accepted for publ. on IEEE Trans. on Cloud Computing, special issue on Green and Efficient Cloud Computing, in press

CO_2-aware Adaptation Strategies for Cloud Applications

Cinzia Cappiello, Nguyen Thi Thao Ho, Barbara Pernici, Pierluigi Plebani, Monica Vitali
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

Abstract—The increasing utilization of cloud resources raises several issues mainly related to environmental impact and, more in general, sustainability. Recently, most of the contributions have focused on energy efficiency achieved through a better physical and virtual resource management. The present paper considers instead the application level, extending the focus to the reduction of CO_2 emissions related to the execution of applications. We aim to exploit adaptivity through the design of an Application Controller that, enacting the right adaptation strategy for a given context, allows the improvement of the trade off between QoS and CO_2 emission reduction. The effectiveness of the approach has been shown running an HPC application in a federated cloud infrastructure.

Index Terms—cloud computing, Green IT, CO_2 emissions, adaptive systems

1 Introduction

In recent years, an efficient usage of IT resources in data centers and cloud infrastructures is one of the main goals for organizations. Approaches to make data centers more energy efficient have been proposed [1] [2] [3] and the debate has been extended more recently to cloud infrastructures. While on the one hand the debate in the cloud and data centers literature has focused mainly on an efficient use of physical resources (i.e., host dynamic management and server consolidation), on the other hand the issue of providing green applications and reducing their environmental impact has also been considered. For instance analysing business processes to lower their environmental impact considering the way they are executed [4] [5].

Goal of this paper is to discuss an additional opportunity for reducing the environmental impact of applications. At a glance, we propose an approach that controls the applications during their entire life-cycle, i.e., from the design to the execution. At design-time, the approach suggests the optimal set of Virtual Machines (VMs) to be deployed according to the application profile; at run-time, adaptation strategies are enacted to reduce CO_2 emissions. These strategies are based on the information known at application level to further reduce environmental impact of applications once deployed in a virtualized or cloud infrastructure. As reported in the paper, potential for improvement is currently up to a 60% reduction of CO_2 emissions. Such a significant reduction has been obtained making the application aware of the characteristics of the cloud platform, providing useful information for dynamically exploiting the VMs with the best characteristics out of the VMs assigned to it. The main idea is to exploit the different characteristics of VMs, even when requested with the same configuration. In fact, their performance depends on the host on which they are deployed and on the current status of the infrastructure (e.g., site and server loads). Moreover, among many VMs, which are supposed to be similar, one or more could be considered "golden machines", i.e., more performant, or with less CO_2 emissions. This information is only available after the deployment and should be used at run time to exploit these machines more than others. Yet, also the number of requested VMs can have a major impact on CO_2 emissions of an application or on its energy consumption: the best solution consists in selecting the right number of resources for the tasks to be handled.

For the moment, the approach presented in this paper has been validated for High Performance Computing (HPC) applications but further types of applications are being investigated. The validity of the approach has been demonstrated using a real application running in the ECO_2Clouds infrastructure[1], which provides a sophisticated monitoring environment including ecometrics to assess environmental impact and performance.

This paper is organized as follows. In Sect. 2 we discuss the state of the art for improving energy efficiency and reducing environmental impact of data centers and cloud infrastructures. Sect. 3 illustrates the overall approach, which is later validated in the case study presented in Sect. 4. Sect. 5 presents the metrics on the basis of which we define the adaptation strategies illustrated in Sect. 6. A technique for selecting the most suitable strategy is discussed in Sect. 7. Finally, validation is discussed in detail in Sect. 8, whereas Sect. 9 concludes the paper outlining possible future work.

2 Related work

Energy efficiency in data centers and clouds has been one of the mainstreams in research in recent years. According to the survey proposed by Vitali and Pernici [3], three main aspects have been considered in the literature under the umbrella of energy efficiency of Information Systems: assessment, measurement, and improvement. The work

• *E-mail: cinzia.cappiello@polimi.it*

1. http://eco2clouds.eu

proposed in this paper mainly focuses on the improvement of the sustainability in Information Systems but with some peculiar aspects. Most of the literature concerns the infrastructural and virtualization layers, aiming at reducing the energy consumed by the systems. We extend the area of investigation to the application layer; then, we propose an approach for adapting applications running on federated cloud environments with the aim of reducing CO_2 emissions instead of energy consumption. These aspects differentiate our work with respect to the related literature. In fact, there are just few contributions that provide some guidelines to quantify and manage emissions in data centers (e.g., [6], [7]).

As the literature on the energy efficiency in Information Systems is significant, we introduce only some approaches, considering three main aspects: VM deployment decisions, run time adaptation and workload rearrangement, and CO_2-aware adaptation.

The first group focuses on the VM placement. Allocation algorithms are used to decide the best location of VMs involved in an application. The decision depends on the policies adopted to allocate the resources, also taking into account the energy efficiency perspective. An example can be found in [8], where resources are allocated using a bin packing algorithm based on an energy-aware heuristic. Algorithms considering eco-metrics for the deployment of VMs have been considered by eco4clouds [9], where a two-layer algorithm has been proposed for multisite deployment. In our approach we are assuming that the VM allocation to a specific host is given and can not be influenced. However, we exploit information coming from the monitoring infrastructure to take decisions about the best configuration in terms of number of VMs needed to execute the job and amount of resources that have to be allocated to each VM to reduce energy consumption while respecting quality of service (QoS) constraints.

The contributions related to the run-time adaptation mainly focus on how to adapt the system to maintain or further improve the energy efficiency. In [10], authors suggest three main adaptation strategies to improve energy efficiency in clouds, namely VM reconfiguration, VM migration, and physical server powering off/on. The issue of an effective resource allocation algorithm is also considered in [11], where resources are allocated taking into account the total energy consumption, the number of violations of the Service Level Agreement (SLA), and the number of migrations. Resources allocation is related to both new and old VMs, which can be moved to improve the system state. The approach uses bin packing for placing new VMs and an algorithm lead by CPU usage for deciding when to migrate a VM. A framework for energy efficiency awareness and adaptation is proposed in [12]. Measuring power consumption at the server level, authors propose models for assessing and predicting energy consumption at server, VM, infrastructure, and service level. Optimization is achieved considering actions as consolidation and migration, trying to maximize the efficiency of all nodes. In [13] the focus is in particular on improving energy efficiency with consolidation practices focusing also on the characteristics of the machines: an older machine is likely to consume more energy to perform a task rather than a more recent one. The paper focuses on consolidation and dynamic server hibernation and resume. In the literature, particular emphasis has been given to the workload consolidation [14] as a way to make the cloud systems more efficient. In [15], the author focuses on the CPU usage as a way for determining where to redirect the workload of a task, based on the existing correlation between the CPU usage and the power consumption. Considering the application level, [16] proposes an approach based on the workload prediction to have a faster rearrangement of the system with the aim of reducing the energy consumption. In this paper, we are not implementing consolidation or migration, since we are considering a system in which we are not in charge of deciding the placement of VMs. We are instead implementing a logic to allocate the application tasks to the assigned VMs taking into account their performance, their power consumption, and their environmental impact in terms of CO_2 emissions. We are also proposing a decision strategy able to detect inactive VMs and to switch them off in order to reduce the power consumption of the application by avoiding idle time.

Finally, moving from energy-awareness adaptation to the CO_2-awareness adaptation, the source of energy becomes an important aspect as it affects the amount of CO_2 emissions. The main issue is to use efficiently the renewable resources available while avoiding peak demand of energy from the electricity providers. In [17], authors propose the usage of Geographical Load Balancing (GLB) to shift workloads and avoid peak power demands. This requires to predict both the incoming workload and the peak demand to the network. The algorithm is implemented as a network flow optimization problem. A similar approach is discussed in [18], which uses both workload shifting and local power generation for avoiding peak loads demands on the energy network. The importance of considering the types of sources of energy has been addressed also in [19] which proposes an integrated framework for sustainable clouds where data centers, communication networks and energy sources information are considered. Some approaches as [20] propose to compensate CO_2 emissions of a Business Process using the Green Compensation Pattern. This pattern does not modify the process or the resources but simply applies alternative actions to balance the emissions of the service. The CO_2 impact of the process is estimated starting from the CPU usage of the VMs involved. Energy consumption of the VM is estimated from its CPU utilization as described in [21], dividing the energy of the physical server between all the VMs deployed on it. Emissions are computed knowing the energy mix of the specific site. An automated method for applying the pattern is proposed. In our work, we consider the energy-mix from an application perspective, trying to manage the distribution of the workload between the VMs assigned to the considered application which have the lower environmental impact. Moreover, we propose a technique to estimate future trends in emissions and to suggest a different deployment time in order to improve the sustainability of the application.

3 Overall approach

The aim of this work is to make the application execution as efficient as possible considering the whole application

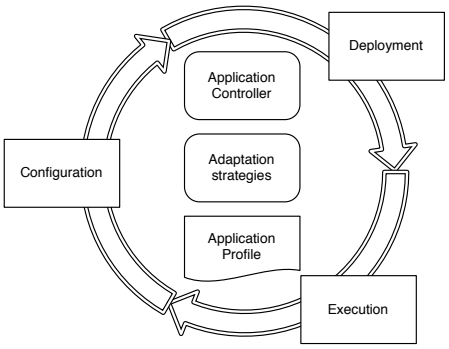


Fig. 1. Application life-cycle

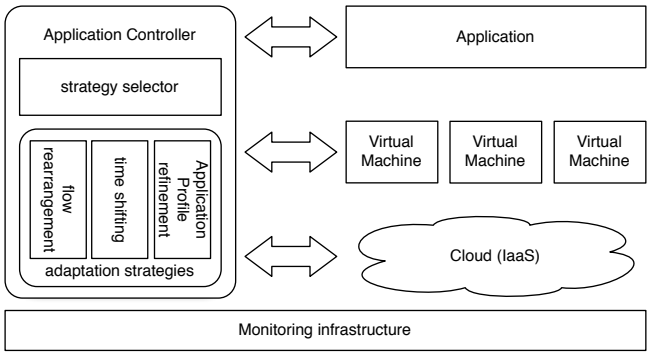


Fig. 2. Application controller

life-cycle: i.e., taking into account the requirements of the application developers and users and, at the same time, adapting the execution to the dynamic environment. As shown in Fig. 1, the first phase concerns the *Configuration*, i.e., the definition of the requested resources, as well as the associated QoS requirements which also include constraints on eco-metrics. The characteristics of applications, as resulting from the configuration phase, are represented in an *Application Profile* associated to the application, that is used as an input to the deployment and execution phases. Once the Application Profile is defined, we assume that the *Deployment* is performed on a cloud infrastructure assigning to the application the requested VMs. Finally, the *Execution* of the application starts. We propose a set of *Adaptation Strategies*, supervised by an *Application Controller* associated to each running application, to enable the satisfaction of requirements with limited resources. Generally speaking, a just-in-time approach can be applied during the execution of applications, in order to make their execution as efficient as possible according to the requested resources and the specified requirements. In addition, if an application, once terminated, needs to be executed again (as might occur in case of HPC applications), the cycle restarts and a new configuration can be requested to optimize the number and type of resources to be deployed. Details on these adaptation strategies are given in Sect. 6.

Fig. 2 shows the architecture of the Application Controller, whose objective is to support the execution of the application selecting and enacting the adaptation strategies for: i) satisfying requirements specified by the user and ii) using the resources efficiently according to the requested parameters. The Application Controller is therefore composed of the following elements:

- a set of *adaptation strategies* enabling adaptation of the application execution; we distinguish between mechanisms that are used during the execution of the application (*run-time adaptation*), such as flow rearrangement and time shifting, and mechanisms that are used to adapt the application configuration in repeated sequences of execution (*design-time adaptation*), using an application profile refinement strategy. For instance, a run-time action might release a VM, while a design-time adaptation might change the number of requested VMs in future deployments of the application, based on the results from previous similar executions;
- an *adaptation strategy selector* that automatically selects among the available possible adaptation strategies, based on: i) an initial configuration of the application controller (e.g., specifying the time intervals for the control actions), ii) the defined strategies, and iii) the context of the execution. The context is defined as the values assumed by the parameters defined for the application, including QoS (e.g., response time) and eco-metrics (e.g., the CO_2 emitted by the application during its execution or its forecast value). A monitoring infrastructure provides the values of eco-metrics and parameters considered in our CO_2-aware strategies which are described in Sect. 5.

The Application Controller selects and enacts adaptation strategies according to the rules defined in the adaptation strategy selector. The Application Controller also controls the requests of the application to the underlying infrastructure (e.g., releasing VMs, requesting additional VMs) getting monitored parameters from the cloud infrastructure (e.g., CPU usage within used VMs). The context parameters, the chosen metrics, and the strategies are application dependent and can vary depending on the considered application.Once the Application Controller is configured, several experiments can be launched using the same control mechanisms and the same action selection logic. The experiments can have different characteristics according to requests of the users; for instance, a single run (instance) of the application may be launched, or a set of instances may be launched together in the same experiment; different constraints on the upper bound of emissions and duration can be specified.

The Application Profile associated to each HPC experiment contains the result of the Configuration phase and specifies the requested VMs, their configuration, and the constraints on QoS and eco-metrics according to four main set of metadata: (i) *Resource metadata*: provide information regarding the type and characteristics of resources (i.e., VMs and storage) requested to execute the application; (ii) *Flow metadata*: provide information regarding the list and the structure of the tasks composing the application and the link between tasks and nodes specified in the resource metadata;

Resource metadata	3 VM (1 CPU, 4 GB RAM, 2GB disk) computation 1 VM (1 CPU, 1 GB RAM, 10GB disk) storage
Data metadata	OceanographicData (OD) 3 GB size, binary files
Flow metadata	see Figure 3 OneYearSimulationTask reads OD OneYearSimulationTask writes 1-year trajectories
Energy and performance requirements	response time: 3 hours CO_2 emission: 10 grCO_2eq

TABLE 1
Eels case study Application Profile

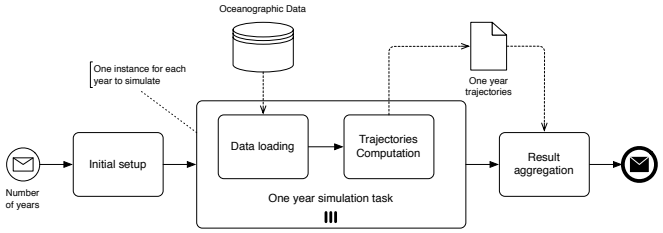


Fig. 3. Eels application workflow.

(iii) *QoS and eco-metric requirements*: refer to energy and performance conditions or constraints within process flows, including eco-metrics such as CO_2 emissions and time constraints; (iv) *Data metadata*: provide information regarding the data used by the application and the dependency among tasks and data. The Application Profile is the input for the Application Controller that uses these metadata to evaluate which adaptation strategies has to be enacted during the execution of the experiment.

In the following, we present an approach able to support adaptation to reduce CO_2 emissions of an application.

4 Running case study

To better explain and motivate our approach, in this article we refer to a HPC application in the ecology domain [22]. The application computes the trajectories followed by eels cohorts during their travel along the Atlantic Ocean to European coasts. The application, based on a parametric prediction model, provides a description of the life-cycle of eel larvae and the characteristic of the water they cross during their travel to the European coasts in terms of temperature and salinity. Although we are referring to a specific application, the presented approach can be extended to many other scenarios, as typical patterns adopted in HPC applications are considered. For instance, the application starts with a data loading phase followed by a computational phase. Moreover, the involved activities can indifferently run in sequence or in parallel. This leaves the freedom to organize them in a structure that can be adapted with the final goal, in our case, to reduce the CO_2 emissions. This flexibility can be feasible on a cloud infrastructure, as each activity runs in its own VMs that can be easily added and removed. Another set of applications that could fit to this scenario includes applications using the MapReduce programming model where several nodes implementing the reduce activity can be dynamically created and removed to run the activities in parallel. Another field of applicability are business processes where some activities have multiple instances and these instances are atomic and can be executed independently.

Tab. 1 shows an example of Application Profile describing the eels application. To run this application two types of VMs are required: (i) one for storing the Oceanographic data (expressed also as requirements in the data metadata section) about the characteristics of the Ocean in terms of temperature and salinity required to compute the trajectories; (ii) the other one for the computation of the eels trajectories (in this case 3 VMs of this type are required). For the energy and performance requirements, we assume that the user asks that the execution of the application cannot last for more than 3 hours and the total CO_2 emissions must be lower than 10 grCO_2eq for a single execution. About the flow metadata, as shown in Fig. 3, the application can be represented as a process composed by a number of separate steps, which can be executed independently in different computational nodes. The input of the application is a temporal interval expressed in years that indicates for how long the eels cohort travels along the ocean, and the objective of the application is to simulate which will be the trajectories that each eel is estimated to follow. For the sake of simplicity, we assume that the size of the cohort is a constant value (i.e., 5'000). This assumption does not affect the generality of the approach since increasing the size of the cohort results only in a linear increase of the execution time of the simulation task.

After an initial setup required to proper initialize the computation variables with respect to the number of years requested to be simulated, for each year an instance of the simulation task is created and executed. This task firstly loads the Oceanographic Data, then it computes the trajectories for each of the eels. Once all the simulation tasks terminate, the application aggregates all the values and returns the results. The simulation task is the core of the application and, due to the nature of the data and of the underlying biological model, a simulation of a given year is independent from the simulation of another year. This means that when simulation of several years are requested, the simulation tasks can run in parallel (as shown in Fig. 3) or sequentially. It is also possible to have a mix, with some simulations running in parallel followed by some other simulations running sequentially.

5 Eco-metrics

In order to offer CO_2-aware cloud environments, it is necessary to monitor the environmental impact of the execution of applications, i.e., their carbon footprint. Carbon emissions can be assessed on the basis of two variables: *power consumption* and *emission factor*. The former can be gathered from the infrastructure through external probes and/or computing hardware functionalities. The latter results from the *energy mix*, that is the variety and quantity of the energy sources used by the analysed cloud site. These two parameters are sufficient for the assessment of the environmental impact, but they are not sufficient for its improvement. In order to be proactive and improve the application greenness it

is necessary to have details of all the variables that might have an impact on the quantity of power consumed by the application. A cloud infrastructure has three main layers to monitor: (i) an infrastructural layer that considers the energy mix and the characteristics of the cloud site together with its resources; (ii) a virtualization layer that contains the VMs, which are responsible to host the applications; (iii) an application layer that includes the applications running on one or more VMs. For each layer we identify the metrics that should be monitored. Such a model can be applied both in infrastructures with a single site (i.e., a data center) or in federated cloud infrastructures. Once data are collected, their analysis is partially dependent on the application that is running, so we let the users specify thresholds for the given metrics in the application profile. Such thresholds drive the enactment of adaptation techniques.

5.1 Infrastructure layer metrics

At the infrastructure layer, it is important to consider the characteristics of the cloud site and of its hosts. At the cloud site level, the approach presented in this paper mainly focuses on the definition of the *energy mix* that measures the variety and amount (in percentage) of the different energy sources that feed the site. The energy mix provides the information to assess the *Carbon Emission Factor*, i.e. the quantity of grams of carbon emissions per each KWh of energy produced. The energy mix can be retrieved in different ways. Some countries publish the real time energy mix via public web sites (e.g., France publishes the energy mix used at the national grid level on on the RTE website [2]). In this case the emission is a weighted average of the sources emission factors on the basis of the percentage of power generated. The source emission factors can be found in different literature contributions. In particular, we consider the carbon footprint of different production technologies given by [23], since they include also the carbon footprint left during the construction phase, maintenance, operation, and decommissioning of the energy source. Formally, if TE is the total energy generated in a country expressed in KWh, SE_k is the energy generated by the k-th source, and ef_k is the emission factor related to the k-th source expressed in g/KWh, the total emission factor ef of the cloud site can be calculated as follows:

$$ef = \sum_{k=1}^{K} \frac{SE_k}{TE} \cdot ef_k \tag{1}$$

For countries in which the real time energy mix is not available it is possible to rely on electricity operators that periodically publish the average energy mix within the associated carbon emission factor of the country in a specific period. In this case, assuming to know the average power consumption (AP) for a specific site, the energy (kWh) consumed in a specific period can be estimated by multiplying AP by the number of hours in the considered period. CO_2 emissions result by multiplying the energy consumed by the emission factor ef (that is a constant calculated considering the average energy mix in a certain period). These two assessment approaches can be also used when the cloud site does not depend on the national grid. In fact, the site might rely on its own energy sources and have a complete knowledge on the trend of its energy mix or it might be fixed since the site has stipulated a specific contract with the energy providers.

Beside the energy mix, traditional indexes for measuring the site energy efficiency, such as PUE, are considered. In addition, we monitor also metrics able to provide information on the energy efficiency class of the machines: the *CPU Utilization*, the *Disk IOPS* (i.e., I/O operations of the disk), and the *power consumption* of the machine (that can be gathered through external probes).

5.2 Virtualization layer metrics

Energy consumed by an application depends on the power consumption of its VMs. To calculate this value, we adopted the approach defined in the ECO_2Clouds Project [24] [25] where the power of a VM depends on both the characteristics of the VM and the other VMs deployed on the same physical host. More precisely:

$$P_{vm} = P_{vm}^{idle} + P_{vm}^{working} \tag{2}$$

where:

- $P_{vm}^{idle} = \frac{P_{host}^{idle}}{\#VM}$. This represents the fraction of the idle power of the host assigned to the VM. This depends on the number of VMs deployed on the same host but it does not depend on the characteristics of the VMs. In fact, regardless of the size of a VM, each of them are equally responsible for the fixed cost of P_{host}^{idle}.
- $P_{vm}^{working} = (P_{host} - P_{host}^{idle}) \cdot CPUusage_{vm}$. This represents the fraction of the working power of the host assigned to the VM. This varies during the time as it depends on the CPU usage, as seen by the host, of the VM.

5.3 Application layer metrics

Application layer metrics characterize the application. The values of such metrics depend on the deployment of the application, i.e., on the way in which the different tasks are assigned to different VMs. In fact, the evaluation of metrics such as *Application response time* and *Application power consumption* can be calculated only by aggregating the corresponding values gathered from the different VMs. In this category, it is worth to highlight a metric, the *Application PUE (A_PUE)* (see [26]), able to detect energy inefficiencies that can occur during the execution of the application. It is defined as the ratio between the actual total amount of power required by all VMs of an application and the estimated power needed to execute the application tasks. Thus, in order to perform the assessment of A_PUE, it is necessary to provide an estimation of the power consumed by the different application tasks during their execution. Such estimation can be based on the theoretical analysis of the characteristics of the application or on empirical knowledge gathered from previous experiments. A_PUE is able to highlight energy waste. Once that the total amount of energy required for executing the application is known,

2. http://www.rte-france.com/fr/

	resource metadata	*data metadata*	*flow metadata*	*energy and perf. req.*
flow rearrangement	X		X	X
time shifting	X			X
app. profile refinement	X	X	X	X

TABLE 2
Application Profile sections used by the adaptation strategies

it is possible to calculate other two important indexes: *Application Energy Productivity* and *Application Green Efficiency*. The former provides the energy consumed by the different application tasks while the latter measures the amount of green energy used to run an application on the basis of the percentage of green sources used to provide electricity inside the analysed site.

6 Strategy Selection

This section shows how adapting applications following specific strategies can have a significant impact on CO_2 emissions reduction. The solution proposed in our approach to support the adaptation is based on the usage of an *Application Controller* that defines and manages adaptation strategies for a running application. In particular, we investigated three adaptation strategies: (i) flow rearrangement, (ii) time shifting, and (iii) application profile refinement.

The first one (Sect. 6.1) operates at run-time along the whole duration of the application execution. It considers applications already deployed, with the objective of finding a way to make them flexible enough to change their behavior in order to reduce CO_2 emissions while maintaining satisfied the other non-functional requirements (e.g., response time, throughput). It is executed automatically, mainly driven by the current status of the system. The second one (Sect. 6.2) operates at run-time but only when the application is about to start, in order to check the possibility to postpone the execution. Based on the current energy mix trend, the system identifies the starting time in which this mix reflects a greener status. Finally, the last adaptation strategy (Sect. 6.3) operates at design time and gives to the developers a support to redefine the application profile, evaluating alternative configurations, in terms of number and size of requested VMs, that can provide better performance with less CO_2 emissions. The design-time adaptation strategy is driven by logs and monitoring data obtained from previous executions. This is particularly suitable in case of HPC applications where the same application can be repeatedly executed in different moments. By analyzing the behavior of previous executions, developer is in the following supported by our solution to find greener alternatives to execute the same work.

All these strategies use as input the information contained in the Application Profile as described in Tab. 2.

6.1 Flow rearrangement

Flow rearrangement affects the structure of the application flow that can be modified by the Application Controller in different ways:

- Rearranging the workload assigned to the tasks composing the application and switching off a VM if no longer needed.
- Skipping tasks if they are defined as 'optional' in the Application Profile under critical timing conditions.

This strategy can be applied to applications that can be decomposed into atomic tasks, where several instances of the same task can run in parallel to improve the performance of the entire application. Referring to our case study, the eels application includes the simulation task that computes the trajectories of a cohort of eels towards the ocean along a set of years. This simulation can be decomposed on several small simulation tasks, each of them considering the movement of the eels along a single year. Due to the nature of the application and the data used, these small tasks can be executed in sequence or in parallel without affecting the final results. Moreover, some tasks, as for instance the result aggregation, can be declared optional if the system realizes that the constraint on the application response time is going to be not respected.

Concerning this strategy, the Application Controller implements mechanisms to support the workload rearrangement and the task skipping. Fig. 4 shows a BPMN representation of the Application Controller behavior when managing one of the tasks of the application flow, where such a task can be executed by several nodes in different Application VMs.

At the beginning, the Application Controller performs the initial workload distribution. If no information about the status and the performance of the application nodes is available, the Application Controller evenly distributes the workload among the nodes. In case the application has been previously executed (common situation in HPC scenarios) the analysis of its previous behavior can be used for defining the initial distribution. Every time an assigned task completes, the Application Controller aims to converge to the workload distribution that minimize CO_2 emissions, by revising the workload distribution on the basis of the information on power, response time and energy mix (using the technique discussed in Sect. 5) that are collected in the meanwhile. Listing 1 details the algorithm of the workload rearrangement activity currently adopted. After computing the CO_2 emissions for each of the running VMs (see line 5), which depend on both response time and the VM power, the tasks are inversely proportional distributed with respect to their emissions: the higher the emissions, the lesser the number of tasks to be included in the worklist (see line 10).

The Application Controller manages the nodes, where the application actually runs, communicating with the *Application Local Agent*. The agent gets informed about the worklist to be executed and informs about the tasks completion. Moreover, the agent controls the execution of the task and, if requested by the Application Controller, it switches off the managed VM. A VM is switched off when the rearrangement algorithm realizes that no task will be assigned to it.

Time complexity of the proposed algorithm linearly depends on the number of the VMs that the application controller has to manage and the number of tasks that have

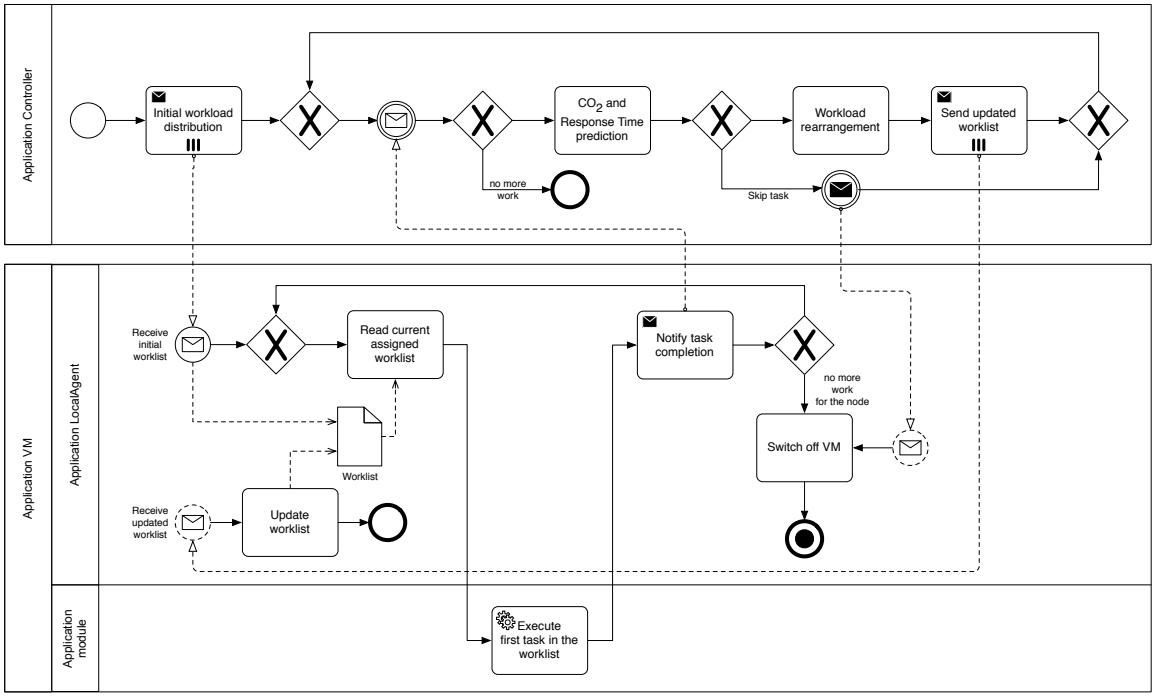


Fig. 4. Flow rearrangement overview.

Listing 1 Workload rearrangement algorithm

Input: $tasks[no\_tasks]$: tasks to be executed
Input: $vmset[no\_vm]$: list of vm nodes
Output: $worklist[no\_vm]$: tasks assigned to nodes

for vm: vmset **do**
  $p \leftarrow$ ESTIMATE_POWER(vm)
  $rt \leftarrow$ ESTIMATE_RESPONSETIME(vm)
  $em \leftarrow$ ESTIMATE_ENERGYMIX(vm)
  $VMCO_2[vm] \leftarrow p * rt * em$
end for
$tot\_CO_2 \leftarrow$ SUM($VMCO_2$)
$j \leftarrow no\_tasks$
for vm: vmset **do**
  $no\_assigned\_tasks \leftarrow no\_tasks * (1 - \frac{VMCO_2[vm]}{tot\_CO_2})$
  if $no\_assigned\_tasks > 0$ **then**
    for i: $no\_assigned\_tasks$ **do**
      $j--$
      $add(worklist[vm], tasks[no\_tasks - j])$
    end for
  else
    SWITCHOFF(vm)
    no_vm--
  end if
end for

to be distributed on them. Although in this work we did not elaborate a deeper analysis on how the increasing of VMs can affect the performances of the algorithm, the instruction composing the algorithm are so basic that its response time is usually negligible with respect to the task duration. On the other hand, network can be the element that can have a significant impact on the algorithm. As the decisions taken by the Application Controller are also based on the data provided by the Application Local Agents, network latency could reduce the performances of the workload rearrangement.

6.2 Time shifting

As described in Sect. 5, the type of energy sources and the percentage of green energy used to produce electricity define the energy mix that influences the quantity of CO_2Ċlearly, since different countries (or regional areas) use in a different way the energy sources, each country supplies a different energy mix and cloud sites might execute the same application by producing different quantities of carbon emissions. For this reason, in a cloud environment, the load distribution between VMs belonging to the same application should be also driven by the analysis of the energy mix of the different sites available.

Analyzing energy mixes we find out that they are characterized by seasonal patterns that describe the way in which the site emission factor varies during the day. For instance, we analyzed the data related to September 2014 about the energy generation in France and UK, available through the RTE [3] and Balancing Mechanism Reporting System (BMRS)[4] websites and we find out the trends depicted in Fig. 5. We registered these pattern by storing the absolute variations in the different time period (i.e., half an hour). In particular, Fig. 5 highlights the periodical variations of the emission factors during week days and weekend days by considering, as starting points, the average values of the period. The France pattern shows that during weekdays the emission factor is slighty lower during the night. The pattern of the weekend is similar but in average the emission factor is

3. http://www.rte-france.com/fr/
4. http://www.bmreports.com/

lower than during the weekdays. The difference between weekend and weekdays is similar in UK but in this case, it is also possible to notice more significant variations of the emission factor during the days. In fact the emission factor in UK decreases in the night but between h4.00 and h6.00 it starts increasing and after some fluctuations it decreases around h21.00. As discussed in Sect. 6.2, these regular trends are exploited to design the time shifting strategy. Such strategy can be applied if the users that ask for the deployment of their application do not specify time constraints that imply the immediate execution of the application. In fact, the time shifting exploits the regular variations of the emission factors to delay the execution of the application and reschedule it in time intervals in which the emission factors is expected to decrease and CO_2 emissions are lower. Details about the time shifting strategy are provided in Listing 2. For example, If users submit their request during the afternoon, the Application Controller, considering the current site emission factor, variations defined in the pattern, the estimated energy consumption and response time of the application can calculate the quantity of CO_2 emissions at different time and decide to delay its execution at the most suitable time (e.g., in the night) for minimizing CO_2 emissions.

Listing 2 Time shifting algorithm

Input: $siteset[no_sites]$: emission factor profiles
Input: $efpattern[time, variations]$: emission factor pattern
Input: app: application to be executed
Input: $site$: site in which the application has to be executed
Output: $starttime$: proposed start time

```
rt ← ESTIMATE_RESPONSETIME(app)
en ← ESTIMATE_ENERGY(app)
em ← ESTIMATE_ENERGYMIX(site)
ef_current ← ESTIMATE_EF(site)
starttime ← ESTIMATE_CURRENTTIME(site)
emissions ← en * ef
for time:efpattern do
    ef_average ← ESTIMATE_EFAVG(site)
    emissions_est = en * ef_average
    if emissions_est < emissions then
        emissions ← emissions_est
        starttime ← time
    end if
end for
```

6.3 Application profile refinement

The application profile refinement is an adaptation strategy applied at design time. The amount of CO_2 emission is closely related to the energy consumed by the applications as reducing energy consumption causes the reduction in CO_2 emissions, provided that the energy mix is the same. In this section, we shift our focus to the achieving of energy efficiency by modifying the application profile, in particular the resource metadata.

In clouds, the more physical resources used, the more power needed to operate them. Minimizing the number of used resources can reduce the consumed power but might increase the execution time, potentially leading to QoS constraints violation. The goal of this adaptation strategy is to provide a means to analyse the impact of minimizing the power on the execution time.

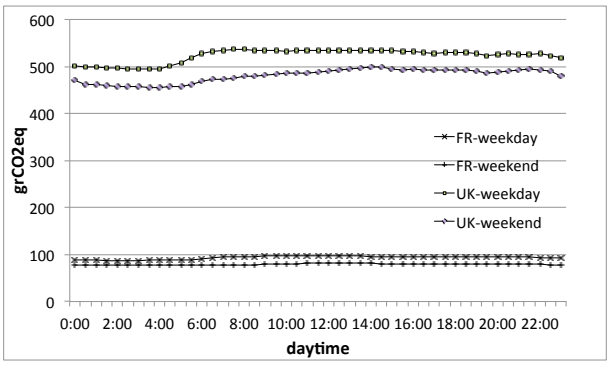


Fig. 5. Energy mix factor patterns.

We propose an analytical approach to compare several configurations that concerns how the resources are used and how the tasks are executed, particularly related to: (i) the number of used resources: it refers to the number of used VMs and their size (e.g., number of CPU, allocated memory and storage, etc.) to execute the tasks; (ii) the execution policy: it refers to the way the tasks are performed (e.g., sequential vs. parallel execution); (iii) the storage access policy: synchronous vs asynchronous access.

We use queueing theory as a mean to model different configurations to execute a given set of tasks. In the queueing model, computing resources (e.g., the VMs) are represented as a network of *stations* and the number of executed tasks is presented as *customers*. The basic station can be either queue station (i.e., station has a queue to store waiting jobs) or delay station (i.e., station has no queue) and is characterized by the service demand (i.e., the time required to serve one job at the station). Other stations perform advanced operations such as *Fork* and *Join* to simulate synchronous/asynchronous access. Each configuration has a queueing model whose inputs include:

- The number of tasks to be executed, no_tasks.
- The number of stations and their type (queue or delay station). The queue station is used in case of having shared elements (e.g., the VM is shared among tasks) whereas the delay station is used for un-shared elements (e.g., the VM is dedicated to each task).
- The set of service demand D of the stations: it varies depending on the size of the VM. For instance, for a given job, the VM with 2 CPU-cores will be executed faster, so its service demand is smaller compared to 1 CPU-core VM.

Some of these inputs can be extracted from the Application Profile. For example from the selected case study, the Flow metadata suggests two stations are present in the model, the *Storage* station to access stored data and the *Application* station to run the computation phase. The Data metadata indicates that the *Storage* is a queue station because the data storage is a shared element, while the *Application* can be a queue or delay station depending on whether the configuration shares the VMs among tasks or not. Other inputs will be given in the configurations, such as the number of

tasks no_tasks and the size of the VMs, thus determining the service demand.

The outputs of these queueing models consist of the execution time T and the resource usage (i.e., CPU usage) of each configuration. The CPU usage is used to compute the power consumption (Eq. 2). Note that the power model adopted in our approach is approximated linearly to the CPU usage. This model has been validated against our cloud infrastructure to confirm its correctness in estimating the power consumption [25]. Moreover, the P_{vm}^{idle} depends strictly on the number of running VMs ($\#VM$) on the physical server. The greater $\#VM$, the less P_{vm}^{idle} each VM has to pay for. The value of $\#VM$, however, cannot be determined at design time. Hence, in this strategy, we use the maximum possible value of $\#VM$ to have the minimum possible value of P_{vm}^{idle} associated to each VM. Other runtime adaptation strategies will exploit the real value of P_{vm}^{idle}.

The execution time T and the power P_{vm} will be used to compute the total energy consumption: $E = T \cdot \sum_{vm} P_{vm}$. The pair $< T, E >$ will drive the decision for the best configuration, considering all the QoS and energy constraints. The chosen configuration indicates the number of used VMs and their size to be updated to the resource metadata in the Application Profile by the Application Controller. For the subsequent runs of the application, the strategy selector module of the Application Controller compares the real monitored data to the previous data produced by the models in order to decide whether the configuration should be changed. This improvement step is necessary as it exploits the runtime value of P_{vm}^{idle}. Other techniques such as ERWS (Energy Response time Weighted Sum) [25] to combine the execution time and the consumed energy can also be used to evaluate the configurations. Listing 3 shows the algorithm to analyze possible configurations for a given application.

Listing 3 Configuration analysis algorithm

Input: no_tasks: number of tasks to be executed
Input: $stations$: number of stations
Input: $station\_types$: types of stations
Input: D: the set of required service demand at each station
Output: $< T, E >$: the set of execution time and estimated energy consumption of the configurations

```
for c: configurations do
    build queueing model m for c
    compute CPU usage CPU_vm from m
    compute execution time T_c from m
    compute consumed power P_c of c, using Eq. 2
    compute consumed energy E_c = P_c · T_c of c
    if T_c and E_c satisfy constraints then
        insert < T_c, E_c > to < T, E >
    end if
end for
return < T, E >
```

7 Adaptation Strategy Selection

In this section we propose a set of techniques that are employed in the strategy selector module of the Application Controller (see Fig. 2) to decide when to apply an adaptation strategy and which strategy is the most suitable given a context. We propose a technique based on the analysis of the correlations among eco-metrics starting from the data collected by the monitoring infrastructure.

In a complex environment such as a cloud infrastructure, it is important to explore possible relations existing among the collected metrics. Knowing these relations has several positive outcomes. Firstly, collected metrics can be considered as sentinels of the behaviour of the application. When the values of the metrics are outside the constraints expressed in the Application Profile, the performance or the sustainability of the application has to be improved. Knowing relations among metrics enables an indirect fixing approach since the value of a metric can be improved modifying the value of a related metric. Secondly, we can perform what-if analysis. In fact, we can predict which metrics will be affected by the modification of the value of one or more other metrics and if this impact will be positive or negative for our goals.

A Bayesian Network (BN) structure has been adopted to represent relations among indicators, where nodes are the metrics and links express relations between couple of metrics. The probability table associated to each link describes how the two metrics influence each other. The BN can be manually created by an expert or can be automatically built starting from the data collected by the monitoring system. To learn the structure of the BN we have computed the correlations between all the monitored metrics and we have selected relevant links and oriented them by using a modification of the Max-Min Hill Climbing (MMHC) Algorithm described in [27]. The algorithm uses the correlations as an initial parent set and applies heuristics to find the parent-children relations which better fit the training data set. The probability tables are computed using Maximum a Priori Estimation. The algorithm is described in [28]. In a dynamic environment, where the configuration and placement of machines could be changed over time and their behaviour can be influenced by external factors (e.g., the deployment of other machines in the server belonging to a different application can decrease performance of a VM), the automatic learning and refinement is more suitable.

Using our case study, we have extracted relations between metrics from the data collected by the monitoring infrastructure during the execution of several experiments. A graphical representations is shown in Fig. 6. Metrics collected are only relative to physical and virtual machines involved in the application. From the evidence emerged in the correlation matrix, we have observed some interesting behaviours. Firstly, the CPU usage of a server (CPU_{host}) is strongly related to the CPU usage of each VM deployed on it (CPU_{vm}), and the power consumed by a server (S_P) is strongly related to its CPU usage. Similarly, a strong relation exists between power (P_{vm}) and CPU usage of a VM. This analysis confirms the strong relation existing among the processor load and the power consumption in CPU-intensive applications. Yet, an interesting relation exists among the power consumption of a VM and the throughput of the task deployed on it (TH_{vm}). This is an indication that reducing power can result in a degradation of the QoS. For example, a negative influence has been discovered between the A_PUE metric (A_PUE) and the VMs throughput: A_PUE gets worse when the throughput of one of the VMs hosting

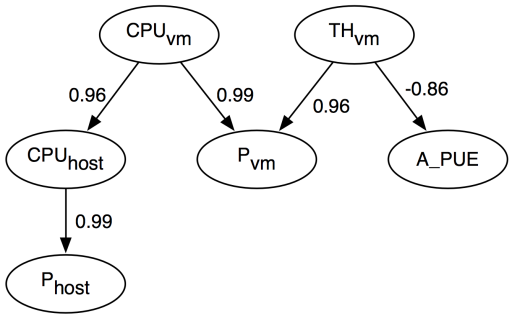


Fig. 6. Correlations among metrics for the Eels application.

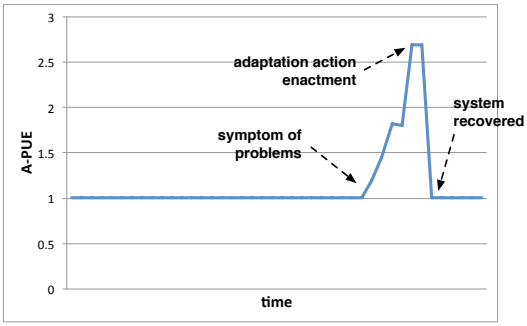


Fig. 7. Turning off a VM when an increasement in A-PUE is observed.

the tasks of the application decreases. As the throughput is a metric that is usually hard to collect, unless the code of the application is modified, its behaviour can be inferred from the A_PUE (Sect. 5.3). The ideal value for A_PUE is 1, meaning that all the power consumed by the VMs goes to run the application. On the contrary, the higher the value of A_PUE, the worse the behaviour of the application due to VMs that are not running any useful computation.

In order to allow the Application Controller to enact the best strategy given a context, it is necessary to make it aware of the effect of the available repair strategies over the monitored metrics. As previously mentioned, this information can be encoded by an expert, or can be automatically learned/refined using machine learning techniques. To learn strategy-to-metrics relations we have implemented an algorithm called Adaptive Strategy Selection (ASS) [28], that observes the outcome of an adaptation strategy over the observed metrics and keeps track of the ability of the strategy of influencing their values. This information is updated after each application of the adaptation strategy, integrating information about the most recent execution and past executions in order to reduce the side-effects of the noise in the monitoring system. This makes the algorithm suitable for a dynamic environment. Knowing which metrics are influenced by the different strategies and which are the metrics that need to be improved, the AAS algorithm can select which is the most likely successful strategy for the given context. As an example, if the A_PUE is violated, the algorithm invokes the flow rearrangement strategy which can use the information provided by the strategy selector to decide to switch off the VM that has terminated the computation by moving its workload to other VMs. Fig. 7 illustrates the trend of the A_PUE during the execution of an application, showing how the decision of turning off a VM brings the values of A_PUE inside the desired interval again. Similarly, an inefficient use of the resources allocated to the VMs can activate the Application Profile refinement strategy to find a better configuration for a future execution of the application, while the flow rearrangement strategy can also be activated by the detection of a modification of the behaviour of a VM in terms of power consumption or response time, which can be an input for rearranging the workload between the different active VMs. Finally, a high value for CO_2 emissions in a site where the application or one of its tasks is running can enact the time shifting strategy to detect a better execution time. Given this knowledge, the awareness about the relations between variables allows the selector to reason about indirect improvements: knowing how metrics interfere with each other enables to enact strategies which can directly improve the violated metrics, or indirectly improve them by intervening over related metrics.

Listing 4 Adaptive Strategy Selection

Input: BN: the BN of relations among indicators
Input: A_set: the set of available adaptation strategies
Input: $Q[no\_strategies, no\_indicators]$: a matrix describing the effect of each strategy over each indicator
Input: $C[no\_indicators]$: the current state of the indicators
Output: a^*: the strategy to be enacted

if I_n is violated **then**
  $w_n = 1$ ▷ *compute weights vector* w_n
else if I_n is near to be violated **then**
  $w_n = 0.5$
else
  $w_n = 0$
end if
for a: A_set **do**
  $p(a|C) \leftarrow \sum f(Q[a, I_n] * w_n)$ ▷ *the impact of strategy a over indicator* I_n
end for
for p: P **do** ▷ *select the parent set* P *of the violated indicators from* BN
  find the state for p which maximizes the probability of improving violated indicators and add it to C'
end for
compute values for w_n according to C'
for a: A_set **do**
  $p(a|C') \leftarrow \sum f(Q[a, I_n] * w_n)$
end for
select a^* with the highest likelihood of success given $p(a|C)$ and $p(a|C')$

Exploiting both the strategy-to-metrics relations and the mutual influences among metrics, the adaptation strategy selection logic allows the Application Controller to enact the best strategy in a given context as briefly described in Listing 4. The algorithm is a simplification of the detailed algorithm described in [28]. After the enactment of the selected strategy, its effect over the indicators is observed and matrix Q is updated accordingly.

8 VALIDATION

The effectiveness of the adaptation strategies discussed in Sect. 6 has been validated on a real federated cloud infrastructure developed in the BonFIRE and ECO_2Clouds EU projects. This platform allowed the deployment of VMs in three main sites: EPCC in UK, HLRS in Germany, and INRIA in France. Involving three different countries gave us the possibility to consider different variation of energy mix. In particular, HLRS had a static energy mix value, whereas in INRIA and EPCC the energy mix varied every 30 minutes according to the data returned by their national grid administrators. In addition, each site provided a monitoring infrastructure able to collect all the metrics at the different layers: i.e., infrastructural, virtual, and application. Concerning the computation of the eco-metrics, all the sites were equipped with PDUs (Power Distributed Units) able to periodically compute the amount of power consumed by all the hosts installed on the site. Based on this value, the power consumed by each VM, the CO_2 emission and all the eco-metrics were computed according to the metrics discussed in Sect. 5.

The application used to validate the approach refers to the case study described in Sect. 4. For the first two adaptation strategies we considered (i) a *baseline* configuration, and (ii) several *optimized* configurations defined to validate the introduced adaptation strategies. More in details, the *baseline* configuration concerns the execution of the eels application using 4 VMs: (i) 3 VMs (2 CPUs, 2 GByte memory) for the actual execution of the simulation tasks; (ii) 1 VM (1 CPU, 1 GByte) for collecting and computing the metrics.

Having 15 years to be simulated, the baseline configuration concerns the execution of these tasks evenly distributed among the three VMs, i.e., 5 years each. In addition, the baseline configuration requires the execution of the application on Sep, 1st 2014 (Monday) at midnight where postponement is not allowed.

As described in details in the related paragraphs, the *optimized* configurations used to validate the flow rearrangement and the time shifting strategies operates, respectively, on how the workload is distributed and on when the application can start. About the application profile refinement the validation refers to the comparison among available application configurations.

To control and enact the run-time adaptation strategies we developed a Java-framework freely available on GitHub[5]. This framework provides two main classes, i.e., the *ApplicationController* and the *ApplicationLocalAgent* (the application-independent part) and two interfaces that the developers willing to make their application adaptive have to implement (the application-dependent part). One interface, the *ApplicationMetric*, needs to be implemented to expose the application metrics that, in addition to the infrastructure and virtual layer metrics, are relevant for the application. Implementation of this interface also requires the connection to the monitoring system as this depends on the infrastructure where the VMs will be deployed. In our case, this has been implemented with a connector to the ECO_2Clouds Monitoring Infrastructure. The second interface is the *StrategySelector* that implements the adaptation strategy selection policies discussed in Sect. 7. Once the Application Controller has been customized, an instance of it runs for each of the application instances, as the role of the Application Controller, especially at run-time, is to manage a single application instance. As a consequence, this configuration easily scales as the Application Controller does not have to control an increasing number of application instances, but it might introduce overhead, as also discussed in the next paragraphs.

5. The framework and its implementation for the case study discussed in this paper can be found at https://github.com/ECO2Clouds/r2/tree/master/application-controller

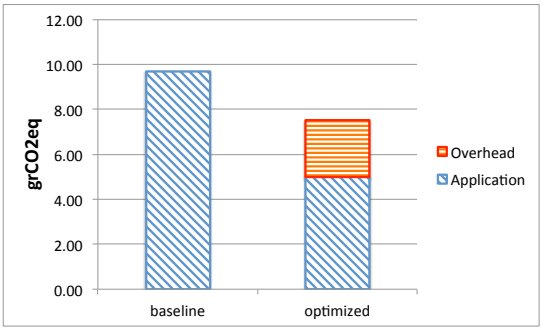


Fig. 8. Workload rearrangement validation results.

8.1 Flow rearrangement validation

The goal of this first validation is to verify that executing an application without (i.e., baseline configuration) and with (i.e., optimized configuration) the possibility to enable the flow rearrangement can reduce CO_2 emissions. To make the executions with the two configurations comparable, we considered the case where the application is deployed only on the site INRIA in France. This case is the worst one since due to energy mix mainly based on nuclear, the space for reduction here is limited. Moreover, focusing only on one country, we can be sure that the possible resulting CO_2 emissions reduction is not too biased by the different energy mix associated to the different countries. Finally, the INRIA site has a peculiar configuration as the installed hosts are very similar to each other, thus identifying and consequently exploiting the differences in energy and CO_2 emissions to reduce them become more difficult. To conclude, none of the tasks belonging to the application are declared as optional. In this way, we do not skip any task to reduce the response time and, thus, CO_2 emissions.

With respect to the baseline configuration, the *optimized* configuration enabling the enactment of this adaptation strategy, requires an additional VM hosting the Application Controller with 0.5 CPU and 256 MByte memory. The 15 years to be simulated are dynamically distributed among the 3 VMs dedicated for executing the simulation according to the approach described in Sect. 6.1. Adding a new VM to host the Application Controller introduces additional CO_2 emissions considered as overhead.

As the VMs run in a cloud shared with other applications, the baseline and the optimized configuration have been executed more than 50 times and the average of CO_2 emissions are considered to provide more reliable results. Fig. 8 reports the results of the validation trail. With the

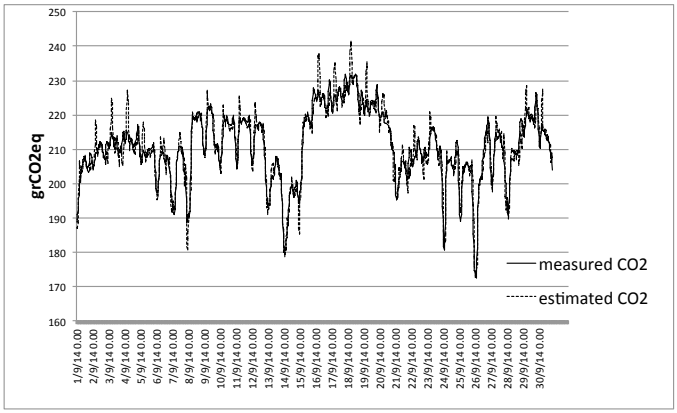


Fig. 9. Real CO_2 emissions vs. estimated CO_2 emissions computed using trends.

baseline configuration, the average CO_2 emissions for the eels application is 9.7 grCO_2eq, whereas with the optimized configuration, the total CO_2 emission is 7.52 grCO_2eq. Thus, enacting when necessary the workload rearrangement adaptation, we can obtain a CO_2 emission reduction of 22.4%, also considering the overhead of 2.52 grCO_2eq (33% of the total) introduced by the Application Controller that manages the enactment of the adaptation strategy.

8.2 Time shifting

Concerning the second adaptation action, the goal of this validation is twofold. On the one hand, we want to verify that using the energy mix patterns gives a good estimation of the CO_2 emissions. On the other hand, we want to quantify the reduction of CO_2 emissions obtained by shifting the execution of the application.

Concerning the first point, we focused on the energy mix factor in UK and we assumed to run the application of our case study only on that site. Energy mix factor has been taken from the Web site, and for the sake of simplicity, here we focus only on the information collected in September 2014. Based on about 100 executions, the monitoring infrastructure reports that the average energy consumption of 3 VMs executing the eels simulation for 3 hours is $136.56\ Wh$. Having the energy consumed, CO_2 emissions are obtained by multiplying this value to the energy mix factor observed during the execution. Considering the *baseline* configuration (i.e., application starts at midnight on Sept. 1st 2014) resulting CO_2 emissions are 188.50 grCO_2eq.

As we have two series of energy mix factors, i.e., the real one and the estimated one, Fig. 9 reports the comparison of the real and estimated CO_2 emissions for our experiments in case the application has been executed in different periods between Sep 1st and Sep 30th 2014. As reported in the figure, we can conclude that the proposed estimation of CO_2 emissions based on the usage of the trends of energy mix factor can be considered reliable, as the average error of our estimation is 1.01% with a worst case of 5.67%.

With the same result, we can also quantify the reduction of CO_2 emissions in case of postponed execution with respect to the baseline. For example, the *optimized* configuration indicates the possibility to postpone the execution till the end of the month. On this basis the Application Controller asks for the current energy mix factor, then uses the trends to estimate the factor for the whole month. If the application has been already executed some times in the past, the Application Controller can use what monitored to know which is the energy consumed by the application to estimate the CO_2 emission. Otherwise, without losing effectiveness, it uses only the energy mix values. As a result, the Application Controller has a series of values as the dashed line in Fig. 9. Here, in the correspondence of the minimum value, there is the time in which it is preferable to execute the application to have lower CO_2 emissions. If the user does not impose any constraint on when the application should start, the Application Controller realizes that the best moment is on Sep. 26th at 4:00h where the application will emit 172.75 grCO_2eq instead of 188.50 grCO_2eq, (7.97% less). As an alternative *optimized* configuration, we ask the experiment to start in no more that one week (expressing this constraint in the Application Controller), then the decision of the Application Controller is to run on Sep. 7th at 23:30h when it is possible to save up to 3.55%. As in this latter case, as the reduction is not so significant, the Application Controller can be properly configure to not postpone the execution if the CO_2 reduction is lower that a threshold decided by the user.

Yet, looking at the figure is also possible to figure out which is the best scenario, i.e., the greater difference from when the execution is requested and when it really happens. In this case, assuming that the user asks for executing the application on Sep. 18th at 08:00h, the Application Controller suggests to shift the execution till Sep. 26th at 4:00h. In this case, the CO_2 reduction is equal to 28.42% (172.75 grCO_2eq instead of 241.38 grCO_2eq).

8.3 Application profile refinement validation

The goal of this section is to illustrate how the profile refinement strategy help in the selection of the optimal configuration for the considered case study. We assume the user has to analyse N years of eels which can be divided into N tasks. Two basic configuration groups are compared: (i) (*Group 1*) One VM for All: only one VM is used to execute N tasks; (ii) (*Group 2*) One VM for Each: one VM is dedicated to a specific task.

All the tasks in the two groups have synchronous access to the storage. For simplicity, we also assume that all VMs are homogeneous. Fig. 10 and Fig. 11 show the corresponding queueing models. There are two stations in each model. The *Storage* station is modeled as a queue station since the data source is shared among tasks. The *Application* station (to execute the tasks) is modeled as a queue station (see Fig. 10) in *Group 1* because the VM is shared among tasks, and as a set of delay stations (see Fig. 11) in *Group 2* because the VMs are not shared. The *Fork* and *Join* stations are used to perform synchronized access to the storage. Other possible configurations have been investigated and the details can be found in [25].

The model inputs include: (i) the number of executed tasks N; (ii) the number of stations; (iii) the station types; (iv) the information about service demand D whose each element is a pair $(D_{storage}, D_{app})$ where $D_{storage}$ is the required time for loading data and D_{app} is the required time

Fig. 10. Group 1: One VM for All.

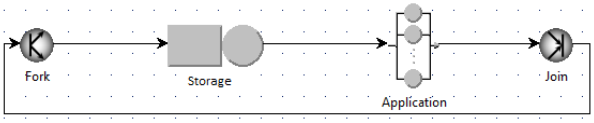


Fig. 11. Group 2: One VM for Each.

for executing a task. We use the simulation tool JMT [29] to build the queueing models and analyse them to obtain analytical results.

According to the results obtained from JMT, the *Group 1* configuration is dominated by *Group 2* in terms of execution time (see Fig. 13). This is explained easily as the *Group 1* uses one VM to execute all tasks. In contrast, *Group 1* seems more beneficial in terms of total energy consumption (see Fig. 12) since *Group 1* uses less resources in performing tasks. However, the increasing trend of the execution time and the consumed energy is far different. As the number of tasks increase, the increasing of consumed energy in *Group 2* is much larger than the increasing of execution time in *Group 1*. For instance, in Fig. 12, the consumed energy of *Group 2* is seven times more than *Group 1*, while in Fig. 13, the execution time of *Group 1* is less than double of *Group 2*. This suggests that a careful analysis of the two metrics are needed before choosing a configuration. Depending on which metric is more important, the user can prefer a configuration accordingly.

As a conclusion for this case study, assuming that an unlimited number of resources is available, *Group 2* provides better results in term of the execution time while *Group 1* consumes less energy. Therefore, the user should base on the constraints of QoS and the energy in order to decide an optimal configuration.

9 Conclusion

This paper presents an approach to reduce the environmental impact of HPC cloud-based applications considering their whole life-cycle. An adaptation mechanism driven by an Application Controller provides a support to developers and users to better configure and execute applications with the aim of reducing CO_2 emissions without loosing performances. Three different adaptation strategies are automatically enacted by the Application Controller driven by the strategy selector module both at design-time and at runtime in order to improve the sustainability of the application while monitoring its performance. The feasibility and the effectiveness of our approach has been validated on an HPC application deployed in a cloud environment and resulted in a significant reduction of CO_2 emissions.

Further improvement of this work mainly goes in two directions. On the one hand, the Application Controller needs to be improved in order to reduce its overhead and it also needs to implement the possibility to enact more

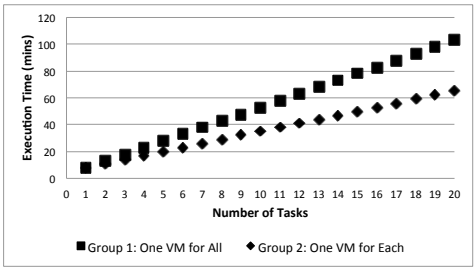


Fig. 12. Comparison of energy consumption among configurations.

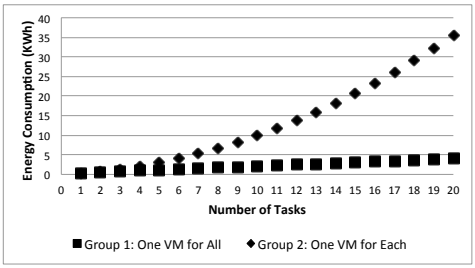


Fig. 13. Comparison of execution time among configurations.

than one adaptation action at the same time, as now they are taken separately. On the other hand, we would like to extend the approach to different kinds of applications (e.g., interactive) and to apply the same approach to other cloud-specific applications (e.g., map-reduce) that are quite similar to the presented case study.

Acknowledgments

This work has been partially supported by the ECO_2Clouds EU-FP7 Project (http: eco2clouds.eu) grant agreement n.318048. The authors would like to thank the ECO_2Clouds project partners for their support in setting up the testing environment for the experimental work here described.

References

[1] J. Huusko, H. de Meer, S. Klingert, and A. Somov, *Energy Efficient Data Centers: First International Workshop, E2DC 2012, Madrid, Spain, May 8, 2012, Revised Selected Papers*. Springer, 2012, vol. 7396.

[2] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya, "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems," *Advances in Computers*, vol. 82, no. 2, pp. 47–111, 2011.

[3] M. Vitali and B. Pernici, "A Survey on Energy Efficiency in Information Systems," *Journal on Cooperative Information Systems*, vol. 23, pp. 1–38, 2014.

[4] A. Nowak, F. Leymann, D. Schumm, and B. Wetzstein, "An Architecture and Methodology for a Four-Phased Approach to Green Business Process Reengineering," in *Proceedings of the ICT-GLOW 2011*, 2011, pp. 150–164.

[5] B. Pernici, M. Aiello, J. Vom Brocke, B. Donnellan, E. Gelenbe, and M. Kretsis, "What IS can do for environmental sustainability: a report from CAiSE'11 panel on green and sustainable IS," *CAISE*, vol. 30, no. 1, pp. 275–292, 2012.

[6] M. Meinshausen, N. Meinshausen, W. Hare, S. C. Raper, K. Frieler, R. Knutti, D. J. Frame, and M. R. Allen, "Greenhouse-gas emission targets for limiting global warming to 2°C," *Nature*, vol. 458, no. 7242, pp. 1158–1162, 2009.

[7] M. Uddin and A. A. Rahman, "Energy efficiency and low carbon enabler green IT framework for data centers considering green metrics," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 6, pp. 4078–4094, 2012.

[8] B. Li, J. Li, J. Huai, T. Wo, Q. Li, and L. Zhong, "EnaCloud: an energy-saving application live placement approach for cloud computing environments," in *IEEE International Conference on Cloud Computing*, 2009, pp. 17–24.

[9] A. Forestiero, C. Mastroianni, M. Meo, G. Papuzzo, and M. Sheikhalishahi, "Hierarchical approach for green workload management in distributed data centers," in *Euro-Par 2014*, 2014, pp. 323–334.

[10] D. Borgetto, M. Maurer, G. Da-Costa, J.-M. Pierson, and I. Brandic, "Energy-Efficient and SLA-Aware Management of IaaS Clouds," in *Proceedings of the 3rd International Conference on Future Energy Systems*, 2012.

[11] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[12] G. Katsaros, J. Subirats, J. O. Fitó, J. Guitart, P. Gilet, and D. Espling, "A service framework for energy-aware monitoring and VM management in Clouds," *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2077–2091, 2013.

[13] R. Giordanelli, C. Mastroianni, M. Meo, G. Papuzzo, and A. Roscetti, "Saving energy in data centers through workload consolidation," *white paper on www.eco4clouds.com*, 2014.

[14] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *Proceedings of the 2008 conference on Power aware computing and systems*, vol. 10, 2008.

[15] C.-H. Hsu, K. D. Slagter, S.-C. Chen, and Y.-C. Chung, "Optimizing energy consumption with task consolidation in clouds," *Information Sciences*, vol. 258, pp. 452–462, 2014.

[16] Q. Liang, J. Zhang, Y.-h. Zhang, and J.-m. Liang, "The placement method of resources and applications based on request prediction in cloud data center," *Information Sciences*, vol. 279, pp. 735–745, 2014.

[17] Z. Abbasi, M. Pore, and S. K. Gupta, "Impact of workload and renewable prediction on the value of geographical workload management," in *Energy-Efficient Data Centers*. Springer, 2014, pp. 1–15.

[18] Z. Liu, A. Wierman, Y. Chen, B. Razon, and N. Chen, "Data center demand response: Avoiding the coincident peak via workload shifting and local generation," *Performance Evaluation*, vol. 70, no. 10, pp. 770–791, 2013.

[19] B. Addis, D. Ardagna, A. Capone, and G. Carello, "Energy-aware joint management of networks and Cloud infrastructures," *Computer Networks*, vol. 70, pp. 75–95, 2014.

[20] A. Nowak, U. Breitenbücher, and F. Leymann, "Automating Green Patterns to Compensate CO2 Emissions of Cloud-based Business Processes," in *ADVCOMP 2014*, 2014, pp. 132–139.

[21] A. Nowak, T. Binz, F. Leymann, and N. Urbach, "Determining power consumption of business processes and their activities to enable green business process reengineering," in *EDOC 2013*. IEEE, 2013, pp. 259–266.

[22] P. Melià, M. Schiavina, M. Gatto, L. Bonaventura, S. Masina, and R. Casagrande, "Integrating field data into individual-based models of the migration of european eel larvae," *Marine Ecology Progress Series*, vol. 487, pp. 135–149, 2013.

[23] M. Lenzen, "Current State of Development of Electricity- Generating Technologies: A Literature Review," *Energies*, vol. 3, no. 3, pp. 462–591, 2010.

[24] ECO2Clouds Project Participants, "Good practices for Cloud Computing Energy Consumption and CO2 Emissions Optimisations," Tech. Rep., 2014. [Online]. Available: http://eco2clouds.eu/wp-content/uploads/D3.5-v1.0.pdf

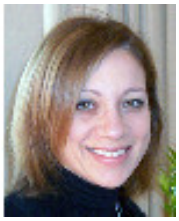
[25] M. Gribaudo, T. Ho, B. Pernici, and G. Serazzi, "Analysis of the Influence of Application Deployment on Energy Consumption," in *Proceedings of E2DC*, 2014.

[26] C. Cappiello, S. Datre, M. Fugini, P. Melia, B. Pernici, P. Plebani, M. Gienger, and A. Tenschert, "Monitoring and Assessing Energy Consumption and CO2 Emissions in Cloud-Based Systems," in *IEEE International Conference SMC*, 2013, pp. 127–132.

[27] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The Max-min Hill-climbing Bayesian network structure learning algorithm," *Machine learning*, vol. 65, no. 1, pp. 31–78, 2006.

[28] M. Vitali, B. Pernici, and U.-M. OReilly, "Learning a goal-oriented model for energy efficient adaptive applications in data centers," *Information Sciences*, vol. 319, pp. 152–170, 2015.

[29] M. Bertoli, G. Casale, and G. Serazzi, "JMT: Performance Engineering Tools for System Modeling," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, pp. 10–15, March 2009.



Cinzia Cappiello is an Assistant Professor in Computer Engineering at the Politecnico di Milano. Her research interests regard data and information quality aspects in service-based and Web applications, Web services, sensor data management and Green IT.

Nguyen Thi Thao Ho is a Ph.D student in Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. Her current research interests include queueing theory, stochastic modeling applied to Green IS and Cloud computing.

Barbara Pernici is full professor of Computer Engineering at the Politecnico di Milano. Her research interests include information systems design, adaptive information systems, service engineering, data quality, and energy efficiency in information systems. She has been elected chair of TC8 Information Systems of the IFIP WG 8.1 on Information Systems Design, and vice-chair of the IFIP WG on Services-Oriented Systems.

Pierluigi Plebani is Assistant Professor at Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano where he also received the PhD degree in Information Engineering. He currently belongs to the Information Systems group and his research interests concern Green IS, Cloud Computing, and IS for energy saving in buildings.

Monica Vitali received the Ph.D. in Information Technology from the Politecnico di Milano, Italy, in 2014. She is currently research assistant at the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano. She is interested in the topic of Energy Efficiency and sustainability in cloud and data center from an Information Systems perspective, in adaptive and self-adaptive systems and services, and in Machine Learning techniques for adaptation.