Energy-Aware Process Design Optimization

Cinzia Cappiello, Pierluigi Plebani, and Monica Vitali
Dipartimento di Elettronica, Informazione e Bioingegneria - Politecnico di Milano
Piazza Leonardo da Vinci, 32 - 20133 Milan, Italy
Email: cinzia.cappiello@polimi.it, pierluigi.plebani@polimi.it, vitali@elet.polimi.it

***Abstract*—Cloud computing has a big impact on the environment since the energy consumption and the resulting CO_2 emissions of data centers can be compared to the worldwide airlines traffic. Many researchers are addressing such issue by proposing methods and techniques to increase data center energy efficiency. Focusing at the application level, this paper proposes a method to support the process design by optimizing the configuration and deployment. In particular, measuring and monitoring suitable metrics, the presented approach provides a support to the designer to select the way in which it is possible to modify the process deployment in order to continuously guarantee good performance and energy efficiency. The process adaptation can be required when inefficiencies occur or when, although the system is efficient, there is still room for improvements.**

I. Introduction

Nowadays, researchers are more and more analyzing the evolution and diffusion of cloud computing from a perspective on sustainability. Assessments of the environmental impact of cloud computing reveal that data centers consume an extremely high amount of electricity [1]. Many contributions show that the reduction of CO_2 emissions, and energy consumption can be achieved by considering several initiatives such as the reduction of the cooling power consumption, the utilization of green energy sources, server consolidation and virtualization, and the utilization of greener machines. In the proposed approaches, less attention has been paid to the analysis of the impact of the design and deployment of the running applications on energy efficiency.

Considering this application viewpoint, in this paper we propose an approach for the optimization of the application design and deployment. The optimization is driven by three main factors: energy consumption, CO_2 emissions, and the satisfaction of performance requirements measured by suitable metrics. The idea is to follow a process design life-cycle where firstly, a business process (BP), i.e., a set of tasks, is designed and deployed on the basis of the characteristics of the system and of the designer expertise. Then, the system continuously monitors the application in order to evaluate if the requirements are satisfied or if there is anyway room for improving the considered metrics. If changes are needed, the designer should enact appropriate adaptation actions, that will take place in the next execution of the BP and that are able to address the discovered inefficiencies, by modifying the workload distribution between different versions of the considered BP. The approach described in this paper aims to support the designer in the interpretation of the monitoring results and in the selection of the adaptation action.

The paper is organized as follows. Section II discusses previous contributions and highlights the innovative aspects of the presented approach. Section III provides a general overview of the approach that is formalized in the following sections: Section IV focuses on the process design and deployment phases, Section V introduces the relevant metrics, Section VI describes the method that supports the designer in the selection of the adaptation action able to optimize the process in terms of performance and energy efficiency. Finally, Section VII validates the approach with an example and Section VIII outlines some possible future work.

II. Related Works

Energy efficiency and the control of CO_2 emissions are a living matter and, in the recent years, many researcher have addressed these aspects at very different granularity levels. The main issue when dealing with energy efficiency and CO_2 emissions consists in finding a direct or indirect way to measure them. Moreover, the improvement in energy efficiency and the CO_2 emissions reduction cannot overlook the impact over performance. A tradeoff between greenness and performance has to be handled in order to respect functional requirements of the provided services. Metrics for describing these two aspects have been formalized in [2] and in [3]. In the former work, the well known Key Performance Indicators (KPIs) for evaluating Quality of Service (QoS) are integrated with a set of energy related metrics called Key Ecological Indicators (KEIs). The latter work introduces the concept of Green Performance Indicators (GPIs), with a role similar to the KEIs. In [4], authors put lights on the importance of considering also how the available resources are used in the delivery of the service, introducing usage centric metrics to this extent. This concept is developed also in [5], where usage centric metrics are defined at the server and the virtual machine level.

Measuring energy is not an easy task, especially when real time data are required and when energy has to be measured for each single server separately. Several approaches consider CPU as the principal contributor to power usage in a server and propose models for computing energy starting from the amount of CPU used on the server [6] [7] [8] [9] [10]. These models compute power usage using a linear relation between CPU usage and power consumption. Parameters of the model are the power consumption values for the server when idle and when at the peak load. The only variable of the model is the CPU load. It has been demonstrated that even a linear approximation of this model is effective in estimating the real power consumption [11]. An interesting aspect is that even in idle state a server consumes more than the 60% of its peak load power. This is why an optimization, even at the application level can significantly contribute to the improvement of energy efficiency and consequently to the reduction of CO_2 emissions, by optimizing the way in which resources are used.

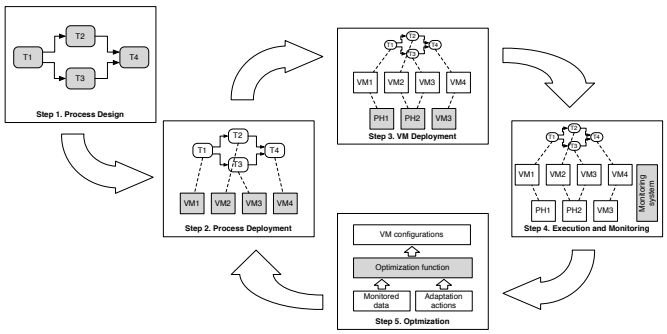

Fig. 1. Energy-aware process design optimization approach

Some studies face the problem of energy efficiency and greenness with a focus at the process level. In [12], authors state that an improvement is possible only if the approach includes a process re-engineering step. Four areas of intervention are identified: process design, measuring, improvement and change, and implementation. Other approaches use process annotation: the business process is enriched with additional information regarding its behavior in terms of energy efficiency [2] [13]. In [14], the authors define a set of patterns for improving energy efficiency of a BP by adding a green dimension to the traditional four dimensions: cost, time, flexibility and quality. Other scholars focus their attention on reconfiguration of the service. The general approach consists in decomposing the process in sub-tasks that can be substituted with similar but greener ones [15] [16] [17].

The approach proposed in our paper has been inspired by the service composition and evolution as we aim to consider variants and adaptation of the tasks that compose a service-based business process, also considering green aspects. The evaluation is based on the estimation of the workload of each task. Given the workload, we will use a small selection of the metrics defined in literature to assess the state of the system and to compare different solutions in terms of energy efficiency, CO_2 and QoS. On the basis of these evaluations and a set of possible actions to enact for improving the process design and deployment, we support the designer decision process: we alert designers if improvements are needed and we also suggest the appropriate changes to apply.

III. Proposed Approach

The optimization of the design of energy-aware BP follows the cycle shown in Fig. 1. First of all, we assume that the application designer already knows which are the tasks that a process should perform and the order in which these tasks have to be executed (Step 1). The BPs we are considering in our approach can be seen as processes composed of different tasks running on Virtual Machines (VMs). A critical aspect about our approach concerns the configuration of such VMs (Step 2). Next, the VMs are deployed on physical hosts that live in a cloud infrastructure (Step 3). The problem of deploying the VMs on the physical host is out of the scope of this paper, as we rely on solutions that have been proposed in the literature (e.g., [18] [19] [20]). Once the VMs of the process have been deployed, the execution commences and the monitoring system starts gathering information about the process in terms of performance and energy impact (Step 4). Such properties are inferred by measuring specific metrics (e.g., CPU usage, response time, energy consumption). At the end of the execution, the monitored data combined with the supported adaptation actions will drive the optimization of the BP (Step 5).

With such system, continuous monitoring and data analysis allow application designers to intervene by means of adaptation actions when changes are required (for instance, when terms in Service Level Agreement are no longer satisfied). The application designer can modify the application deployment in different ways. In fact, we consider a BP with different tasks that can be assigned to different VMs adopting various configurations. The BP execution could be changed in different ways. For example, it is possible to modify the VMs configuration or the way in which the tasks of the process are deployed on the VMs, it is also possible to duplicate or remove duplicated VMs and reallocate the workload. More details about the actions that we consider in this paper are provided in Sec. IV.

The plethora of possible actions makes difficult the decision of the application designers. Indeed, they should select the more favorable adaptation action in terms of performance and energy efficiency but such decision requires a complete knowledge of the impact that the execution of the different adaptation actions has on the system (i.e., monitored metrics). The proposed approach aims to support the designer in such a decision making process by suggesting the most suitable adaptation action. The decision is based on estimations based on previous executions. The methodology used to achieve this goal is described in details in the following sections, according to the five steps identified in Fig. 1.

IV. Process Design and Deployment

The first phase consists in the design and deployment of the BP and includes the first three steps shown in Fig. 1.

A business process BP is a collection of tasks $\{T_i\}$ which are executed providing a valuable output. Tasks are linked by control structures (e.g., sequence, choice, parallel) and consequently are invoked with a given probability P_i. Such probability is an information that reflects the typical interaction of the user with the process. It is usually obtained from the analysis of historical data derived from previous executions. In case of first execution these values will be defined by the application designer based on his/her experience.

$$BP = \{\langle T_i, P_i \rangle\}$$

Assuming that WL_{BP} is the total workload of the business process, the workload for each task is given by:

$$WL_{T_i} = WL_{BP} \cdot P_i$$

It is necessary to consider that for the execution of each task, one or more variants are available, i.e. functionally equivalent tasks. As the workload of the task can be distributed among the variants, the percentage of workload that will be assigned to each variant is defined as:

$$T_i = \{t_{i,j}\} = \{\langle V_{i,j}, D_{i,j} \rangle\} \text{ where } \sum_j D_{i,j} = 1$$

where $t_{i,j}$ is a variant of the task, $V_{i,j}$ is the configuration of the VM hosting the task and $D_{i,j}$ is the percentage of workload of the task T_i assigned to the variant $t_{i,j}$.

A configuration of a task variant consists of a specific configuration of the VM on which the task runs. This means that the same task can run on different VMs:

$$V_{i,j} = \langle qCPU, qRAM, qStorage \rangle$$

where $qCPU$, $qRAM$, and $qStorage$ indicates the number of CPUs, the amount of memory, and the amount of storage available for the VM on which the task runs.

In our approach, the distribution among the variants of the same task, is supported by the optimization algorithm proposed Sec. VI. This distribution will affect the workload submitted to each variant for a given task as:

$$WL_{t_{i,j}} = WL_{T_i} \cdot D_{i,j}$$

In the process deployment phase, each variant $t_{i,j}$ is assigned to a dedicated $VM_{i,j}$, taking into account its configuration $V_{i,j}$:

$$t_{i,j} \xrightarrow{assigned\_to} VM_{i,j}$$

and each VM is than deployed on a physical server PH_l:

$$VM_{i,j} \xrightarrow{deployed\_on} PH_l$$

As a consequence of the adoption of variants at task level, also variants at business process level can be defined as the application of one or more adaptation actions A_z. An action A_z is a complex action originated from the composition of one or more atomic actions over the BP. An action can be expressed as $A_z = \{a_{z,v}\}$, where $a_{z,v}$ has an impact on a single task belonging to the business process. We can define atomic actions as $a_{z,v} \in \{Add(t_{i,x}), Del(t_{i,x}), Mod(t_{i,y}, t_{i,x})\}$, where:

$Add(t_{i,x}) : T_i \rightarrow T_i'$ where $T_i' = T_i \cup t_{i,x} \wedge \sum_j D'_{i,j} = 1$

$Del(t_{i,x}) : T_i \rightarrow T_i'$ where $T_i' = T_i - t_{i,x} \wedge \sum_j D'_{i,j} = 1$

$Mod(t_{i,y}, t_{i,x}) : Del(t_{i,y}) + Add(t_{i,x})$

In addition to these actions, also the Nop action has to be considered as the application designer can decide to avoid any modifications.

Thus, applying a composite action A_z to a business process BP we obtain a variant BP' consisting in a different set of task variants and in a different workload distribution among them: i.e., $BP \xrightarrow{A_z} BP'_z$. Thus, the goal of the optimization described in Sec. VI is to identify the variant that better improves both the performances of the process and the energy efficiency assessed by using the metrics introduced in Section V.

V. Process Execution and Monitoring

Monitoring is fundamental to gather data about performance and energy efficiency of an application. Such properties are inferred by measuring specific metrics. In details, the set of metrics considered in this work includes:

- *VM resource usage parameters*: *CPU* and *memory utilization* percentages for a running application over a run time interval are calculated by using the ratio between the amount of used and allocated CPU/memory. The resource usage parameters also include *IOPS* that is defined as the total number of I/O operations per second.
- *Throughput*: number of performed transactions over a period of time.
- *Energy consumption*: refers to the energy consumed by the application in a specific period of time.
- *CO_2 emissions*: quantity of CO_2 emitted by executing the specific application.
- *Application performance*: is the ratio between the throughput of the VM in a certain time interval and the energy consumed.
- *Response time*: refers to the time spent to serve a single request.

More formally, each metric can be modeled as:

$$m_h =< name, V > \quad (1)$$

where the name identifies the metric and V corresponds to the set of admissible values. It is represented by its extremes, i.e., $V = [v_{min}, v_{max}]$.

Metrics can be collected at the BP level or at the VM/task variant level. For each of these levels, users can operate a restriction on the admissible range of values for each m_h, in order to require a specific energy consumption, or performance. This restriction can be formalized as:

$$r_h(c) =< m_h, req, w > \quad (2)$$

where $c \in \{BP, \{VM_{i,j}\}\}$, and $req \in m_h.V$. More precisely, req represents the restriction on the range of admissible values for metric m_h computed over the component c. This restriction corresponds to the values required by the user for the given metric. Finally $w \in [0, 1]$ is the weight that defines the relevance of the metric for the specific user (or application) and provides a prioritization of the requirements. Such requirements and constraints can be used to derive the initial deployment of applications as well as provide impetus for run-time adaptation of application deployment. Note that the constraints can be *hard constraints* if the user considers that they must be satisfied or *soft constraints* if the user also accepts a constraint violation. Note that for the hard constraint the relevance weight will be $r_h(c).w = 1$ while for the soft constraints $r_h(c).w \leq 1$.

Setting the set of values $m_h(c).req$, the users identifies the values related to desired (satisfaction zone) and undesired (alarm zone) behavior for each metric. Also a warning zone is defined slightly before the alarm zone, and its range of values is identified automatically on the basis of the designer requirements. In fact, such a range of values depends on the strategy that the designer wants to adopt for the system improvement. Two possible approaches are possible: proactive and reactive. Adopting a proactive approach implies that a warning is triggered before reaching a real violation, so that the system can react in advance. It requires the usage of an extended warning zone between the satisfaction and the alarm zone. On the contrary, using a reactive approach implies a warning zone very close to the alarm threshold. In this case the

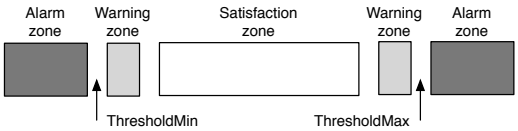

Fig. 2. Variables requirements restriction intervals

system reacts only when it is significantly close to a violation. A general representation of the three zones for the metrics is shown in Fig. 2, even if for some metrics either *ThresholdMin* or *ThresholdMax* can be defined.

VI. Process Optimization

In the approach presented in this paper, inefficient situations are addressed by an adaptive behavior. The selection of the best adaptation action to trigger requires performing two sequential steps: a local and a global optimization.

Local optimization

This step requires the analysis of each process variant BP'_z in order to define the way in which the workload should be distributed among the tasks defined in the process variants. In fact, since the BP variants mainly consist in adding or deleting task variants, a redistribution of the workload is always needed. Such redistribution should be performed by considering that the workload has an impact on all the metrics considered in this paper. This statement is confirmed by existence of a documented positive correlation between workload and CPU load [21]. Furthermore, it has been demonstrated that CPU load is directly proportional to the response time [7], the energy consumption (see Sec. II), and, as a consequence, the CO_2 emissions and application performance. The existence of a correlation between the workload distribution and memory and IOPS depends on the analyzed business process. Anyway, it is possible to affirm that a change in the workload distribution affects such two variables only in two ways: (i) the metrics are unaltered (there are no dependencies with the workload) (ii) the variables are dependent on the workload in a directly proportional way.

On the basis of these considerations, our goal is to find the set of workload distributions for each task variant $\{WL_{t_{i,j}}\}$ that maximizes the following optimization function:

$$max\ F(\{WL_{t_{i,j}}\}) = [\Delta CO_2(\{WL_{t_{i,j}}\}), \Delta E(\{WL_{t_{i,j}}\}), \sum_{c=1}^{C}\sum_{h=1}^{H} r_h(c).w\ SSR(\{WL_{t_{i,j}}\})] \quad (3)$$

where ΔCO_2 and ΔE are related to the minimization of emissions and energy consumption. They are calculated as the difference between the quantity of emissions and consumed energy before and after the execution of the BP variant. The number of satisfied soft requirements (SSR) refers to weighted sum of the satisfied requirements, considering their relevance for the users. The optimization problem should be solved by considering the hard constraints that must be satisfied to guarantee the system effectiveness and efficiency.

The solution of the optimization problem $\{WL^*_{t_{i,j}}\}$ is the vector of workload distribution on the different tasks and related variants that maximize the defined function. Such vector will be calculated for each considered adaptation action A_z. In this way we will have Z workload vectors and, considering such vectors, we store in the decision matrix DM the vector X_z that contains the estimated values of the three attributes of function F after the execution of the adaptation strategy (that is in correspondence of the workload vector $\{WL^*_{t_{i,j}}\}$):

$$X_z = F(\{WL^*_{t_{i,j}}\}) = [\Delta CO_2(\{WL^*_{t_{i,j}}\}), \Delta E(\{WL^*_{t_{i,j}}\}), \sum_{c=1}^{C}\sum_{h=1}^{H} r_h(c).w\ SSR(\{WL^*_{t_{i,j}}\})] \quad (4)$$

Global optimization

The global optimization aims to identify the best adaptation action to perform in order to maximize the goal function among the $\{A_z\}$ set. We consider techniques from the Multiple Criteria Decision Making theory [22]. Each alternative is associated with a vector X_z that contains the three estimated values of the attributes after the execution of the adaptation strategy A_z. Such estimation can be performed on the basis of the results of previous executions as discussed later in this section. The different vectors compose the decision matrix DM that supports us in the strategy selection. The selection is carried out by executing two main steps. First of all, the elimination of all the dominated solutions is required. A solution X_m is dominated if there is another solution which has better results for all the considered attributes X_k: $X_{k,n} \geq X_{m,n}$ for each n (where n is the considered attribute and $n \in \{1, 2, 3\}$ in our case). Secondly, we select the solution. The decision broker will consider all the non-dominated alternative solutions. As a first step, all the attributes are normalized in the range of values $[0, 1]$, so that they are comparable even if expressing different goals. Then, the selection of the most suitable alternative can be performed by using one of the following methods [22]:

- MAXIMIN rule: for each non-dominated solution X_z, the attribute with the lowest value is identified. Results for all the solutions are compared and the one with the highest "lowest value" is preferred.
- MAXIMAX rule: for each non-dominated solution X_z, the attribute with the highest value is identified. Results for all the solutions are compared and the one with the highest "highest value" is preferred.

□

Both global and local optimizations rely on the computation of the *application independent* terms, i.e., ΔCO_2, ΔE, and *application dependent* terms, i.e., the number of soft constraints that can be satisfied.

Application independent estimation

The estimation of energy consumption and CO_2 emissions are considered application independent as these parameters mainly depend on the CPU load. For this reason, we can estimate these two parameters analyzing physical servers having the same average CPU load of the application that we are

going to deploy, regardless of the applications that are actually running.

In order to evaluate the factor $\Delta E(\{WL_{t_{i,j}}\})$, for each $WL_{t_{i,j}}$, first we consider logs of previous executions of the task variant $t_{i,j}$ in order to find the relationships between workload and CPU load. Thus, given a $WL_{t_{i,j}}$ we retrieve the correspondent $CPULoad(t_{i,j})$. Once we have this value and we know the characteristics of the physical host PH_l (i.e., peak power) on which the $VM_{i,j}$ is deployed, we estimate the energy by gathering the power consumed by a similar task deployed on a similar host. Such a similarity is computed by comparing the CPU load of the analyzed task and the characteristics of the host where it runs, to the hystorical data of the hosts usage, looking for hosts with similar characteristics that run applications with a similar CPU load.

The evaluation of the CO_2 emissions is based on the emission factors (gCO_2e/kWh) provided by the national grids. Considering the consumed energy $E(t_{i,j})$ CO_2 emissions is computed multiplying the energy consumed by the emission factor.

Application dependent estimation

In this section we estimate the last part of the objective function described in Eq. 3, i.e., the number of metrics that will be satisfied SSR.

As before, we try to predict the trends of the state of the variables for a given distribution of the load between the available variants of each task. In this case we can not use the similarity criteria introduced in the previous paragraph. In fact, the value of the variables is strictly dependent on the application and it is very unlikely to find examples of the variables behaviors in historical data for all the possible configurations of the process.

Instead of using similarity, we propose an alternative technique based on correlations between variables. To this extent, we build a Bayesian Network [23]. This network expresses dependencies among variables using a directed acyclic graph representation, where a parent-child relation between two nodes defines a causal dependency of the child from the parent. A Conditional Probability Table (CPT) at each node expresses the probability for the node to get each one of the allowed values given the state of its parents. A conditional independence relation among variables representable as nodes in a Bayesian Network is also defined [23]:

Definition 1: the conditional independence property between nodes in a Bayesian Network implies that a node is independent on all other nodes given its parents, children and children's parents.

These properties of a Bayesian Network allow a simple inference mechanism for predicting the most likely state of a subset of variables given the state of all the other nodes in the network.

The usefulness of a Bayesian Network representation for our problem is twofold. First of all, it allows us to establish dependence and independence relations among variables. In this way, if we know that a subset of variables changes, we can predict which are the variables that will be influenced and

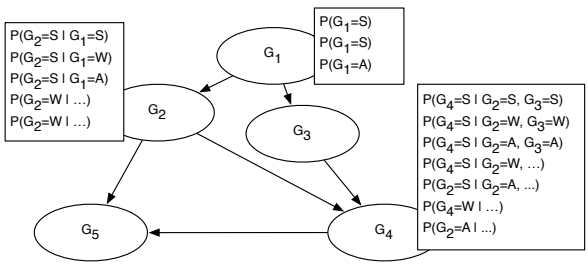

Fig. 3. Bayesian Network expressing relations among variables states

which are the ones that remain the same. This allows us to restrict the problem and to focus only on a subset of variables. Second, even if the network is not able to predict the exact value of the variables, it can be used to predict future states of the requirements. In order to use a Bayesian Network representation, variables have to be transformed into discrete values, representing the satisfaction of the requirements defined over them (see Sec. V and Eq. 2). In particular, each variable will be expressed with 1 out of 5 values indicating if it is satisfied (satisfaction zone), if it is violated (alarm zone), or if it is still satisfied but near to be violated (warning zone) as shown in Fig. 2. As an example, the CPU load has two alarm and warning intervals, since it can be violated either if its value is too high or too low. Indeed, if the CPU usage of a server is too low, it is inefficiently used, while if it is too high it can have a negative impact on performance. An example of a Bayesian network for our context is shown in Fig. 3 [1], where each node is a variable and the CPTs contain parent dependent state probabilities for each node. Through the Bayesian Network, we are able to predict future states of the variables given the modifications in the states of other variables.

The first step consists in building the network, by discovering the relations among variables. Most of the time, Bayesian Networks are manually created by experts. However, several techniques are available in the state of the art to learn it from historical values of the monitored variables. Here we use an approach similar to the one described in [24] to learn the structure of the network. Then, we learn CPTs using a Maximum a Priori Estimation approach [25].

Once the network is created, it can be used to evaluate a particular variant of the process. The evaluation procedure starts when a request arrives. The input of the evaluation process is a set of values of CPU load for the tasks that we want to modify. In order to estimate future values, we use the inference properties of the Bayesian Network. The input of the inference has to be the set of values for all the nodes in the network where the values of the node that we want to estimate are hidden. So, starting from the input, the algorithm works as follows:

1) The set of input values I are converted into discrete values between 1 and 5 using the defined requirements.
2) The set of variables V that are influenced by the set of input variables I are selected using the conditional independence rules of a Bayesian Network (see Def. 1).

[1] S = Satisfaction Zone, W = Warning Zone, A = Alarm Zone

3) An input vector $InfInput$ for the inference procedure is defined as follows:
   - take the vector with the state of the variables obtained by the monitoring system.
   - modify the values of the variables in I with the values computed at step 1.
   - hide the variables in V from the vector.
4) Run inference with input vector $InfInput$ obtaining the most likely states for each variable in V associated with a probability P_V for the variable to get that state.
5) Return as output a binary vector $InfInput'$ obtained from $InfInput$. A value equals to 1 means that the requirements upon the variable are satisfied, while a value equals to 0 means that they are violated. An additional vector $Conf$ of the same dimension is also created with the values P_V for each predicted variable. The probability value for all the variables that do not belong to V will be 1, because they are considered as constant.

The probability value returned by the inference procedure is used as a confidence parameter in the estimation of the value for the third attribute of the optimization function.

VII. Validation

To provide a validation of our approach, we consider a sample BP (see Fig. 4), where a set of tasks are performed to buy goods using an e-commerce Web site. This BP involves several VMs where the tasks are installed and executed. Using the notation introduced in Sec. IV, our BP is defined as:

$$BP = \{\langle BS, 1.0\rangle, \langle RO, 0.90\rangle, \langle VO, 0.95\rangle, \langle RP, 0.855\rangle, \langle IP, 0.855\rangle \langle CA, 0.045\rangle\} \quad (5)$$

To execute the business process and monitor the metrics introduced in Section III, we relied on the BonFIRE infrastructure[2]. For the sake of simplicity, and considering that the BonFIRE platform is not now able to monitor the energy related metrics and the I/O usage:

- As the memory usage does not change significantly during the experiment, in this validation we focus on the CPU Usage.
- For the energy-related metrics, we rely on the results published in [8] to estimate the power of a physical host based on the CPU load.
- We assume that each task of the BP lives on a dedicated VM and that the resulting VMs are deployed on different physical hosts.
- Although on a physical host, in addition to our VMs, other VMs of other applications are running, we assume that the CPU load of these external VMs remains constant as we are interested in the variation of consumption of our business process.
- The VMs can be hosted on data centers sited in France and England where the average CO_2 emission factors are, respectively, 146 gCO2e/kWh [3] and 567.17 gCO2e/kWh [4].

The two possible configurations of the VM used for our validation are:

- $VM^{2,2}$ = ⟨ 2 cores, 2 GBytes, 10Gbytes ⟩.
- $VM^{4,4}$ = ⟨ 4 cores, 4 GBytes, 10Gbytes ⟩.

where a Linux Debian Squeeze v.5 distribution is installed, along with Oracle Glassfish v3.1 to run the activities composing our BP.

Creating the Bayesian Network

The creation of the Bayesian Network requires a knowledge about previous executions of the BP. At least during the first design of our BP this kind of information is not available. Nevertheless, as common behavior in case of service-based business processes, we can assume that other instances of the same tasks composing our BP have been previously used in other processes.

Starting from the historical data collected from the monitoring system, and using the procedure described in Sec. VI, we obtain a Bayesian Network describing the relations among variables in the system. We consider to have metrics at two levels:

- Task Level: metrics at this level are CPU load (CPU) of the VM, throughput (TH) and response time (RT) of the task variant running on the VM.
- BP level: metrics at this level are response time (RT), throughput (TH), and application performance (AP) of the whole process.

The network obtained for the considered example is shown in Fig. 5, representing a simplified version of the real network. In the picture, only a general task is represented, while in the real network the same structure is replicated for each of the tasks and their variants in the BP. In our example, we have six tasks. The response time of all the tasks and their variants contribute to the response time of the whole process, and the same can be stated for the throughput.

Initial configuration

The application designer decides to initially run the six tasks on six VMs having the same configuration (i.e., the $VM^{2,2}$). For these VMs, the designer initially does not add any variant, so the workload of a given task is entirely sent to the unique available VMs, thus:

$$BS = RO = VO = RP = IP = CA = \langle VM^{2,2}, 1.0\rangle$$

As hard constraint, we have $RT(BP) \leq 25min$. As soft constraints, we have:

- $\forall\, T_i\ [70\% \leq CPU(T_i) \leq 90\%\ ;\ RT(T_i)]$
- $TH(BP) \geq 5tpm$
- $AP(BP) \geq 8.5tpKWh$.

[2] http://www.bonfire-project.eu/

[3] http://www.eumayors.eu/IMG/pdf/seap_guidelines_en.pdf

[4] http://www.defra.gov.uk/publications/

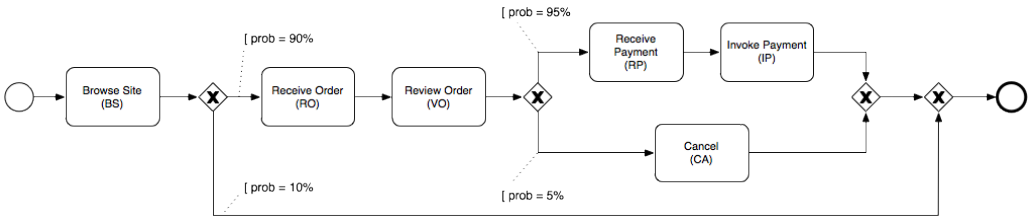

Fig. 4. Business process sample

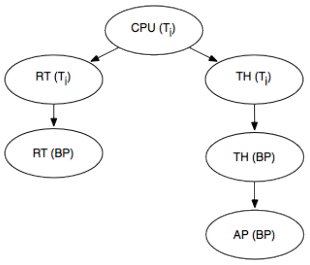

Fig. 5. Bayesian Network created for the considered BP

TABLE I. VALIDATION RESULTS

Metric		*First*	*Estimation*	*Actual*
Average CPU load	BS	88.24%	70.59%	74.06%
	RO	20.84%	41.68 %	62.2 %
	VO	68.38%	82.05%	85.5 %
	RP	57.58%	63.3%	34.93 %
	ID	42.82%	47.10%	48.23%
	CA	0.2%	0.2%	1.93 %
Average BP Throughput		4.34 tpm	4.7 tpm	6.1 tpm
Average BP Response Time		22.8	20.0	17.67
Energy Consumption		31.974 KWh	27.193 KWh	26.731 KWh
CO_2 emissions		4,668 gCO_2e	4,964 gCO_2e	4,903 gCO_2e
Application Performance		8.14 tpKWh	10.37 tpKWh	13.69 tpKWh

The warning zone is set to be 5% away from the alarm threshold for all the metrics.

Assuming that all of these VMs have been deployed in the French data centre (the deployment of the VMs is under the control of the cloud hypervisor), we obtain the results reported in the second column of the Table I (i.e., "First").

Adapting the deployment

Based on the results of the initial configuration, the application designer aims to improve the quality of the process by adapting the BP. According to the adaptation mechanisms presented in Section IV, the application designer plans to modify the configuration of the BS task by doubling both the number of CPUs and the amount of memory to a BS' variant, i.e.:

$$BS' = \langle VM^{4,4}, 1.0 \rangle$$

As we assume that in France there are no available VMs which such features, this task will run in England. So, the actions that the application designer takes into account are $Mod(BS, BS')$ and Nop.

Considering the first adaptation action, as in this example we assume that the application designer proposes only this variant, the solution of the local optimization (see Eq. 3) is simply such a variant and all the workload for the first task of the BP is sent to BS'. Indeed, what we would like to demonstrate in this validation is how the enactment of an adaptation action can influence the values of the monitored metrics. As a consequence, the two alternatives to be considered are X^{Nop} and X^{Mod}.

About the alternative X^{Nop} the values for the ΔCO_2^{Nop}, ΔE^{Nop} result 0, since no modification has been done to the process and this does not affect the energy metrics. About the number of satisfied soft constraints, these values are computed considering the data monitored during the previous execution and constraints over the indicators. About CPU usage, only the first task satisfies the given constraints, while constraints about response time and throughput at the task level are satisfied for all tasks except for the first one in which response time is higher than the constraint and throughput is lower. At the BP level, throughput is not satisfied and the same for application performance.

The computation of the values for the alternative X^{Mod} follows the application independent and dependent estimation. These estimations are based on the monitoring data associated with previous execution of the tasks composing our BP that, as previously said, can be done for other instances in different processes. The result of this estimation is reported in the third column of Table I (i.e., "Estimation").

About the application independent estimation, the energy consumed by the application requires the knowledge of the CPU load. Looking at the data about previous executions, we search for instances of BS task running on a $VM^{4,4}$ and we discover that, having the same workload as in the initial execution: (i) the CPU load of the BS task is reduced by the 10% in the average, and (ii) the task runs faster by 20%. This change in the CPU load and in the response time will also affect the remaining tasks. For this reason, following the structure of the BP, we analyze the previous executions to estimate the CPU load given the new workload. For instance, having a BS faster by 20% means that the workload rate for the RO increases of the same amount (the effect of the branch is considered insignificant) and looking at the data, we understand how, under this circumstances, the CPU load for the RO double (from 20.84% to 41.68%). The same kind of assumption has been also done for the other tasks, but for the CA that we know it is invoked very rarely for which the CPU load remains unchanged. For the sake of simplicity, these estimations are intentionally rough as a real estimation requires a lot of assumptions and data available to the application designer. For this reason, in the real scenario, we can assume that such data, along with the experience of the designer, will result in a more accurate estimation. Having the CPU load for each of the tasks, the computation of the energy consumed by the application follows the assumptions in [8].

Exploiting the Bayesian Network computed starting from the monitored data (see Fig. 5), we estimate the number of satisfied indicators for the CPU loads estimated in column 3 of Tab. I. According to that, the state of CPU(BS) changes

from warning to satisfied and CPU(VO) changes from alarm to warning. All other values for CPU(T_i) are not affected. This impact RT(BS) and TH(BS), now having a high probability of being satisfied, while RT(VO) and TH(VO) were already satisfied and the improvement of CPU(VO) does not change this state. The modification of RT(BS) and TH(BS) affects positively TH(BP) with a consequent improvement of AP(BP). As a consequence, the total count of satisfied indicators increases of 5, counting indicators in the warning state as satisfied. The hard requirements about RT(BP), already satisfied in the previous configuration, is still satisfied.

As a consequence, the values of the objective function for the two alternatives are:

$$X^{Nop} = (\Delta CO_2^{Nop}, \Delta E^{Nop}, \#SSR^{Nop}) = (0, 0, 12)$$
$$X^{Mod} = (\Delta CO_2^{Mod}, \Delta E^{Mod}, \#SSR^{Mod}) = (-296, 4.78, 17)$$

As can be observed, neither of the two solutions is dominant. In order to decide the best of the two, one of the techniques described in Sec. VI has to be applied. For example, using the MAXIMIN approach, firstly, for each alternative the minimum value among the three attributes is considered, i.e., for X^{Nop} action $min(0, 0, 12) = 0$ while for the X^{Mod} action $min(-296, 4.78, 17) = -296$. Secondly, the solution to prefer is the one that is associated with the highest number: in our case the maximum between 0 and -296 is 0 and then the X^{Nop} solution is considered as suitable (i.e., doing nothing). This reflects the higher importance given at the CO_2 emissions with respect to the performance and energy consumption.

VIII. Final Remarks

In this paper we presented an approach for supporting the energy-aware adaptation of business processes at design time driven by an optimization problem that takes into account both infrastructural, applicative, and environmental standpoints. In particular, the adaptation actions available to the application designer rely on the definition of variants for the tasks composing the business process.

This approach is now heavily based on the assumption that monitored data about previous executions are available, and information able to give the values for the interesting metrics are all included in this knowledge base. For future development of our approach, we aim to relax this assumption, trying to understand if we can infer the values of the metrics in case the data needed to compute them analytically are not present.

Acknowledgment

This work has been supported by the ECO_2Clouds project (http://eco2clouds.eu/) and has been partly funded by the European Commission's IST activity of the 7th Framework Programme under contract number 318048.

References

[1] Greenpeace, "How Clean Is Your Cloud?," tech. rep., 04 2012.

[2] 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 ICT-GLOW 2011, Toulouse, France*, vol. 6868 of *LNCS*, pp. 150–164, Springer-Verlag, 2011.

[3] A. Kipp, T. Jiang, M. Fugini, and I. Salomie, "Layered green performance indicators," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 478–489, 2012.

[4] R. P. Larrick and K. W. Cameron, "Consumption-Based Metrics: From Autos to IT," *Computer*, pp. 97–99, 2011.

[5] D. Chen, E. Henis, C. Cappiello, *et al.*, "Usage centric green performance indicators," in *Proceedings of the Green Metrics 2011 Workshop (in conjunction with ACM SIGMETRICS 2011)*, 2010.

[6] D. Meisner, B. T. Gold, and T. F. Wenisch, "Powernap: eliminating server idle power," in *Proc. of the 14th Int'l conference on Architectural support for programming languages and operating systems*, ASPLOS XIV, (New York, NY, USA), pp. 205–216, ACM, 2009.

[7] M. Pedram and I. Hwang, "Power and Performance Modeling in a Virtualized Server System," in *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, pp. 520–526, 2010.

[8] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing*, pp. 1–21, 2010.

[9] 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: Where Energy, Computing and Communication Meet*, p. 25, 2012.

[10] G. Katsaros, J. Subirats, J. Oriol Fitó, J. Guitart, P. Gilet, and D. Espling, "A service framework for energy-aware monitoring and VM management in Clouds," *Future Generation Computer Systems*, 2012.

[11] 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.

[12] S. Seidel, J. vom Brocke, and J. C. Recker, "Call for Action: Investigating the Role of Business Process Management in Green IS," *Sprouts: Working Papers on Information Systems*, vol. 11, no. 4, 2011.

[13] C. Cappiello, M. G. Fugini, A. M. Ferreira, P. Plebani, and M. Vitali, "Business Process Co-Design for Energy-Aware Adaptation," in *Intelligent Computer Communication and Processing (ICCP), 2011 IEEE International Conference on*, pp. 463–470, 2011.

[14] A. Nowak, F. Leymann, D. Schleicher, D. Schumm, and S. Wagner, "Green Business Process Patterns," in *Proceedings of the 18th Conference on Pattern Languages of Programs*, ACM, 2011.

[15] K. Hoesch-Klohe and A. Ghose, "Carbon-Aware Business Process Design in Abnoba," *Service-Oriented Computing*, pp. 551–556, 2010.

[16] A. Mello Ferreira, K. Kritikos, and B. Pernici, "Energy-Aware Design of Service-Based Applications," *Service-Oriented Computing*, pp. 99–114, 2009.

[17] A. De Oliveira, G. Frederico, and T. Ledoux, "Self-optimisation of the energy footprint in service-oriented architectures," *Proceedings of the 1st Workshop on Green Computing (GCM'10)*, 2010.

[18] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, CCGRID '10, pp. 826–831, 2010.

[19] G. Dhiman, G. Marchetti, and T. Rosing, "vgreen: a system for energy efficient computing in virtualized environments," in *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, ISLPED '09, pp. 243–248, 2009.

[20] A. Younge, G. von Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers, "Efficient resource management for cloud computing environments," in *Green Computing Conference, 2010 International*, pp. 357–364, 2010.

[21] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, "Server workload analysis for power minimization using consolidation," in *Proceedings of the 2009 conference on USENIX Annual technical conference*, USENIX'09, (Berkeley, CA, USA), pp. 28–28, USENIX Association, 2009.

[22] E. Triantaphyllou, *Multi-Criteria Decision Making Methods: A comparative Study*. Springer, 2004.

[23] S. J. Russell, P. Norvig, and E. Davis, *Artificial intelligence: a modern approach*, vol. 2. Prentice hall Englewood Cliffs, 2010.

[24] 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.

[25] R. E. Neapolitan, *Learning bayesian networks*. Pearson Prentice Hall Upper Saddle River, 2004.