Plug4Green: A Flexible Energy-aware VM Manager to Fit Data Centre Particularities

Corentin Dupont[a,*], Fabien Hermenier[b], Thomas Schulze[c], Robert Basmadjian[d], Andrey Somov[a], Giovanni Giuliani[e],

[a] *CREATE-NET, Trento, Italy*
[b] *University of Nice Sophia Antipolis, CNRS, I3S, UMR 7271*
[c] *University of Mannheim, Mannheim, Germany*
[d] *ONE LOGIC GmbH, Passau, Germany*
[e] *HP Italy Innovation Centre, Milan, Italy*

Abstract

To maintain an energy footprint as low as possible, data centres manage their VMs according to conventional and established rules. Each data centre is however made unique due to its hardware and workload specificities. This prevents the ad-hoc design of current VM managers from taking these particularities into account to provide additional energy savings. In this paper, we present Plug4Green, an energy-aware VM placement algorithm that can be easily specialized and extended to fit the specificities of the data centres. Plug4Green computes the placement of the VMs and state of the servers depending on a large number of constraints, extracted automatically from SLAs. The flexibility of Plug4Green is achieved by allowing the constraints to be formulated independently from each other but also from the power models. This flexibility is validated through the implementation of 23 SLA constraints and 2 objectives aiming at reducing either the power consumption or the greenhouse gas emissions. On a heterogeneous test bed, Plug4Green specialization to fit the hardware and the workload specificities allowed to reduce the energy consumption and the gas emission by up to 33% and 34%, respectively. Finally, simulations showed that Plug4Green is capable of computing an improved placement for 7,500 VMs running on 1,500 servers within a minute.

Keywords: Extensibility, Constraint Programming, Cloud Computing, Data Centre, Resource Management, Energy Efficiency, Service Level Agreement.

1. Introduction

Cloud data centres provide powerful ICT facilities to host a large spectrum of applications. Originally, data centre operation management has been focused

*Corresponding author: cdupont@create-net.org, tel: +39 0461 408400 ext:403

on improving metrics like performance, reliability, and service availability. Furthermore, due to the rise of service demands, data centres have to constantly evolve in size and complexity. This and the continuous increase of energy cost have prompted the ICT community to add energy efficiency as a new key metric for improving data centres facilities.

VM consolidation is the norm to improve energy efficiency. In practice, an *ad-hoc* VM placement algorithm considers the data centre and the workload properties to allocate the VMs among the servers according to energy objectives [1, 2, 3, 4, 5]. However, the hardware refreshing, the workload characteristics but also the wide variety of SLAs make each data centre unique. As a result, the original algorithm may not be appropriate anymore, while its *ad-hoc* nature may prevent it from being upgraded according to the new data centre properties.

These evolutions are calling for a VM allocation approach that is flexible enough to address data centres complex and changing nature. In other words, a VM placement algorithm has to take into account a large spectrum of tuning possibilities and constraints associated with data centre specificities. As an example, an energy-aware algorithm should be able to provide additional energy savings by being fine-tuned according to multiple particular hardware and SLAs. We define the following requirements on flexibility that have to be met by a VM placement algorithm: i) be extensible with users and operators new constraints and requirements, especially in the case of new SLA definitions associated with new services. ii) be adaptable to any data centre and its particularities, and be adaptable to new hardware installed.

To satisfy the requirements, we propose to use the Constraint Programming (CP) [6] paradigm. This paradigm and its associated algorithms have already been applied to address common users requirements such as performance and fault tolerance [7, 8]. However, the lack of energy models prevented it to explicitly address the energy related concerns which become of vital importance in upgraded and consolidated data centres with improved capacity.

In this paper we present Plug4Green, an energy aware VM manager based on CP, with a special focus on extensibility. We show how the flexibility realized in our framework can address new requirements arriving in a data centre. Furthermore, we show that an increased flexibility, by allowing to fine-tune the algorithms, allows better energy savings. We validate our approach by implementing a use case on energy efficient VM management in data centres while meeting the requirements on performance. The paper main contribution, in terms of its practical value, is threefold:

- *Flexibility:* We propose and implement 23 VM placement constraints to address common concerns such as hardware compatibilities, performance, security issue, and workload instability. We also propose 2 objectives: the first one reduces the overall energy consumption while the second one reduces the greenhouse gas emission. The usage of CP makes placement constraints, objectives, and algorithms independent from each other, which is crucial for extensibility: new concerns can be added in the VM manager without changing the existing implementation.

- *Efficiency:* We show that using our framework in a realistic cloud data centre environment allows to reduce the overall energy consumption up to 33% and the gas emission up to 34%. These savings are achieved by considering the servers hardware heterogeneity, their different energy-efficiency and different compositions of SLAs.
- *Scalability:* We show by simulation how such an approach can be scalable. In particular, we were able to compute the improved placement of 7,500 VMs on 1,500 servers, while respecting their SLA.

This paper is organized as follows: Section 2 presents the design of Plug4Green. Its implementation is discussed in Section 3. Section 4 evaluates its practical benefits. Section 5 introduces related works and Section 6 concludes the paper.

2. Design

Plug4Green is extensible. The architecture (Section 2.1) allows to extend the engine by adding new concerns, without modifying the underlying algorithms. In particular, new constraints (Section 2.2) can be added easily.

2.1. Architecture

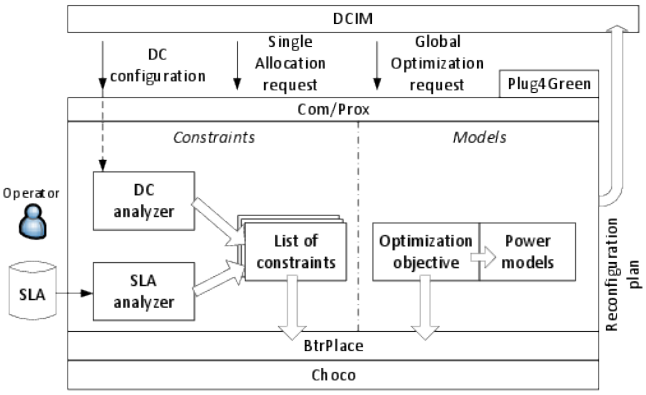


Figure 1: Plug4Green architecture

Figure 1 depicts the architecture of Plug4Green. Plug4Green considers a set of *SLA* constraints along with the *data centre configuration* to compute a *reconfiguration plan* as an output. The *data centre configuration* captures all the relevant ICT resources of a data centre with their energy-related attributes and interconnections in an XML format. The reconfiguration plan is a set of actions (powering on or off a server, migrating a VM, ...) that satisfies all the constraints and minimizes the current objective. The objective can be to minimize either the power consumption of a federation of data centres, or the CO_2 emissions. The diagram shows the clear separation between the *Constraints*

part ("what" we want to do) and the *Models* part ("how" to solve the problem), which is fundamental for extensibility.

Plug4Green is called by the Data Centre Infrastructure Management (DCIM) for two different events: *Single Allocation* or *Global Optimisation*. The *Single Allocation* event is triggered when a new VM have to be allocated. Plug4Green will compute and return the best server to allocate the VM on, taking into account the characteristics of the VM, the current state of the data centre, the SLAs and the current objective. The *Global Optimisation* event is itself triggered regularly and Plug4Green will return a reconfiguration plan. In manual mode, the data centre operator validate or decline this reconfiguration plan, while in automatic mode, it is enacted automatically. Plug4Green will then execute the plan to reduce the overall data centre power consumption or gas emission while also respecting the SLAs. The *Com/Prox* layer ensures that Plug4Green can be plugged to different existing DCIM. Its the only part that must be updated to connect to a new DCIM. Currently, Plug4Green can be integrated into VMWare[1], Eucalyptus[2], and HP Matrix Operating Environment[3] infrastructures.

2.2. Constraints

Numerous SLAs exists in a data centre. Furthermore those are more and more extended with energy concerns. Our framework provides a language to express SLAs based on CP, that also takes into account energy constraints. To show the flexibility of our approach, we prepared an extensive number of SLA and energy constraints using this language, as showed in Table 1.

SLAs are usually provided as part of an English-written contract between a client and an IT service provider. Upon receiving this contract, the Capacity Planning Team (CPT) of a data centre have to translate it into our SLA schema. The SLA schema is a format allowing the CPT to use the pre-defined constraints detailed in Table 1. Once the SLA file is ready, it can be submitted to Plug4Green. The SLA constraints will then be translated automatically to lower level CP constraints and processed by a CP engine, with the process shown in Figure 2.

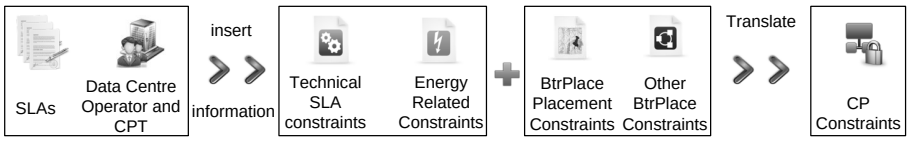


Figure 2: Translation of the SLA contract into technical SLAs and then to Constraints

Depending on the topology of the data centre, a different SLA contract can be applied to different groups of servers in the data centre: in this way it is possible to have several SLA contracts active within the same data centre.

[1] `http://www.vmware.com`
[2] `http://eucalyptus.com`
[3] `http://h18004.www1.hp.com/products/solutions/insightdynamics/overview.html`

Cat	Constraint	Restriction
Hardware	HDDCapacity	minimum amount of hard disk space available for a VM
	CPUCores	minimum number of CPU cores available for a VM
	CPUFreq	minimum CPU frequency available for a VM
	MemorySpace	minimum amount of memory space available for a VM
	GPUCores	minimum number of GPU cores available for a VM
	GPUFrequency	minimum GPU frequency available for a VM
	RAIDLevel	minimum Raid level available for a VM
QoS	MaxVMperServer	maximum number of VMs per server
	MaxCPULoad	maximum load of CPUs for a server
	MaxVLoadperCore	maximum virtual load associated to a CPU core
	MaxVCPUperCore	maximum number of virtual CPU associated to a CPU core
	MaxVRAMperPhyRAM	maximum amount of virtual RAM per physical RAM
	MaxServerAvgVCPUperCore	Same as MaxVCPUperCore but averaged for all cores of a server (not Core per Core)
	MaxServerAvgVRAMperPRAM	Same as MaxVRAMperPRAM but on a server basis
	Bandwidth	minimum network bandwidth available for a VM
Security	DedicatedServer	a VM will be hosted on a server with no other VMs
	Access	a certain secure access possibility for a VM (e.g. VPN)
Energy	MaxServerPower	maximum power consumption for a server
	DelayBetweenVMMigrations	minimum delay between two successive VM migrations
	DelayBetweenServerOnOffs	minimum delay between two state changes for a server
	VMPaybackTime	allow a VM migration only if the energy spent for the migration is paid back within the given time interval.
	SpareNodes	minimum amount of servers that are kept free (spare capacity) in the data centre
	SpareCPUs	minimum amount of CPUs that are kept free in the data centre

Table 1: SLA Constraints

3. Implementation

In this section we provide details on the model that allowed us to easily build the constraints presented earlier. We then present the power objective model and the heuristics we used to increase the scalability of our framework and the quality of the computed configurations.

3.1. The Plug4Green model

Plug4Green extends the flexible consolidation manager BtrPlace [8]. The flexibility of BtrPlace (and consequently of Plug4Green) comes from the usage of CP [6]. CP allows modelling and solving combinatorial problems where the problem is modelled by stating constraints (logical relations) that must be satisfied by its solution. To use CP, a problem is modelled as a Constraint Satisfaction Problem (CSP), comprising a set of variables, a set of domains representing the set of possible values for each variable and a set of constraints that represent the required relations between the values of the variables. A solver computes a solution for a CSP by assigning each variable to a value that simultaneously satisfies all the constraints. A CSP can be augmented to a Constraint Optimisation Problem (COP) by stating an objective that requires to minimize or maximize the value of a given variable. The algorithm used to solve a CSP or a COP is independent of the constraints composing the problem and the order in which they are provided. When no timeout is specified, the CP solver computes and returns the solution that lead to the best solutions according to the objective and the constraints. Otherwise, it returns the best solution computed so far that still satisfies the constraints. By using CP, we achieve the important goal of separating two concerns: the development of a placement objective and the development of constraints that specialise the objective.

In practice, BtrPlace embeds the CP solver Choco[4] and provides a core CSP, called the VM Repacking Scheduling Problem (VRSP) that is dedicated to the placement of VMs to the servers. The VRSP is an abstract placement algorithm modelling the current memory and CPU demands of the VMs, the server's state and the future placements of the VMs. By default, BtrPlace *does not* provide variables, constraints and objectives that are related to energy concerns. These variables are then provided by Plug4Green, as an extension of the VRSP. We use the VRSP as a pivot model to solve energy-efficient VM placement and server management problems.

Table 2 is summarizing the energy variables that we created, and the variables that were reused from BtrPlace. As an example, Listing 1 presents the definition of the constraint *MaxServerPower* is term of these variables:

[4]Choco: http://www.emn.fr/z-info/choco-solver/

Energy related variables	
P	Future global power consumption of the data centre federation
$P(s)$	Future power consumption of a server s
E_{reconf}	Energy spent by the reconfiguration plan
$E_{move}(v)$	Energy spent for the migration of VM v
$E_{onoff}(s)$	Energy spent for switching on or off a server s
Variables used from BtrPlace model	
hosters	Association array VM/Server of the resulting configuration
$card(s)$	Number of VMs that a server s will host
$n^{CPU}(s)$	Future CPU load of a server s
$n^{RAM}(s)$	Future RAM usage of a server s
$n^{HDD}(s)$	Future HDD usage of a server s

Table 2: Model variables

```
1 public void injectMaxServerPower(VRSP model, int maxServerPower) {
2     model.post(eq(P, plus(mult(card, β), α)));
3     model.post(leq(P, maxServerPower));
4 }
```

Listing 1: Definition of constraint *MaxServerPower*

As a reminder, $maxServerPower$ ensures a maximum power consumption for a server. Using the CP operators *eq*, *plus* and *mult* we first define the power of a server P (one of the variables in Table 2). α and β are defined in the Section 3.3. We then create the constraint that this power P must be less or equal than a threshold $maxServerPower$. This constraint is then injected into the *model*. The constraint definition is declarative. It does not state *how* we want to solve it but only *what* we desire. This is a clear separation of concerns: we were not obliged to revise the solving algorithm to define this constraint.

3.2. From SLA to constraints

As can be seen in [9], SLAs commonly contain metrics related to the hardware infrastructure the customer is guaranteed to be provided with. They define for example a certain amount of memory, hard disk space, CPU frequency or a certain RAID level. In addition to this first category, the technical SLA contains also QoS-related constraints. Within the third category we capture constraints concerning security issues, such as secure access possibilities (e.g. VPN) or the guarantee that one VM will only be hosted on one server. The fourth category include energy related constraints: constraint on the energy consumption for servers, the time between VM movements, and the amount of free resources that must be kept available in the data centre.

In order to be included in Plug4Green the constraints need to be mapped into rules understandable by the CP engine. The constraint can be either implemented over pre-existing constraints in BtrPlace or using the low-level constraints provided by the Choco library. For instance, the constraints related

to hardware metrics are usually implemented using the *Fence* constraint provided by BtrPlace to restrict the VM placement to a given group of servers. In a pre-selection process, a set of servers having a satisfying hardware is computed. A *Fence* constraint is then instantiated with this set, allowing an allocation to be performed only on this set of servers. In practice, 17 of the 23 constraints bundled currently inside Plug4Green were developed without relying on pre-existing constraints in BtrPlace.

The output of Plug4Green is also dependent on energy-related constraints. The simplest of these constraints is *MaxServerPower*: it allows the data centre operator to specify the maximum power that a server or a group of servers can consume. This constraint takes into account the fact that Plug4Green does not work with perfect information. Indeed, despite Plug4Green aims explicitly at reducing the energy of the overall federation of data centres, it may not be aware, for example, that the cooling system of a specific group of servers is not efficient, or that its power feed is weak. This information may not be included in the power model of Plug4Green, showing the usefulness of the constraint. In practice, to satisfy this constraint Plug4Green will limit the number of VMs hosted by the servers, to keep the overall power under the threshold.

The energetic and performance costs of a VM movement itself are also considered. Indeed, when a VM is migrated in live, some energy is spent. Plug4Green provides the data centre operator three ways of acknowledging this fact. The first and preselected possibility is to take the VM life-time into account, if available. If we know in advance the remaining time that a VM will be online before being shut down, its easy to compute whereas its worthwhile to migrate it. We simply compare the energy saved by the VM if we move it to the energy cost of the move, as shown in Section 3.3 However, especially in Cloud computing, the remaining life-time of VM may not be available. The *DelayBetweenMove* provides a second opportunity to control the migrations by stating a minimal delay between migrations of given VMs. This simple solution can be used whenever no information about the remaining life-time is available. Finally, a more advanced version considers the migration as an investment that must be worthy. This management opportunity is granted through the *VMPaybackTime*. For this constraint, the data centre operator defines a mean life time for VMs, depending on the data centre typology. This time will be used to compute if its worthwhile to move a VM.

In addition to the acknowledgement of energy consumed for the movement of VMs, a data centre operator needs to deal with the problem of rapid fluctuations of workload. One solution is to ensure a specific amount of resources is always available to absorb the variations. We implemented the constraints *SpareNodes* and *SpareCPUs* to allow the data centre operator to define the associated values as a function of time. In this way the best trade-off between reliability and energy efficiency is achieved. If, for instance, the historical load pattern of the data centre shows that during night time there is only a low variation of the number of VMs but in the time between eight and nine rapid rise of number of VMs occurs (e.g. due to the start of the working day), the *SpareNodes* or *SpareCPUs* parameter values are kept small during night-time and is increased

before office hours.

3.3. Optimisation Objectives

In this section, we first introduce the power prediction models that estimate the power consumption of a server. We then present two objectives implemented as a proof of concept to optimise a data centre usage: *minEnergy* to reduce the energy consumption, and *minGasEmission* to reduce the CO_2 emission.

3.3.1. Power Consumption Prediction

In order to derive the optimisation objectives mentioned, we need to design appropriate power consumption prediction models. In this section, we provide models for idle and dynamic power consumption estimations. We introduce the corresponding models for servers by breaking down into their constituent components (e.g., processor, memory, hard disk, see Figure 3).

When a *Single Allocation* or a *Global Optimisation* is triggered, the framework collects all the necessary information from the data centre management framework, in particular the *dynamic* parameters. This information is passed to Plug4Green using the schema described in [10]. The prediction models described in this section are then used by Plug4Green to build an objective function, adapted for the current configuration and state of the data centre. The built objective function (either *minEnergy* or *minGasEmission*) is thus recomputed and may be different for each call of Plug4Green.

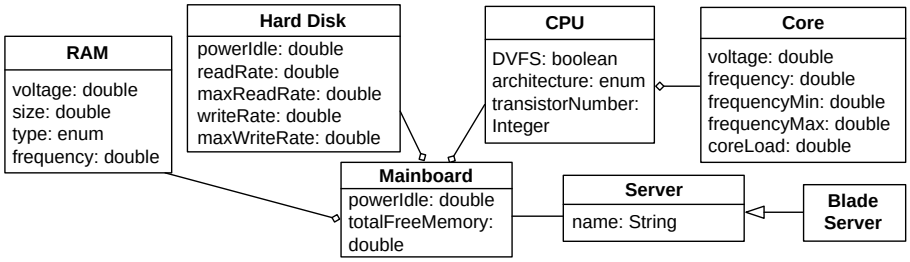


Figure 3: Server UML class diagram

Idle power. A server is in idle state when all of its constituent components are inactive but still powered. To this end, the major contributors to the idle power consumption of a blade server are the processor, memory, and hard disk. The idle power consumption of a *multi-core* processor i is given by equation (1) where t denotes the total number of cores, ν_s the total number of transistors of the s^{th} core, I_{us} and V_{us} the current and the voltage of the u^{th} transistor of the s^{th} core, respectively.

$$P_{CPU}(i) = \sum_{s=1}^{t} \sum_{u=1}^{\nu_s} I_{us} \times V_{us} \tag{1}$$

In [11], an approach was presented to model the current I_{us} in terms of the voltage V_{us}, by taking into account the processor frequency. Hence, the most relevant energy-related parameters are the architecture (e.g., Intel, AMD), the number of transistors (in the order of millions), voltage as well as the corresponding C-states[5] with which the processor is operating. Note that the last two attributes are dynamic. Concerning the *memory* modules, they consume power during the idle state while refreshing the memory ranks holding stored data. The idle power consumption of a memory module j is given by:

$$P_{RAM}(j) = \sum_{\nu=1}^{r} s_\nu \times f_\nu \times c \times V_\nu^2 \tag{2}$$

where s_ν denotes the size (GB), f_ν indicates the frequency (MHz), and V_ν represents the voltage of the v^{th} memory module, whereas c is a constant. In [11], values for c where derived based on the type (e.g., DDR2, DDR3) of the memory modules. Consequently, the most relevant energy-related attributes are the size, frequency, voltage, and type of the memory modules. All the above-mentioned attributes are static ones and can be found within the manufacturers data sheets. A *hard disk* is in idle state when neither read nor write operations take place. Moreover, during the idle state, most of the mechanical parts of the disk are stopped. Consequently, manufacturers provide in their data sheets the average idle power consumption for hard disks which can be used as a best estimate.

Given a blade server s composed of processors, memory modules and hard disks, its idle power consumption is estimated by the equation (3). l, m, and n denote respectively the total number of processors, memory modules as well as hard disk drives. c represents the power consumption of the mainboard which can be estimated from observation data[6].

$$P_{idle}(s) = \sum_{i=1}^{l} P_{CPU}(i) + \sum_{j=1}^{m} P_{RAM}(j) + \sum_{k=1}^{n} P_{HDD}(k) + c \tag{3}$$

Dynamic power. The dynamic power denotes the power that is consumed by a server to carry out operations of the running VMs such as accessing the memory or the hard disk as well as executing instructions by the processor. As in the idle part, the major contributors to the dynamic power consumption of blade servers are the processor, memory and hard disk. Equation (4) models the dynamic power consumption of a multi-core i processor based on the following well known CMOS circuit formula. t denotes the total number of cores, C_{eff} the effective capacitance (i.e. activity factor), f the frequency, and V the voltage of the core.

$$P_{CPU}^{'}(i) = \sum_{s=1}^{t} C_{eff}(s) \times f(s) \times V^2(s) \tag{4}$$

[5]Technology enabling for a processor to choose from a set of power related saving modes.
[6]For the blade servers of the evaluated testbed, c takes a value between 70-85 W.

In [12], the authors showed that the power consumption of a multi-core processor is not the pure summation of the consumption of its constituent cores. Consequently, the authors modelled the power consumption of the processor by dividing it into different levels and identified the contribution of each level to the overall power consumption. For a processor, its frequency, voltage, utilization rate but also its specific energy efficient mechanism such as Intel SpeedStep [13] play a major role in the computation of the dynamic power consumption. Note that all the above-mentioned parameters are dynamic. DDR3 memory modules have a constant power consumption (in average 9.5 W) regardless read or write operations are carried out. Since in cloud computing data centres, the availability of an application profile of VMs in terms of number of accesses to the memory for read/write is challenging, the most relevant energy-related attribute is the total available free memory space (GB) of the system. This dynamic parameter helps at defining when a memory access is probable: the more free memory space the system has, the less probable will be access to the memory. Finally, the hard disk consumes power when its mechanical as well as electrical parts are used to perform read or write operations. Furthermore, an additional start up power is induced whenever the disk changes its state from idle to accessing modes. As in the case of memory, since its not possible to have a detailed profile of the application, then the most relevant energy-related attributes for the hard disk are the read and maximum read rates as well as write and maximum write rates all expressed as MB/s. Those parameters are dynamic and help in providing guesses with respect to one of the three states (e.g., idle, start up, accessing) the hard disk could be. Additionally, the power to access the hard disk is in average 1.4 times greater than the one of the idle state. Further details on the probabilistic approaches adopted for memory and hard disk can be found in [14]. As a result, the overall power consumption of a blade server is the summation of the P_{idle} and the dynamic powers of its CPUs, RAM and HDDs.

3.3.2. Power Objectives Elaboration

Having the power consumption prediction models described in the previous section at the disposal of Plug4Green, we are now able to compute the power objectives. However, using directly the power consumption prediction models at each step of the optimisation process would be too costly in terms of computation time and resources. Furthermore, this approach would not take advantage of the CP, where the objective function must be stated as a *constraint programming variable* that must be minimized, and thus cannot be written as a simple java function. Our approach to solve this problem is the following: In a first step, Plug4Green groups the servers into families that share similar hardware characteristics (e.g., processor, memory, hard disk), and similarly the VMs are grouped into families that share similar characteristics according to the SLA (e.g., small, medium, large). Note that such an assumption is possible since it is common for a data centre to have families of similar equipment and because VMs often share similar run-time characteristics as well. Plug4Green will then generate for each server its idle and dynamic power consumption patterns under several usage conditions, using the VMs families to simulate the load, and store

them in two vectors α and β. This means that the necessary values are retrieved and stored in vectors before and not during the search process which results in a much faster search. We can obtain the pre-computed version of the power consumption for the server i in family k by using the following equation:

$$P(s_i^k) = X_i \times \alpha_k + \sum_{j=1}^{p} h_{ij} \times \beta_{kl} \tag{5}$$

where α_k denotes the idle power of the family of servers k, and β_{kl} denotes the power consumption of the VM l if running on a server from family k. $h_{ij} = 1$ if the node s_i^k is hosting the VM ν_j^l and 0 otherwise. X_i is a variable with a value of 1 if there is at least one VM in a server s_i^k, and 0 otherwise. We assume that, if a server contains on VMs, it can be switched off by Plug4Green and then consumes no energy. We denote as PUE_d the Power Usage Effectiveness of the data centre d from a federation of D data centres. The global power consumed by this federation is computed by Plug4Green as:

$$P = \sum_{d=1}^{D} (PUE_d \times \sum_{i=1}^{n} P(s_i^k)) \tag{6}$$

To reduce energy consumption, consolidation through VM migration is a common solution. This operation has however an energy cost that is integrated in the power objective function of Plug4Green. This way, Plug4Green will not migrate a VM if the cost of the move is too high compared to the expected energy gain. We compute the energy needed for the migration of a VM $E_{move}(i)$ based on the characteristics of the source server, the destination server and the VM itself, as detailed in the power consumption evaluations in [15]. We also include an energy penalty $E_{onoff}(j)$ for switching on and off a server. Indeed, a certain amount of time is needed to switch on or off a server and during this time, no workload can be carried out despite a certain energy consumption.

As an approximation, we assume that the energetic situation in the data centre is stable between two reconfigurations during a delay T_{reconf} (in seconds). At the next reconfiguration and to take into account changes in the data centre like VM termination, Plug4Green will recompute the power objective. Using equation (6), Plug4Green computes P_{bef} and P_{aft}, the power of the federation before and after application of the reconfiguration plan, respectively. The global energy saved by the reconfiguration plan, at federation level is therefore:

$$E_{tot} = (P_{bef} - P_{aft}) \times T_{reconf} - \sum_{i=1}^{p} E_{move}(i) - \sum_{j=1}^{n} E_{onoff}(j) \tag{7}$$

Similarly, Plug4Green is computing Q_{total}, which is the total quantity of carbon emissions saved by the reconfiguration plan, by replacing "PUE" by "CUE" in the equations. As stated at the beginning of this section, our objectives *minEnergy* or *minGasEmission* consists of minimizing E_{tot} or Q_{tot}, respectively.

3.4. Reducing the Solving Duration

Computing a configuration according to an objective may be time consuming for large infrastructures as selecting a satisfying server for each running VM is NP-Hard [7]. To solve a COP, the constraints (see Section 3.2) are used by the solver to remove inconsistent variable assignments, while the power objective variable is used to select values that are relevant to save energy. However, this can be a very time-consuming process. To help reduce this duration, so-called *search heuristics* are used to indicate to the solver the variables to focus on in priority, and supposed good values to try first. A *search heuristic* is thus attached to each objective (*minEnergy* or *minGasEmission*). The objective is to find good solutions as soon as possible in the search tree. A search heuristic is tightly coupled to an objective but completely independent from the stated constraints to maintain the composability of Plug4Green. This way, an arbitrary number of constraints can be used with the same search heuristics. In Plug4Green, the heuristics are typically guiding the solver into finding values for the variables related to the position of the VMs on the servers and the state of the servers.

For each objective, its search heuristic suggests to migrate the VMs from the least loaded, or least energy-efficient servers, to highly-loaded and energy-efficient servers. The notion of efficiency depends on the objective: The PUE is used for the *minEnergy* objective and the CUE is used for the *minGasEmission* objective. Once the new placement for each VM is computed, the heuristics makes the solver try to turn off unused servers.

4. Framework Evaluation

As stated before, the goal of Plug4Green is to improve data centres energy efficiency through placement algorithms that are easy to specialize. In this section, we first discuss the extensibility of Plug4Green. We then demonstrate the impact of its specializations to reduce the power consumption or the gas emission on a heterogeneous data centre federation running an industrial workload. We finally evaluate its scalability within a simulator for a data centre with up to 2500 servers.

4.1. Extensibility of Plug4Green

The design of Plug4Green allows the integration of new concerns. Each constraint and objective is a plug-in composed by a XML Schema and a Java implementation. A developer can then develop a new constraint or objective as a plug-in and integrate it to Plug4Green with an automatic and deterministic specialization process. Contrary to methods derived from Linear Programming, the developer must not provide a linear model for his constraint. This eases tremendously the expertise that is required to express constraints. In practice, it took only a few hours for an engineer to create and test a new constraint.

To demonstrate this flexibility, we developed 23 placement constraints and 2 objectives. They have been developed to match a large range of expectations from the clients and data centre operators in terms of hardware compatibilities,

performance level, resource sharing or energy capping. Plug4Green exposes a core set of variables and relations to manipulate VMs and servers. In the case those core variables would not be sufficient to express a user's problem, new variables can be added to express meaningful information, and then be linked with low-level relations to the Plug4Green variables. These low-level relations consist of either basic constraints in VM placement problem such as assignment, scheduling, or counting constraints, but also arithmetic, logic and domain constraints.

As an example, Plug4Green does not currently support the thermal-aware management of VMs. Current approaches [1, 2, 16] propose heuristics derived from thermal models to estimate the impact of the VMs management on the server temperature. To integrate a thermal model inside Plug4Green, the knowledge-specific information would be defined with new variables, linked to Plug4green variables with arithmetic constraints. Once these links established, the temperature variables would be available to express new concerns: for example, capping the server temperature to disallow hotspot, or performing a thermal load balancing to reduce the cooling costs.

4.2. Experiments on Cloud Testbed

To evaluate the practical efficiency of Plug4Green in an environment as realistic as possible, a trial has been set inside a private cloud with a state-of-the-art cloud stack running two workloads derived from industrial traces.

4.2.1. Environment Setup

The cloud simulates an heterogeneous data centre federation. It is composed of two racks (see Table 3), each embedding a HP C7000 blade enclosure. The first data centre (DC1) has 4 BL 460c to host VMs using VMWare ESX v4.0 native hypervisor. 3 additional blades are used to manage the cloud, to schedule the workloads using the open-source scheduler *JobScheduler*[7], and to run Plug4Green. The second data centre (DC2) has 3 BL 460c to host VMs also using VMWare ESX. 2 additional blades are used to manage the cloud and to monitor the system and the energy usage of the federation using *Collectd*. The racks are connected to a single LAN and a SAN device stores all the datas, including the VMs images.

	Enclosure 1	**Enclosure 2**
Processor model	Intel Xeon E5520	Intel Xeon E5540
CPU frequency	2.27GHz	2.53GHz
CPU & Cores	Dual CPU - Quad core	Dual CPU - Quad core
RAM	24 GB	24GB

Table 3: Characteristics of the Racks/Enclosures

[7] `http://sourceforge.net/projects/jobscheduler/`

Plug4Green has been evaluated against 2 synthetic workloads derived from real traces within the private cloud of a corporation in Italy[8]. Figure 4 depicts a weekly load pattern. The first workload reproduces this trace, compressed into 24 hours. The second workload focuses on a single work day (depicted by the black frame), compressed into 12 hours. The first workload considers a week-end with a low load and therefore more energy saving possibilities. The second workload is more challenging since the load is higher on average.

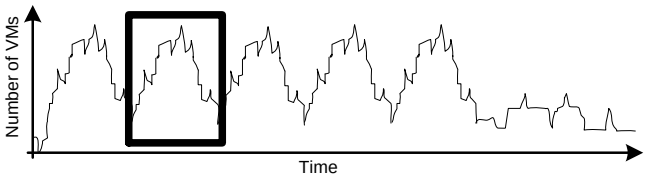


Figure 4: Schematic view on the weekly load patterns

4.2.2. Specializing Plug4Green to fit an heterogeneous federation

We evaluate here the practical efficiency of Plug4Green at managing a federation of data centres having different PUE and CUE. The experiments have been run using different data centre configurations.

In a first experiment, we evaluate the effectiveness of the *minEnergy* placement objective using 3 scenarios. In the "No P4G" scenario, Plug4Green is not used. An ad-hoc heuristic deploys the VMs on servers with a load-balancing placement objective. Idle servers are not turned off and VMs are not migrated. In the "P4G same PUE" scenario, Plug4Green is used and all the servers expose the same PUE. This is equivalent to ignoring the PUE parameter. Finally, in the "P4G different PUE" scenario, the servers in DC1 and DC2 have a PUE set to 1.5 and 2.5, respectively. Plug4Green can then benefit from the servers in DC1 that are more energy-efficient.

Figure 5 shows the result. The savings in the total federated sites energy increases to over 33% compared to the "No P4G" scenario, with an improvement of over 13% due to the consideration of the different PUE efficiency. In practice, we observed Plug4Green allocated more VMs on DC1, which was more energy-efficient overall with its lower PUE.

The second experiment evaluates the effectiveness of the *minGasEmission* placement objective using three scenarios similar to those used in the previous experiment. In the "No P4G" scenario, Plug4Green is not used. In the "P4G same CUE" scenario, Plug4Green is used and all the servers have the same CUE. Finally, in the "P4G different CUE" scenario, the servers in DC1 and DC2 have a CUE of 0.400 g/Wh and 0.250g/Wh, respectively. Figure 6 shows a reduction of the gas emissions by 34% in the "P4G same CUE" scenario with respect to "No P4G" with an increase of over 9% due to the consideration of the CUE

[8]due to privacy issue, we cannot disclose the corporation name and the workload details.

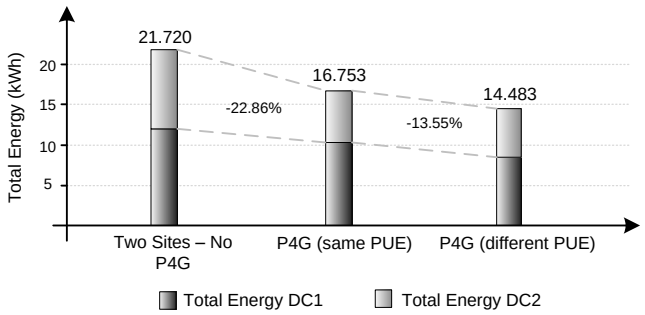


Figure 5: Energy consumption of two data centres with different PUE values

differences. Again, the behaviour of Plug4Green was to run more VMs on DC2, the most efficient data centre from the emission perspective.

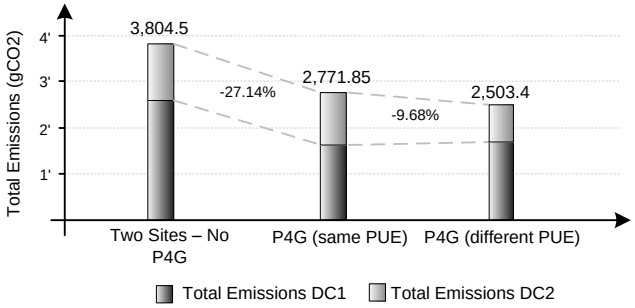


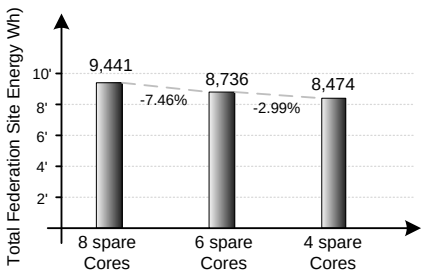
Figure 6: Energy consumption of two data centres with different CUE values

4.2.3. Specializing Plug4Green to fit the particularities of the workload

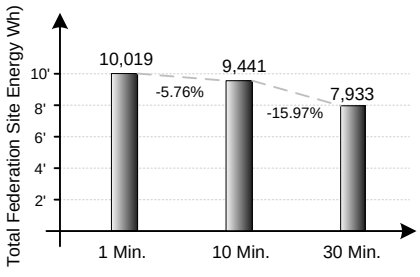
We evaluate here the practical benefits of Plug4Green when it is specialized to fit the workload. In practice, we measure the data centre energy consumption depending on different variations of a set of constraints.

The first experiment evaluates the savings when Plug4Green controls the aggressiveness of the VM consolidation. Frequent arrival of VMs may lead to a scheduling delay as additional servers may need to be booted on emergency to host them. One solution consists in ensuring at all time a certain amount of free resources (such as idle servers) in the data centre to be able to boot the VMs faster. This number should, however, be considered carefully. A too small value will be ineffective while a high value would augment the overall power consumption. With Plug4Green, this fine grain tuning is done easily through a *spareCPU* constraint. A CPU is considered "spare" when it is not associated with any VCPU. Figure 7a shows the energy savings when the number of spare cores varies from 8 to 4 cores. This confirms that the number of spares resources

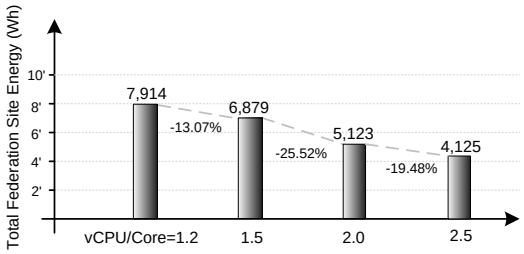
should be set to a minimum to improve energy efficiency. Maintaining at most 4 cores available instead of 8 allowed an extra 10% saving.



(a) SpareCPUs constraint impact



(b) *DelayBetweenMove* constraint impact



(c) *MaxVCPUperCore* constraint impact

Figure 7: Impact of constraints on the global energy consumption

The second experiment evaluates the savings when Plug4Green controls the frequency of the migrations. A VM migration is indeed costly in terms of energy but also in terms of performance. It is then useful to disallow to migrate too frequently the same VMs. With Fit4Green, this parameter is controlled easily through a *DelayBetweenMove* constraint. Plug4Green will then consider as candidate for migration only the VMs that have been last migrated at least the required amount of time ago. Figure 7b shows the energy saving depending on the migration time interval. With a 30-minute timeout, Plug4Green saved an extra 20% of energy compared to the 1 minute timeout, because it prevented unnecessary VM migrations.

The last experiment evaluates the savings when Plug4Green is used to control the resource sharing. Its objective is to assess to which extent we can improve the achievable energy saving, when an energy relevant SLA parameter constraint is relaxed with respects to its standard value. Based on observations from the testbed, *MaxVCPUperCore* constraint has been identified as the most important one of this category. Figure 7c demonstrates the impact of this parameter on the overall energy consumption. If we relax the constraint up to 2.5 VCPU/core, we reduce the energy consumption by up to 45%. This is not a surprise as with

a factor of 2.5 its possible to consolidate twice as much VMs on a server than with factor of 1.2. This means that Plug4Green used only one half of the servers to run the workload and can switch off the other half.

4.3. Scalability of Plug4Green

Placing VMs on servers with regard to their resource requirements is a NP-Hard problem. The scalability of Plug4Green is then determined by the size of the configurations (number of VMs and servers) and their associated constraints that Plug4Green is able to solve in a reasonable time. To evaluate this scalability, we generate 5 sets of configurations that are composed of 500 up to 2,500 servers. Each set is composed of 50 configurations, with different VM templates and different initial placements. We run Plug4Green on each configuration with the constraints evaluated in Section 4. The solving process stops once the first solution is computed. We then analyse the solving process and the estimated energy savings.

To provide a realistic evaluation with simulation data, configurations are generated from the testbed described in Section 4.2.1 and common practices. Servers are identical to those used in the cloud testbed with an equal repartition between the models used in the two enclosures. Each VM instantiates a template randomly selected among the three used in the testbed (m1.small, m1.large, m1.xlarge). The amount of VMs in each configuration equals five times the number of servers, according to a consolidation ratio observed in industry [17]. Finally, the initial VM placement is computed randomly but ensures that their SLA is initially satisfied.

Figures 8, 9 and 10 depict the results. "P4G" denotes the usage of Plug4Green without any additional constraints. The "+spare" label denotes the addition of one *SpareCPUs* constraint to keep 1% of all the PCPUs directly available. The "+vcpu" label denotes the addition of a *MaxVCPUperCore* constraint to restrict to at most 2, the number of VCPU attached to a single PCPU. Finally, the "+delay" label denotes the addition of a *DelayBetweenMove* constraint to prevent the migration of any VMs migrated less than 30 minutes ago. We consider for the simulation that 5% of the VMs, randomly selected, are in this state.

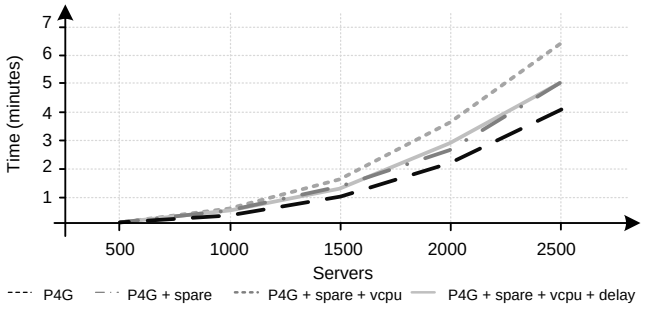


Figure 8: Solving duration to compute the improved configurations

Figure 8 shows that the solving time increases exponentially with regard to the data centre size. This is expected as the problem addressed by Plug4Green is NP-hard. Without additional constraints, 30 seconds are required to compute an improved configuration for a data centre with 1,000 servers. Doubling the size of the data centre requires 4 times more time. We however observe that Plug4Green is able to compute an improved configuration in one minute in a data centre with up to 1,500 servers running 7,500 VMs. At this scale, we observe that the addition of constraints does not alter significantly the solving process. Above that limit, the solving time gets more dependent on the constraints. The *DelayBetweenMove* constraint reduces the computation time as it reduces the number of VMs that have to be considered by Plug4Green in each time slot. However, the *SpareCPUs* and the *MaxVCPUperCore* constraints increase the computation time by 25% each. With a 1,500 servers, this adds a 15 seconds overhead. With 2,500 servers, the overhead equals one minute. This overhead is explained by the constraints implementation: each of these constraints extends BtrPlace to expose the mapping of the VCPUs to the PCPUs. This extension injects one *bin packing* [18] constraint into the CP solver that cannot scale linearly with the data centre size. By default extensions of Plug4Green, even identical, are not shareable. This includes a redundancy that alters the performance. Providing a sharing mechanism for the extensions would lower this overhead.

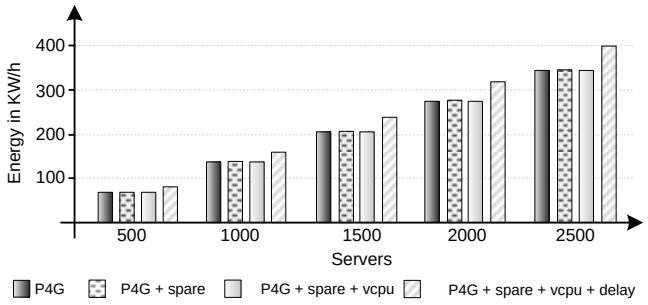


Figure 9: Energy consumption of the improved configurations

Figure 9 shows the energy consumption of the improved configurations. This value was computed using the power consumption prediction model. We first observe that the *SpareCPUs* and the *MaxVCPUperCore* constraints do not alter the quality of the improved configurations. For the *SpareCPUs* constraint, Plug4Green was able to keep free the requested amount of PCPU capacity without having to turn on additional servers. We also observe that the *DelayBetweenMove* constraint is reducing the saving opportunities by 10%. This is explained by the particular setting of this experiment: the VMs that are not allowed to be migrated due to the constraint have been selected randomly. The selected VMs are spread over numerous servers which prevented Plug4Green to turn them off.

Figure 10 shows the number of migrations to execute to reach the improved configurations. We observe the *SpareCPUs* and the *MaxVCPUperCore* con-

straints did not alter the number of migrations. This shows Plug4Green was able to consider these constraints without having to re-arrange additional VMs. The *DelayBetweenMove* constraint reduces the amount of migrations by 5% which is the number of VMs considered by the constraint.

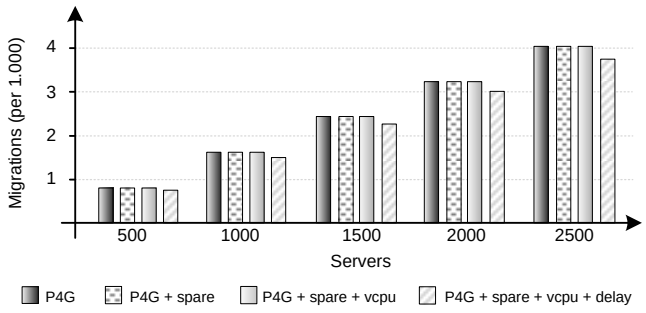


Figure 10: Number of migrations to reach the improved configurations

5. Related Work

This section discusses recent advances in the area of energy-aware frameworks for data centres. The literature relevant to our work can be divided into three groups: (i) heuristic-based approaches, (ii) the existing flexible and extensible frameworks, and (iii) the power consumption prediction models.

5.1. Heuristic Based Approaches

The problem of consolidating and rearranging the allocation of VMs in a data centre in an energy efficient manner is described in [19]. n [19], the authors propose heuristics to compute for each VM to be moved, the appropriate server that leads to minimizing the overall data centre power consumption. This is similar to the First Fit Decreasing algorithm which has been used in previous works [20, 21, 3], with the addition of power-awareness for choosing the server. In [5], the authors proposed the Modified Best Fit Decreasing, which will allocate a new VM to an active physical machine that would take the minimum increase of power consumption. [7] also proposes algorithms for VM reconfiguration and (re)allocation. The main advantage of heuristic based methods is that they are fast and easy to configure. However, in many situations they cannot lead to the optimal solution if the data centre is heterogeneous. Furthermore they will be hard to extend if new uses cases appear in the data centre. We propose a larger framework that can cope with an arbitrary number of constraints user-defined which ensure the flexibility of the framework and its extensibility regarding new constraints that may come in the future.

5.2. Extensible and flexible frameworks

A few flexible and extensible frameworks for VM allocation have been proposed recently. For example, BtrPlace [8] is a CP-based flexible consolidation manager. As already detailed in Section 3.1, Plug4Green leverages on Btrplace [7, 22]. BtrPlace does not take into consideration energy related problems and does not provide an operator with the opportunity of setting optimization objectives. In contrast to BtrPlace, Plug4Green directly addresses energy consumption problem. In this work, Plug4Green proved the practical benefits of flexibility to address energy related problems. This required numerous extensions: the development of a power model and different model extensions, two objectives with their associated heuristics, 7 energy-related constraints, and a domain-specific language to directly exhibit energy concerns and metrics such as PUE, CUE and Watts, to the end-users.

Similar modular consolidation manager adopting CP paradigm is presented in [23]. The authors ensure high availability for VM placement by guaranteeing at any time a certain number of vacant servers to allocate VMs with regards to placement constraints. The authors ensure high availability for VM placement by guaranteeing at any time a certain number of vacant servers to allocate VMs with regards to placement constraints. The scalability is demonstrated with 32 servers and 128 VMs only.

In [24], the authors propose an hybrid approach based on a Business Rules Management System (BRMS) and CP to manage VMs. The BRMS monitors and analyses the servers state at a period of time to detect overloaded servers and bottlenecks. Once a problem is identified the BRMS models its instance and sends it to the CP solver. A user can express constraints through the BRMS but the resulting specialization cannot be deterministic contrary to Plug4Green. In contrast to our manager, both the systems presented in [23] and [24] are not addressing energy-efficiency problems.

Some preliminary theoretical and practical aspects of Plug4Green were investigated in [25]. Energy-aware VM allocation was the primary goal while this work focuses on flexibility. For this purpose, we created seven new SLA constraints, notably energy-oriented, and a new power objective model has been included. Three new heuristics has been developed, allowing finding good solutions quickly. A complete experimentation has been carried out with new prototype, evaluating the impact of several popular SLA constraints on the energy saving. In this work, we demonstrate an energy saving of 33% while it was 18% in federated cloud data centre experiment in [25], due to new energy-aware constraints and heuristics. The scalability of the framework has been also greatly improved. Plug4Green is about 30 to 40 times faster which makes it capable of managing larger data-centres.

Nefeli [26] is a cloud gateway that places VMs with regard to user preferences called "hints". Nefeli expects that the users are aware of the role each VM plays in the infrastructure and communicate this information to the cloud as a hint. The VM placement is computed using simulated annealing. A hint is then implemented as a scoring function that evaluates the quality of the placement

with respect to its concern. This approach makes Nefeli flexible: Nefely can be extended by programming new hints. As a difference with Plug4Green, the approach does not separate the model from its resolution method. The specialization made by the hints is also not composable as each score is, by nature, relative to the others. Despite the authors discuss some energy-related hints, their system as a whole does not make a special emphasize on energy efficiency. Finally, Nefely has not been evaluated in terms of scalability.

5.3. Server Power Models

In [27], the authors propose a model to predict the average power consumption of a server regardless of its utilisation. The two main benefits are the followings: (i) it is simple to compute and no dynamic information is required, and (ii) it is similar to the method of estimating a system's power consumption based on the manufacturer's specifications. However, it provides very rough predictions especially for heterogeneous software and hardware environments. In [4], a linear model estimates the power consumption according to the server's CPU utilisation. This approach is not suitable for not CPU-intensive workloads. The model in [28] follows a similar approach while taking into account the utilisation of the hard disk. Authors in [29] extend the CPU and disk utilisation model by looking at performance counters of the system such as the amount of instruction-level parallelism, the activity of the cache, or the utilisation of the floating-point unit. However, performance counters are accessed differently on each processor type. As a matter of fact, this model is not usable across heterogeneous systems. In contrast to the above-mentioned models, which provide one linear model for the whole server, our approach aggregates different models for different components based on their behaviour. In addition, our approach does not need any calibration phase. Consequently, our models are suitable not only for homogeneous, but also for heterogeneous environments like cloud data centres.

6. Conclusion

Trends in application design, workload volatility, but also hardware heterogeneity make each data centre unique. However, the ad-hoc design of current energy-aware VM managers prevent them to take these particularities into account to provide additional savings. In this paper, we presented a flexible energy-aware VM manager named Plug4Green. Thanks to Constraint Programming, Plug4Green can be easily specialized to support various combinations of SLAs, power models and energy policies. Its flexibility has been verified through the implementation of 23 meaningful SLAs and 2 energy policies. Its practical effectiveness has been evaluated on an industrial testbed. While the default version of Plug4Green reduced the power consumption and the gas emission by 27% and 23% respectively, its specialization to fit the hardware heterogeneity improved the saving by up to 34%. Furthermore, additional specializations to fit the workload particularities reduced the power consumption by 9% to 50%. Finally, scalability experiments on simulated data have shown that Plug4Green

is able to compute an improved placement for 7,500 VMs running on 1,500 servers in a minute, while respecting their SLA.

In future works, we will focus on data centres powered by renewable energies. This requires indeed a new look on the energy efficient management of VMs as the nature and the availability of the energy is varying over time.

Acknowledgments

This work has been carried out within the European Projects FIT4Green (FP7-ICT-2009-4) and DC4Cities (FP7-ICT-2013.6.2). Experiments presented in this paper were carried out using the Grid'5000 experimental testbed[9], being developed by INRIA with support from CNRS, RENATER and several universities as well as other funding bodies.

Availability

Plug4Green is licensed under the terms of the Apache 2.0 License. The prototype is available for download at `https://github.com/fit4green/Plug4Green`.

[1] R. K. Sharma, C. E. Bash, C. D. Patel, R. J. Friedrich, J. S. Chase, Balance of power: Dynamic thermal management for internet data centers, IEEE Internet Computing 9 (1).

[2] J. Moore, J. Chase, P. Ranganathan, R. Sharma, Making scheduling "cool": temperature-aware workload placement in data centers, in: Proceedings of the annual conference on USENIX Annual Technical Conference, ATEC '05, USENIX Association, Berkeley, CA, USA, 2005, pp. 5–5.

[3] A. Verma, P. Ahuja, A. Neogi, pmapper: Power and migration cost aware application placement in virtualized systems, in: Middleware 2008, Vol. 5346 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008.

[4] X. Fan, W.-D. Weber, L. A. Barroso, Power provisioning for a warehouse-sized computer, in: Proceedings of the 34th annual international symposium on Computer architecture, ISCA '07, ACM, New York, NY, USA, 2007.

[5] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Generation Computer Systems 28 (5) (2012) 755–768.

[6] F. Rossi, P. v. Beek, T. Walsh, Handbook of Constraint Programming, Elsevier Science Inc., New York, NY, USA, 2006.

[9] `https://www.grid5000.fr`

[7] F. Hermenier, S. Demassey, X. Lorca, Bin repacking scheduling in virtualized datacenters, in: Proceedings of the 17th international conference on Principles and practice of constraint programming, CP'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 27–41.

[8] F. Hermenier, J. Lawall, G. Muller, Btrplace: A flexible consolidation manager for highly available applications, IEEE Transactions on Dependable and Secure Computing 10 (5).

[9] A. Paschke, E. Schnappinger-Gerull, A categorization scheme for sla metrics, in: Multi-Conference Information Systems, MKWI'06, 2006.

[10] R. Basmadjian, H. de Meer, R. Lent, G. Giuliani, Cloud computing and its interest in saving energy: the use case of a private cloud, Journal of Cloud Computing 1 (1).

[11] R. Basmadjian, F. Niedermeier, H. De Meer, Modelling and analysing the power consumption of idle servers, in: Sustainable Internet and ICT for Sustainability (SustainIT), 2012, 2012, pp. 1–9.

[12] R. Basmadjian, H. de Meer, Evaluating and modeling power consumption of multi-core processors, in: Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet, e-Energy '12, ACM, New York, NY, USA, 2012.

[13] V. Pallipadi, Enhanced intel speedstep technology and demand-based switching on linux, Intel Developer Service.

[14] R. Basmadjian, N. Ali, F. Niedermeier, H. de Meer, G. Giuliani, A methodology to predict the power consumption of servers in data centres, in: Proceedings of the 2nd International Conference on Energy-Efficient Computing and Networking, e-Energy '11, ACM, New York, NY, USA, 2011.

[15] W. Voorsluys, J. Broberg, S. Venugopal, R. Buyya, Cost of virtual machine live migration in clouds: A performance evaluation, in: Proceedings of the 1st International Conference on Cloud Computing, CloudCom '09, Springer-Verlag, 2009.

[16] J. Xu, J. A. B. Fortes, Multi-objective virtual machine placement in virtualized data center environments, in: Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, GREENCOM-CPSCOM '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 179–188.

[17] Virtualization penetration rate in the enterprise, Tech. rep., Veeam Software (2011).

[18] P. Shaw, A constraint for bin packing, in: M. Wallace (Ed.), Principles and Practice of Constraint Programming CP 2004, Vol. 3258 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 648–662.

[19] D. M. Quan, R. Basmadjian, H. de Meer, R. Lent, T. Mahmoodi, D. Sannelli, F. Mezza, L. Telesca, C. Dupont, Energy efficient resource allocation strategy for cloud data centres, in: E. Gelenbe, R. Lent, G. Sakellari (Eds.), Computer and Information Sciences II, Springer London, 2012, pp. 133–141.

[20] N. Bobroff, A. Kochut, K. Beaty, Dynamic placement of virtual machines for managing sla violations, in: Integrated Network Management, 2007. IM '07. 10th IFIP/IEEE International Symposium on, 2007, pp. 119–128.

[21] T. Wood, P. Shenoy, A. Venkataramani, M. Yousif, Black-box and gray-box strategies for virtual machine migration, in: Proceedings of the 4th USENIX conference on Networked Systems Design & Implementation, NSDI'07, USENIX Association, Berkeley, CA, USA, 2007, pp. 229–242.

[22] F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller, J. Lawall, Entropy: a consolidation manager for clusters, in: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, VEE '09, ACM, 2009, pp. 41–50.

[23] E. Bin, O. Biran, O. Boni, E. Hadad, E. Kolodner, Y. Moatti, D. Lorenz, Guaranteeing High Availability Goals for Virtual Machine Placement, in: International Conference on Distributed Computing Systems, 2011.

[24] R. Krogt, J. Feldman, J. Little, D. Stynes, An integrated business rules and constraints approach to data centre capacity management, in: D. Cohen (Ed.), Principles and Practice of Constraint Programming, Vol. 6308 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010.

[25] C. Dupont, T. Schulze, G. Giuliani, A. Somov, F. Hermenier, An energy aware framework for virtual machine placement in cloud federated data centres, in: Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet, e-Energy '12, ACM, 2012, pp. 4:1–4:10.

[26] K. Tsakalozos, M. Roussopoulos, A. Delis, Hint-based execution of workloads in clouds with nefeli, Parallel and Distributed Systems, IEEE Transactions on 24 (7) (2013) 1331–1340.

[27] S. Rivoire, P. Ranganathan, C. Kozyrakis, A comparison of high-level full-system power models, in: Proceedings of the 2008 conference on Power aware computing and systems, HotPower'08, USENIX Association, 2008.

[28] T. Heath, B. Diniz, E. V. Carrera, W. Meira, Jr., R. Bianchini, Energy conservation in heterogeneous server clusters, in: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming, PPoPP '05, ACM, New York, NY, USA, 2005, pp. 186–195.

[29] D. Economou, S. Rivoire, C. Kozyrakis, Full-system power analysis and modeling for server environments, in: In Workshop on Modeling Benchmarking and Simulation, 2006.