Section 4) which maximised the mean CCC when predicting the CPU utilisation of all the deployed VMs were determined in the first conducted experiment (Section 5.4.1). Then, the optimum number of absolute error samples to consider when calculating the “best forecast” of the CPU utilisation of a VM (refer to Section 4 for more details) was determined in the second experiment (Section 5.4.2). Finally, in the third experiment (Section 5.4.3) these parameter values were introduced in the Ecoefficiency Tool to evaluate if its prediction precision increased. Moreover, the degradation of the CCC when performing such forecasts for multiple time intervals in the future was determined.

5.4.1. Experiment 1: predictors' parameter tuning

The conducted experiment in this subsection was exactly the same as the one described in Section 5.3.1 but using a batch workload (see Section 5.2) instead.

In this case, the VMs causing high resource utilisations were the 6 Worker VMs (distributed as detailed in Section 5.2). The Master VM was not considered in the calculations because its

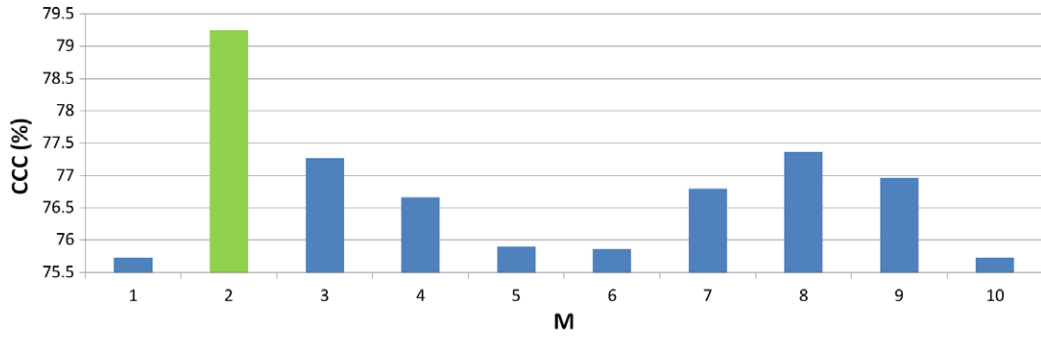


Fig. 7. Mean CCC (moving average predictor), Exp. 1 (Web).

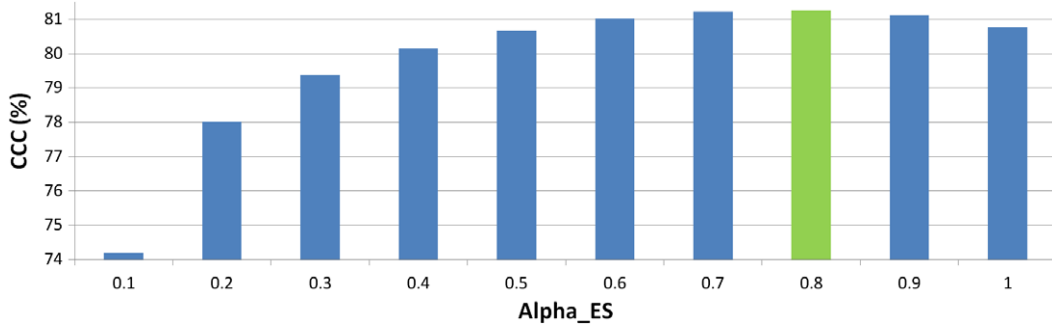


Fig. 8. Mean CCC (exponential smoothing predictor), Exp. 1 (Web).

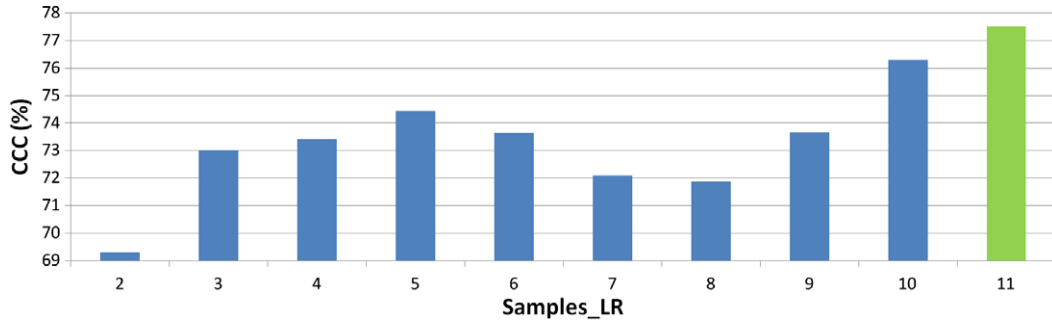


Fig. 9. Mean CCC (linear regression predictor), Exp. 1 (Web).

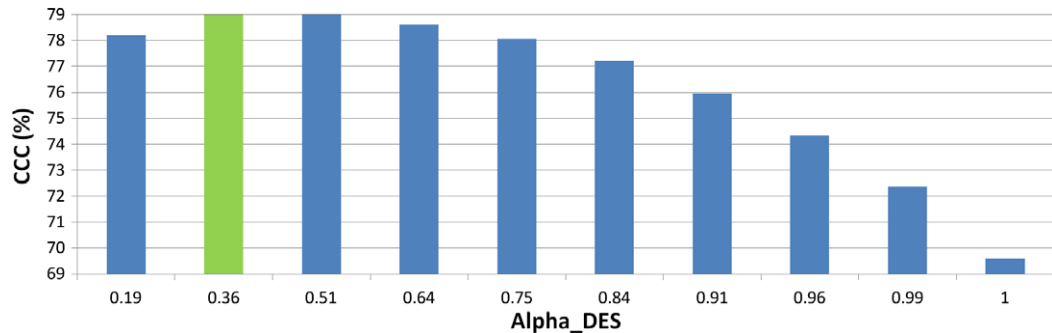


Fig. 10. Mean CCC (double exponential smoothing predictor), Exp. 1 (Web).

resource utilisation was negligible. The average of the individual CCCs (between the forecasted CPU utilisations and the real values) of all the *Worker VMs* was calculated for each parameter value and internal predictor.

Similarly to when a web workload was used, the results show that for a batch workload a value of $M = 2$ for the moving average predictors also leads to the highest CCC value (Fig. 17).

Nevertheless, for the exponential smoothing predictors the CCC becomes maximised for $\text{Alpha_ES} = 1$ (Fig. 18). This means that the best forecast when using this predictor type is actually the value

of the last observed sample. This discovery makes complete sense for a batch workload, as tasks tend to use the CPU where they are allocated (slot) at a constant 100% utilisation while they are executed, and not at intermediate utilisation levels as in the web workload. Therefore, the best estimation of CPU utilisation is the present one, as the task will keep using the whole CPU while it is being executed in a VM.

The linear regression predictors have the best average precision for a value of $\text{Samples_LR} = 4$ (Fig. 19), and not 11 as in the web workload. There are two reasons to explain this phenomenon. On

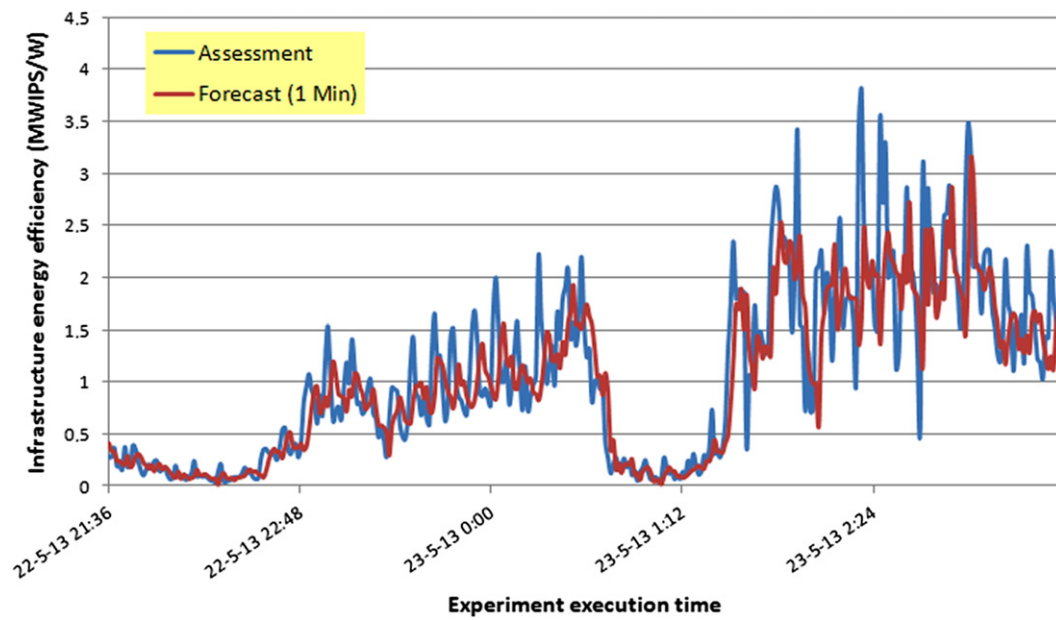


Fig. 11. Infr. energy efficiency assessments vs. forecasts, Exp. 1 (Web).

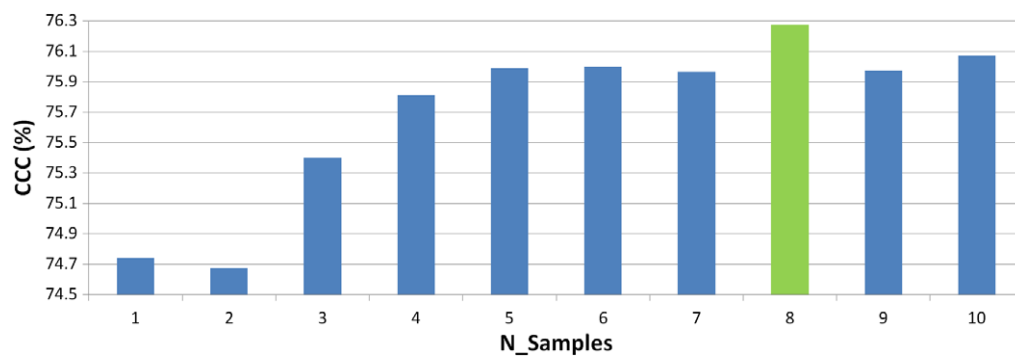


Fig. 12. Mean CCC (determines the "best forecast"), Exp. 2 (Web).

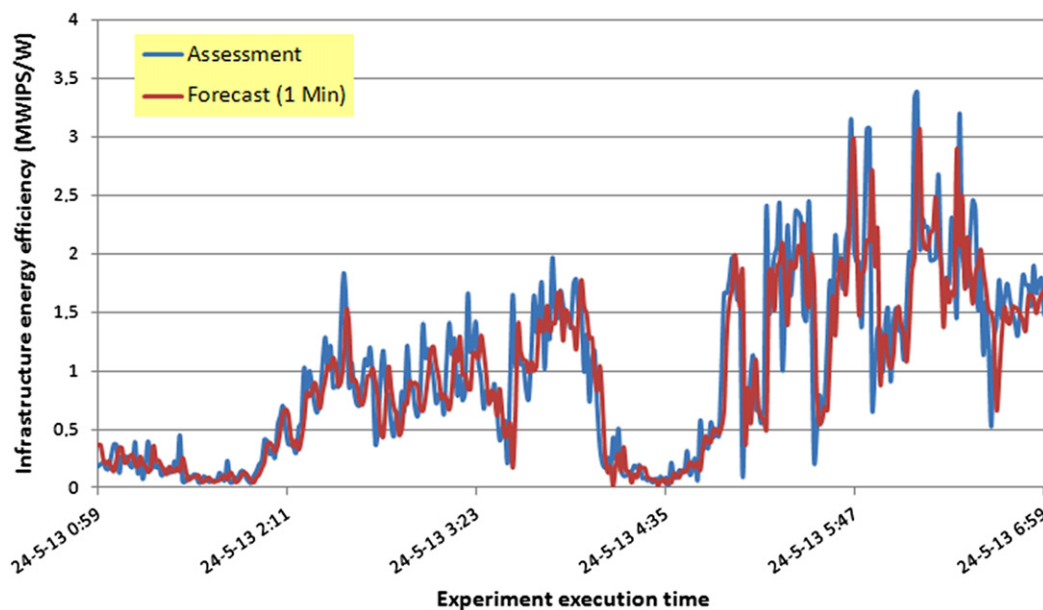


Fig. 13. Infr. energy efficiency assessments vs. forecasts, Exp. 2 (Web).

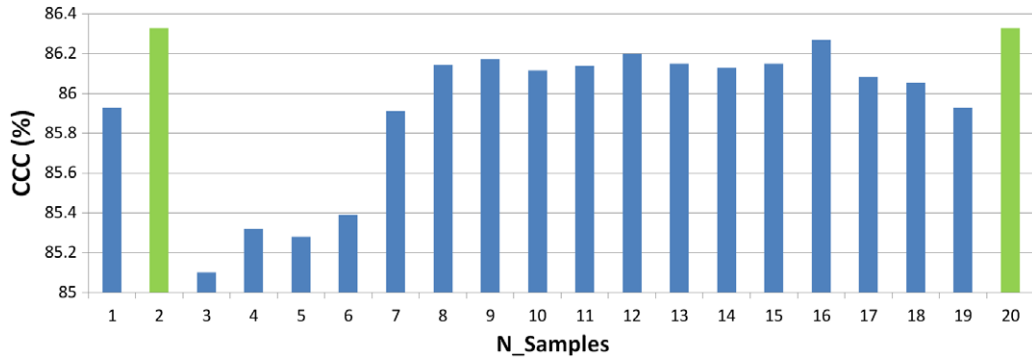


Fig. 14. Mean CCC (determines the “best forecast”), Exp. 3 (Web).

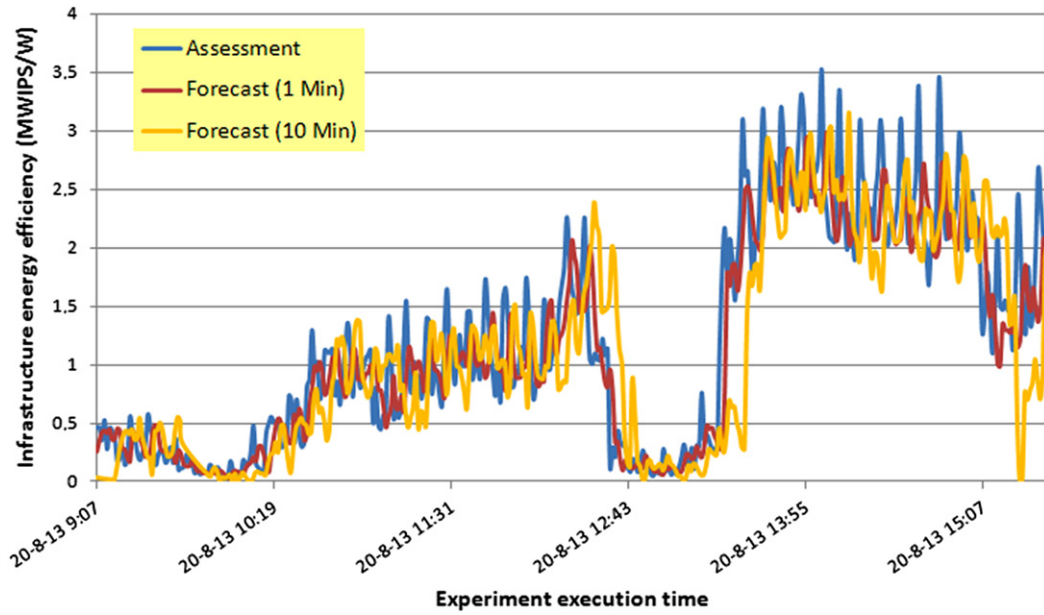


Fig. 15. Infr. energy efficiency assessments vs. forecasts, Exp. 3 (Web).

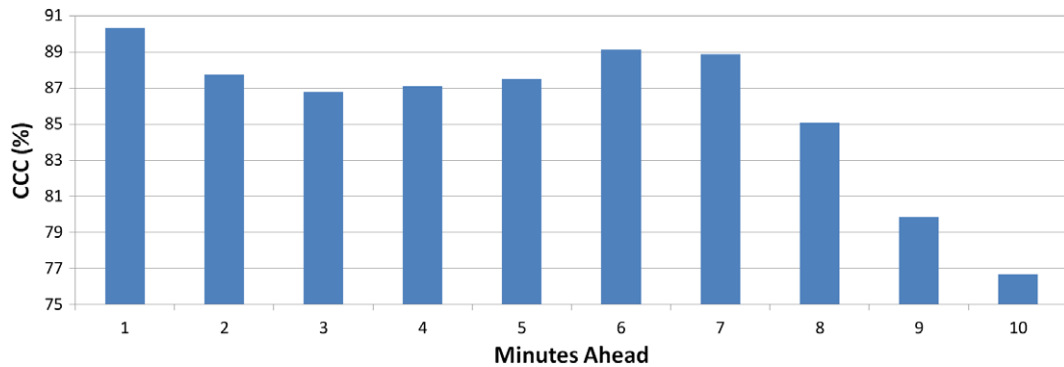


Fig. 16. CCC degradation of the infrastructure’s energy efficiency forecasts for different time interval predictions, Exp. 3 (Web).

the one hand, the CPU utilisation changes happen very abruptly when running a batch workload (when a task starts or stops being executed). So, the predictor needs to be very reactive when these changes happen. If a smaller value of *Samples_LR* is used, the predictor becomes adjusted to the new CPU utilisation values faster. On the other hand, given that when a task is running the CPU utilisation tends to be very stable, not many values need to be considered to predict the next one, as it will be very similar to

the current one (this fact has also been observed in the exponential smoothing predictor).

The double exponential smoothing predictor obtained the greatest CCC when $\alpha_{DES} = 0.84$, with a corresponding $\beta_{DES} = 0.43$ (see Fig. 20).

In this first experiment, the Ecoefficiency Tool was configured with its initial default configuration values ($M = 3$, $\alpha_{ES} = 0.2$, $Samples_LR = 10$, $\alpha_{DES} = 0.36$ and $\beta_{DES} = 0.1111$).

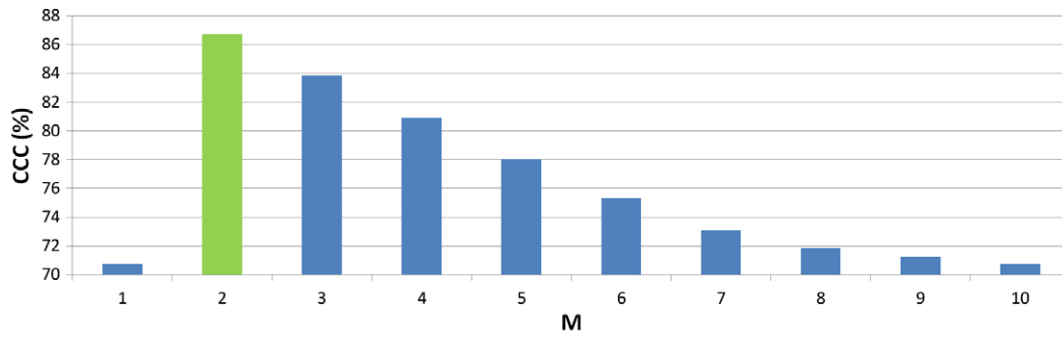


Fig. 17. Mean CCC (moving average predictor), Exp. 1 (Batch).

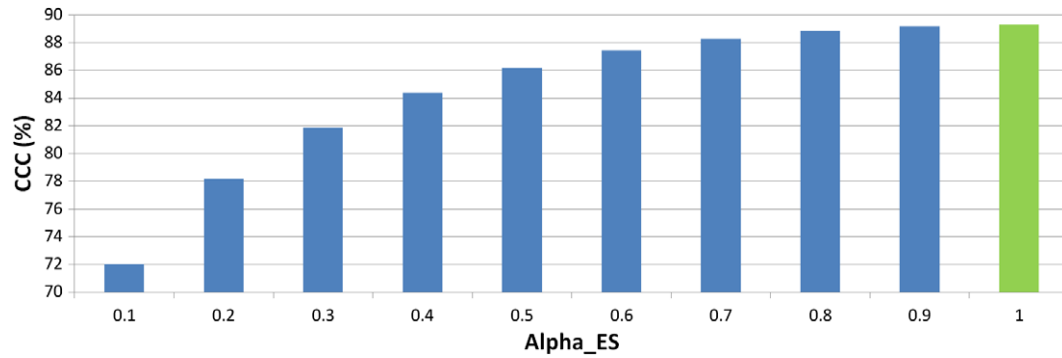


Fig. 18. Mean CCC (exponential smoothing predictor), Exp. 1 (Batch).

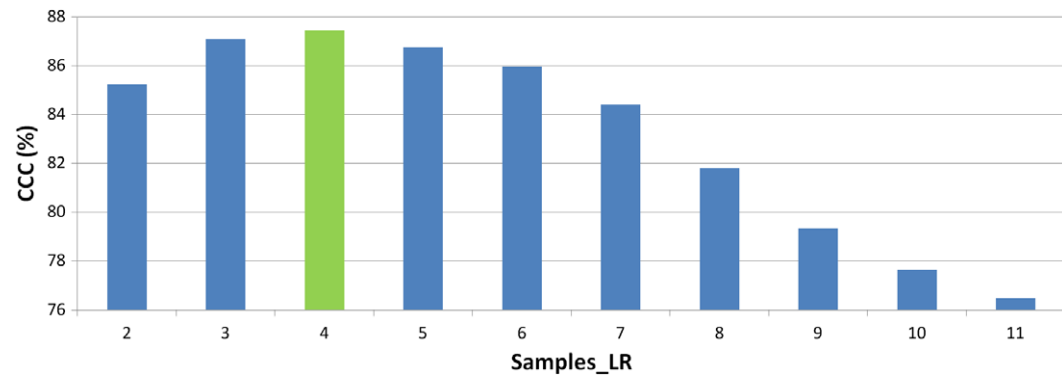


Fig. 19. Mean CCC (linear regression predictor), Exp. 1 (Batch).

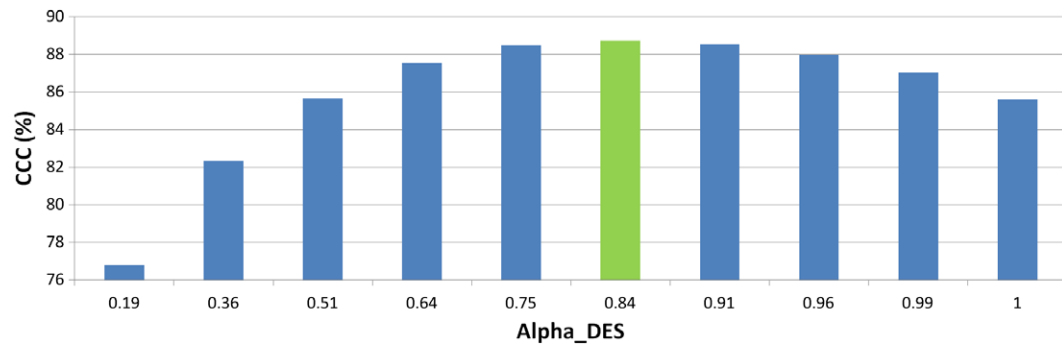


Fig. 20. Mean CCC (double exponential smoothing predictor), Exp. 1 (Batch).

In this case, the infrastructure energy efficiency forecasts (performed for one minute ahead in the future) obtained a CCC of 88.36%, and can be observed in Fig. 21. The objective of the following experiments will be to improve this result. Nevertheless, the obtained CCC is already very high.

5.4.2. Experiment 2: last samples error tuning

The conducted experiment in this subsection was exactly the same as the one described in Section 5.3.2 but using a batch workload (described in Section 5.2) instead. Refer to that subsection for more details.

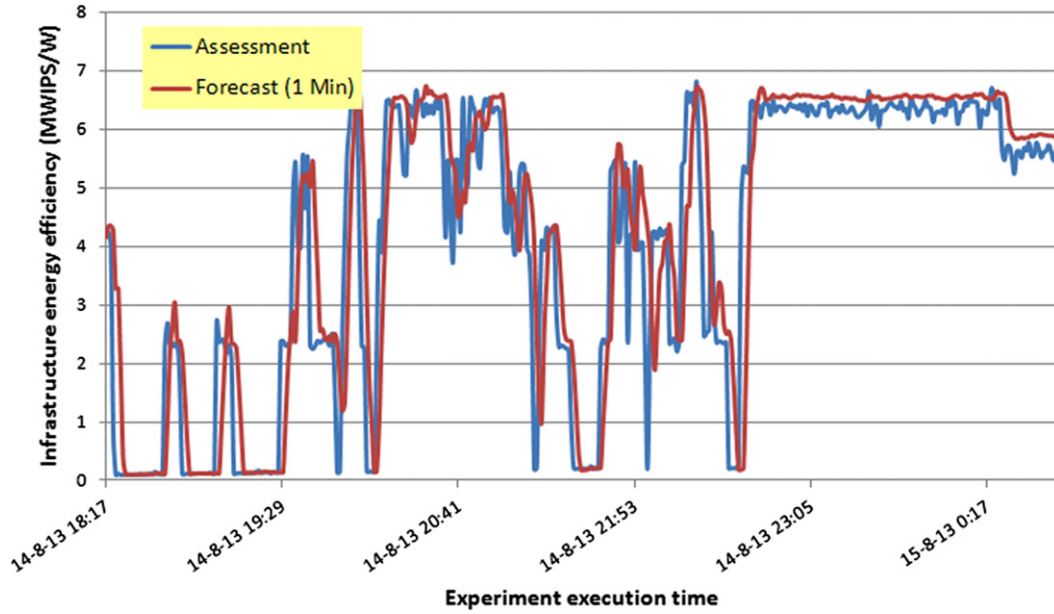


Fig. 21. Infr. energy efficiency assessments vs. forecasts, Exp. 1 (Batch).

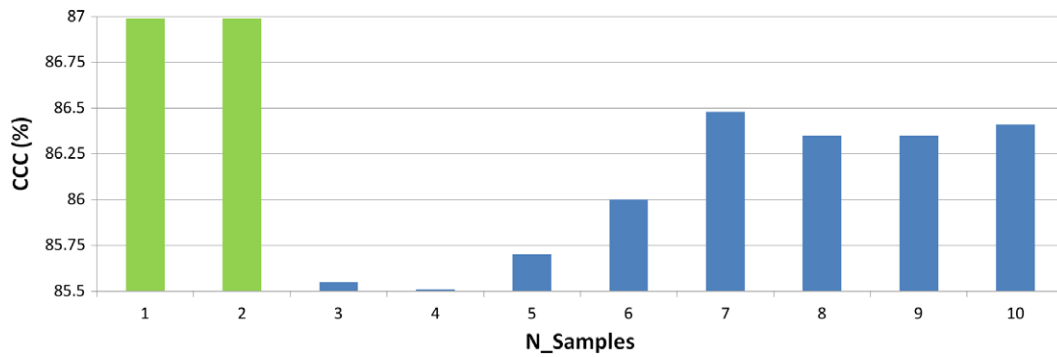


Fig. 22. Mean CCC (determines the “best forecast”), Exp. 2 (Batch).

After configuring the optimum parameter values obtained in Section 5.4.1 inside of the Ecoefficiency Tool, the previous experiment was repeated in order to determine the value of $N_Samples$ which provided the best precision for the “best forecasts” (see Section 4) of the VM CPU estimators. The obtained results are displayed in Fig. 22. In this case, the values of $N_Samples = 1$ and $N_Samples = 2$ were the ones maximising the CCC. The chosen value has been $N_Samples = 1$, since it makes the estimator more reactive and this is more appropriate when working with a batch workload.

There seems to be a big precision difference between using $N_Samples = 1$ or 2 instead of using $N_Samples = 3$ or 4. However, observe that such a behaviour can also be observed in Fig. 14. In that case, the difference of CCC between choosing $N_Samples = 2$ and 3 was of 1.2% approximately, while now it is of about 1.5%. This would suggest that using a value of $N_Samples = 2$ would be the best choice in both Web and Batch scenarios; meaning that a more reactive estimator when switching between the individual forecasts of the individual predictors provides the best precision in both cases. Nevertheless, consider that in Experiment 2 when using a Web workload, a value of $N_Samples = 8$ was initially considered to be the optimum one. As already argued before, a deeper study on the combination of the individual predictions needs to be therefore conducted.

The Ecoefficiency Tool was configured with the values obtained in the previous experiment ($M = 2$, $Alpha_ES = 1.0$, $Samples_LR =$

4, $Alpha_DES = 0.84$ and $Beta_DES = 0.43$). Note that, casually, the initial value of $N_Samples$ was actually the optimum one which was found during this experiment. The infrastructure energy efficiency forecasts (performed for one minute ahead in the future) obtained a CCC of 87.99%, and can be observed in Fig. 23.

In the previous experiment (without having tuned any parameter) the CCC was actually greater, 88.36%. If the two experiment's energy efficiency waveforms are compared (Figs. 21 and 23), it can be seen that in the first experiment there were some tasks being queued (as the energy efficiency looks like a “mountain” during the first simulated day) but not during the first simulated day of the second experiment. So, the CPU utilisation of the VMs during the second experiment performed more abrupt variations than in the first experiment, which was more stable. These variations are actually the hardest values to predict and therefore causing a decrease in the overall CCC.

5.4.3. Experiment 3: final results

In the previous experiment, it was determined that a value of $N_Samples = 1$ was the one providing the values for the “best forecasts” (refer to Section 4) that maximised the overall CCC of the CPU estimators. In fact, this value had already been set in the previous experiment, as it was the initial default value when the experiments were started. Therefore, it would reasonable to expect that the CCC for the energy efficiency forecasts of the infrastructure were similar to the previous experiment. In fact, this time the

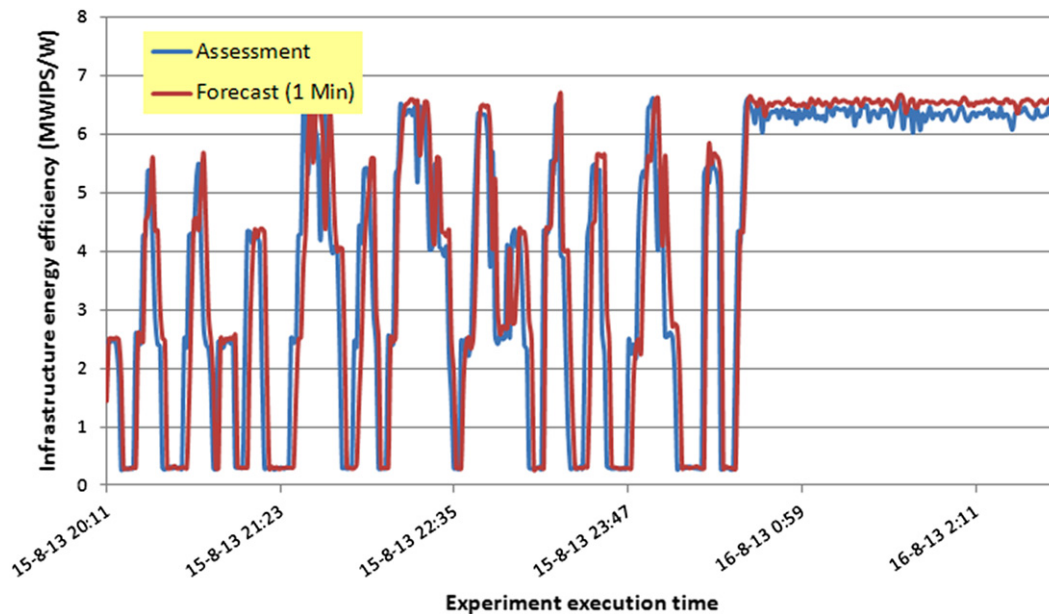


Fig. 23. Infr. energy efficiency assessments vs. forecasts, Exp. 2 (Batch).

obtained CCC for the infrastructure energy efficiency forecasts was greater: 91.35%. From the waveform displayed in Fig. 24, it can be observed that the energy efficiency did not suffer so many variations as in the previous experiment. This explains why the obtained CCC was greater in this one.

So, on average it can be concluded that the CCC is improved when the individual predictor configuration parameters are tuned, but it is highly dependent on the way the tasks are submitted to the COMPSs PM. If tasks are queued, the CPU utilisation is more stable, easier to predict and therefore leads to a greater CCC. However, if more CPU utilisation variations take place (due to new tasks arriving or dying when there are no tasks in the queue waiting to be executed), the CCC tends to decrease.

Finally, the degradation of the CCC as the energy efficiency of the infrastructure is forecasted for longer time spans in the future has been evaluated. The obtained results are depicted in Fig. 25. In this case, it can be observed how the degradation is more pronounced than when operating with web workload. Note that these time intervals represent the experiment execution time and not the simulated time (refer to Section 5.2 for more details).

Observe that while in the web workload the CCC decreased 10% when the energy efficiency was predicted 9 min ahead in the future, the same 10% degradation takes place when it is predicted for only 3 min in the future when a batch workload is used. This is explained by the fact that in batch workloads the energy efficiency can change very abruptly, from values around 5 MWIPS/W to 0 MWIPS/W (for instance, when a task ends its execution and there are no tasks waiting in the scheduler queue). Such variations do not take place when a web workload is used.

6. Conclusions and future work

6.1. Conclusions

IaaS providers have a great interest in optimising the overall energy efficiency of their infrastructure. This requires precise knowledge of the current and future energy efficiency at different levels (VM, node, infrastructure or service) in order to take the appropriate decisions (e.g. VM replication, migration, cancellation). The current available tools do not provide such information in real time. To tackle such limitation, mathematical models to calculate and forecast it at the different aforementioned levels have been provided

in Section 3.1 while the impact on the overall infrastructure energy efficiency of performing such actions has been addressed in Section 3.2.

A precise way to determine the future CPU utilisation of a VM is needed to perform all the previous forecasts. To this end, a variable estimator that can be used for CPU utilisation forecasting has been developed in Section 4.

The assessment and forecast of the infrastructure energy efficiency has been the most important model deduced in this publication, because it is what IaaS providers want to optimise and it forms the basis of other models, such as the forecasts for potential actions described in Section 3.2. So, the conducted experimentation has been focused in determining the configuration parameter values of the internal CPU estimators which provide the best precision when forecasting the energy efficiency of the infrastructure for different workload types.

In general, the forecasts' precision improves significantly when the individual predictor configuration parameters are properly tuned (see Section 4). However, the way the individual CPU utilisation forecasts of the different internal predictors are combined may not be the optimal one, as in some cases an individual predictor achieved a CCC higher than the one achieved by the "best forecasts".

Nevertheless, the obtained precision also depends on the conducted experiment run. If the workload causes numerous spontaneous variations in the resource utilisation, this reduces the overall achieved precision because the internal predictors are not able to anticipate such sudden changes. This effect can be observed when a batch workload is executed. On the one hand, if tasks are waiting to be executed in the scheduler queue, the execution is more "stable" and the overall achieved precision increases. On the other hand, if more CPU utilisation sudden changes take place (e.g. when a task starts or finishes its execution and no tasks are waiting in the queue) the overall precision decreases.

Web workloads tend to be on average more stable than batch workloads (they do not experience sudden CPU utilisation changes from 0% to 100% or vice versa). However, they are very noisy since their resource utilisation levels tend to oscillate around a given average value, depending on the number of simultaneous users accessing the web service. This fact causes that the obtained best configuration parameter values for the individual predictors mentioned in Section 4 make them less reactive and more focused on noise filtering.

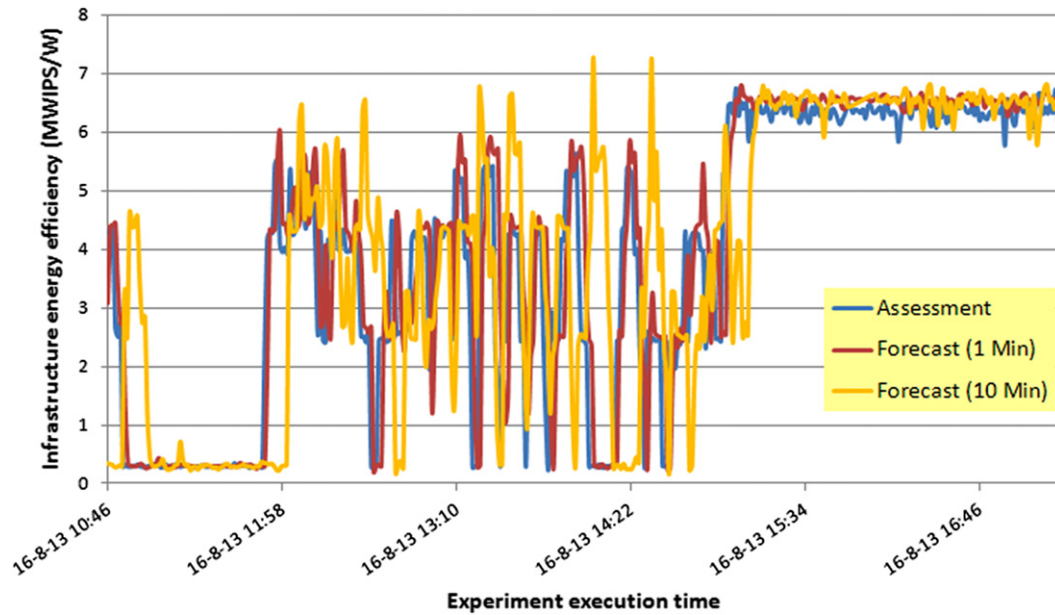


Fig. 24. Infr. energy efficiency assessments vs. forecasts, Exp. 3 (Batch).

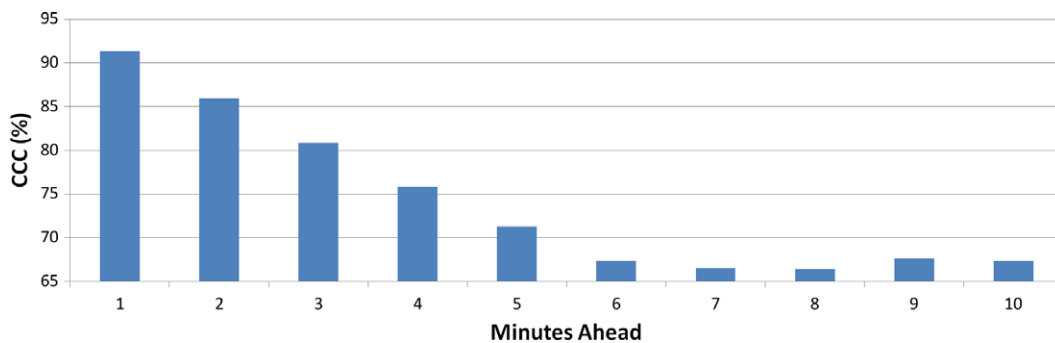


Fig. 25. CCC degradation of the infrastructure's energy efficiency forecasts for different time interval predictions, Exp. 3 (Batch).

Batch workloads behave differently and have been found to require more reactive predictors. They allow them to overcome sudden utilisation variations in a fast manner. Furthermore, given that during a task execution the CPU utilisation (of each CPU) remains stable around 100% and without much noise, despite the predictors being very reactive the obtained forecasts are still precise.

The forecasts' precision degradation as they were made for a longer period of time in the future has been evaluated. It has been observed that for web workloads, the precision remains quite stable until predictions up to 7 min in the future, and decreases by 10% for 8 min ones. The reason for this is that during the experimentation, each simulated hour (and hence associated load level) lasted only 8 min. Therefore, predictions up to that time interval were accurate as the number of simultaneous users accessing the web service remained more or less constant. For batch workloads the degradation was much faster: 10% degradation for 3 min predictions. This last result was also observed to be highly dependent on the execution, as it varied depending on the number of sudden resource utilisation variations during the conducted experiment.

6.2. Future work

The currently adopted performance metric, MWIPS, evaluates the performance of integer and floating-point arithmetic. It is focused on the performance being delivered by the CPU, but not in the one which could be measured considering the system's memory, hard disk or network card. So, the performance calculation could be extended by also considering such components.

A system power model which apart from CPU metrics also considers the memory, disk or network components could be used to replace the current one. Moreover, instead of considering a linear relationship between the captured metrics and the power consumption, other non-linear relationships such as polynomial or exponential could be considered.

The way the individual forecasts are combined in the developed CPU estimator may not be the optimal one, as in some cases the CCC of an individual predictor was higher than the one achieved by the 'best forecasts'. So, alternative ways to combine the information provided by the individual internal predictors should be studied. For instance, the individual forecasts could be averaged using weights based on the mean average error of each individual predictor.

It has been observed that sudden variations in the resource utilisation lead to an overall decrease in the forecasting precision. Therefore, a way to improve the current CPU estimator is to provide it with means to anticipate such spontaneous changes. For example, when running a batch or mixed workload, having information about the estimated remaining execution time of each task and the scheduler queue status could help in improving this aspect.

Each workload type requires different configuration parameters. A possible improvement could be to keep track, for each individual predictor, of the parameter value which would have maximised the CCC over a given number of CPU utilisation samples in the past. Based on this result, the configuration parameter value could be changed in real-time in order to adapt the predictors to the workload type of each individual VM.

Acknowledgements

This work has been supported by the OPTIMIS project (<http://www.optimis-project.eu/>) [9] and has been partly funded by the European Commission's IST activity of the 7th Framework Program under contract number 257115. This work expresses the opinions of the authors and not necessarily those of the European Commission. The European Commission is not liable for any use that may be made of the information contained in this work. It is also supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contract TIN2012-34557, and by the Generalitat de Catalunya under contract 2014-SGR-1051. The conducted experimentation has been possible thanks to the computing resources provided by the Umeå University, Sweden.

References

- [1] Richard Brown, et al., Report to Congress on Server and Data Center Energy Efficiency: Public Law 109-431, 2008.
- [2] Power Scorecard. Electricity from coal, 2000. URL: http://www.powerscorecard.org/tech_detail.cfm?resource_id=2.
- [3] Íñigo Goiri, Ryan Beauchea, Kien Le, Thu D. Nguyen, Md. E. Haque, Jordi Guitart, Jordi Torres, Ricardo Bianchini, Greenslot: scheduling energy consumption in green datacenters, in: 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, SC'11, ACM, Seattle, Washington, USA, ISBN: 978-1-4503-0771-0, 2011, pp. 20:1–20:11. 12–18 November 2011. <http://dx.doi.org/10.1145/2063384.2063411>.
- [4] San Murugesan, Harnessing green it: Principles and practices, IT Prof. 10 (1) (2008) 24–33.
- [5] Michael Grubb, Christiaan Vrolijk, Duncan Brack, Tim Forsyth, The Kyoto Protocol: a Guide and Assessment, Royal Institute of International Affairs, London, 1999.
- [6] Xiaobo Fan, Wolf-Dietrich Weber, Luiz Andre Barroso, Power provisioning for a warehouse-sized computer, in: 34th Annual International Symposium on Computer Architecture, ISCA'07, ACM, San Diego, California, USA, ISBN: 978-1-59593-706-3, 2007, pp. 13–23. 9–11 June 2007. <http://dx.doi.org/10.1145/1250662.1250665>.
- [7] Rajkumar Buyya, Anton Beloglazov, Jemal H. Abawajy, Energy-efficient management of data center resources for Cloud computing: A vision, architectural elements, and open challenges, in: International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA. 2010, 12–15 July 2010, Las Vegas, Nevada, USA, 2 Volumes, 2010, pp. 6–20.
- [8] Murray G Patterson, What is energy efficiency?: Concepts, indicators and methodological issues, Energy Policy (ISSN: 0301-4215) 24 (5) (1996) 377–390. [http://dx.doi.org/10.1016/0301-4215\(96\)00017-1](http://dx.doi.org/10.1016/0301-4215(96)00017-1).
- [9] Optimis – optimized infrastructure services, 07, 2013. URL: <http://optimis-project.eu>.
- [10] Gregory Katsaros, Josep Subirats, J. Oriol Fito, Jordi Guitart, Pierre Gilet, Daniel Espling, A service framework for energy-aware monitoring and VM management in Clouds, Future Gener. Comput. Syst. (ISSN: 0167-739X) 29 (8) (2013) 2077–2091. <http://dx.doi.org/10.1016/j.future.2012.12.006>.
- [11] Unixbench benchmark suite, 06, 2011. URL: <http://code.google.com/p/byte-unixbench/>.
- [12] Virt-top, 07, 2013. URL: <http://people.redhat.com/~rjones/virt-top/>.
- [13] M.K. Patterson, The effect of data center temperature on energy efficiency, in: ITHERM 2008. 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems, 28–31 May 2008, Lake Buena Vista, Florida, USA, 2008, pp. 1167–1174. <http://dx.doi.org/10.1109/ITHERM.2008.4544393>.
- [14] Ehsan Pakbaznia, Massoud Pedram, Minimizing data center cooling and server power costs, in: Proceedings of the 2009 ACM/IEEE International Symposium on Low Power Electronics and Design, ISLPED'09, ACM, New York, NY, USA, ISBN: 978-1-60558-684-7, 2009, pp. 145–150. <http://dx.doi.org/10.1145/1594233.1594268>.
- [15] Lei Ji, Kevin Gallo, An agreement coefficient for image comparison, Photogramm. Eng. Remote Sens. 72 (7) (2006) 823–833. <http://dx.doi.org/10.14358/PERS.72.7.823>.
- [16] I Lawrence, Kuei Lin, A concordance correlation coefficient to evaluate reproducibility, Biometrics (1989) 255–268.
- [17] Chenguang Liu, Jianzhong Huang, Qiang Cao, Shenggang Wan, Changsheng Xie, Evaluating energy and performance for server-class hardware configurations, in: 6th IEEE International Conference on Networking, Architecture and Storage (NAS), 29–30 July 2011, Dalian, Liaoning, China, 2011, pp. 339–347. <http://dx.doi.org/10.1109/NAS.2011.23>.
- [18] Meikel Poess, Raghunath Othayoth Nambiar, Kushagra Vaid, John M. Stephens Jr., Karl Huppler, Evan Haines, Energy benchmarks: a detailed analysis, in: 1st International Conference on Energy-Efficient Computing and Networking, e-Energy'10, ACM, Passau, Germany, ISBN: 978-1-4503-0042-1, 2010, pp. 131–140. 13–15 April 2010. <http://dx.doi.org/10.1145/1791314.1791336>.
- [19] Young Lee, Albert Zomaya, Energy efficient utilization of resources in Cloud computing systems, J. Supercomput. (ISSN: 0920-8542) (2006) 1–13.
- [20] Gaurav Dhiman, Giacomo Marchetti, Tajana Rosing, vgreen: a system for energy efficient computing in virtualized environments, in: 14th ACM/IEEE International Symposium on Low Power Electronics and Design, ISLPED'09, ACM, San Francisco, CA, USA, ISBN: 978-1-60558-684-7, 2009, pp. 243–248. 19–21 August 2009. <http://dx.doi.org/10.1145/1594233.1594292>.
- [21] Anton Beloglazov, Jemal Abawajy, Rajkumar Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing, Future Gener. Comput. Syst. (ISSN: 0167-739X) 28 (5) (2012) 755–768. <http://dx.doi.org/10.1016/j.future.2011.04.017>. Special Section: Energy efficiency in large-scale distributed systems.
- [22] John C. McCullough, Yuvraj Agarwal, Jaideep Chandrashekar, Sathyanarayan Kuppuswamy, Alex C. Snoeren, Rajesh K. Gupta, Evaluating the effectiveness of model-based power characterization, in: Proceedings of the 2011 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'11, Berkeley, CA, USA, 2011, pp. 12:1–12:14. SENIX Association.
- [23] Dimitris Economou, Suzanne Rivoire, Christos Kozyrakis, Partha Ranganathan, Full-system power analysis and modeling for server environments, in: Workshop on Modeling, Benchmarking and Simulation (MoBS), International Symposium on Computer Architecture-IEEE, June 2006 <http://dx.doi.org/10.1145/102111.1715>.
- [24] Josep Subirats, Jordi Guitart, Assessing and Forecasting Energy and Ecological Efficiency on Cloud Computing Platforms. Technical Report, Universitat Politècnica de Catalunya Departament d'Arquitectura de Computadors, 09, 2010. URL: <http://www.ac.upc.edu/app/research-reports/html/RR/2013/48.pdf>.
- [25] Gregory Katsaros, Pascal Stichler, Quantifying ecological efficiency in Cloud computing, in: Jorn Altmann, Kurt Vanmechelen, Omer F. Rana (Eds.), GECON'13, in: 10th International Conference on Economics of Grids, Clouds, Systems, and Services, vol. 8193, Springer International Publishing, Zaragoza, Spain, ISBN: 978-3-319-02413-4, 2013, pp. 74–89. 18–20 September 2013.
- [26] Suzanne Rivoire, Mehul A. Shah, Parthasarathy Ranganathan, and Christos Kozyrakis. Joulesort: a balanced energy-efficiency benchmark, in: ACM SIGMOD International Conference on Management of Data, SIGMOD'07, ACM, Beijing, China, ISBN: 978-1-59593-686-8, 2007, pp. 365–376. 11–14 June 2007. <http://dx.doi.org/10.1145/1247480.1247522>.
- [27] Spyros Makridakis, Steven C. Wheelwright, Rob J. Hyndman, Forecasting Methods and Applications, John Wiley & Sons, 2008.
- [28] Paul Jensen, Forecasting Theory: Double Exponential Smoothing, September 2013. URL: http://www.me.utexas.edu/~jensen/ORMM/omie/operation/unit/forecast/double_exp.html.
- [29] SPECweb2005 Release 1.20 Benchmark Design Document, 12, 2009. URL: <http://www.spec.org/web2005/docs/designdocument.html>.
- [30] Scott Trent, Michiaki Tatsubori, Toyotaro Suzumura, Akihiko Tozawa, Tamiya Onodera, Performance comparison of php and jsp as server-side scripting languages, in: Valérie Issarny, Richard Schantz (Eds.), 9th ACM/IFIP/USENIX International Conference on Middleware, 1–5 December 2008, Leuven, Belgium, in: Lecture Notes in Computer Science, vol. 5346, Springer, Berlin, Heidelberg, ISBN: 978-3-540-89855-9, 2008, pp. 164–182. http://dx.doi.org/10.1007/9783540898566_9.
- [31] E. Tejedor, R.M. Badia, Comp superscalar: Bringing grid superscalar and gcm together, in: 2008. CCGRID'08. 8th IEEE International Symposium on Cluster Computing and the Grid, 19–22 May 2008, Lyon, France, 2008, pp. 185–193. <http://dx.doi.org/10.1109/CCGRID.2008.104>.
- [32] COMP Superscalar, 08, 2013. URL: <http://compss.bsc.es>.
- [33] David G. Kendall, Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain, Ann. Math. Statist. (1953) 338–354.



tion and middleware.

Josep Subirats received a M.Sc. degree in telecommunications from the Universitat Politècnica de Catalunya (UPC), Spain, in 2010, after completing his Master Thesis in the Mobile Marine Robotics Research Center of the University of Limerick, Ireland. He obtained a M.Sc. degree in Computer Architecture, Networks and Systems also from the UPC in 2013. He is currently working in the Autonomic Systems and e-Business Platforms group of the Barcelona Supercomputing Center as a research engineer. His research interests include green computing and the smart management of resources in datacenters using virtualiza-



Jordi Guitart received the M.S. and Ph.D. degrees in Computer Science at the Technical University of Catalonia (UPC), in 1999 and 2005, respectively. Currently, he is an associate professor at the Computer Architecture Department of the UPC and an associate researcher at Barcelona Supercomputing Center (BSC). His research interests are oriented towards energy-aware computing and the smart management of resources in virtualized datacenters. He is involved in a number of European projects.