Energy Aware Infrastructure for Green Cloud Data Centres

Corentin Dupont
University of Trento
Via Sommarive, 5
38123 Trento, Italy
cdupont@create-net.org

ABSTRACT

Data centres are powerful facilities which aim at hosting ICT services. They have huge needs in term of power supply. The current trend is to, on the one hand, try to reduce the overall consumption of a data centre, and on the other hand to prioritize the utilization of renewable energies over brown energies. Renewable energies tend to be very variable in time (e.g. solar energy), and thus renewable energy aware algorithms tries to schedule the applications running in the data centres accordingly. However, one of the main problems is that most of the time very little information is known about the applications running in data centres. More specifically, we need to have more information about the current and planned workload of an application, and the tolerance of that application to have its workload rescheduled. In this paper, we present Plug4Green, a flexible VM manager able to reduce energy consumption in data centres. We propose a research plan to extend Plug4Green in order to also foster the usage of renewable energies.

General Terms

Cloud Computing, Data Centre, Resource Management, Energy Efficiency, Renewable Energies, Service Level Agreement, Extensibility, Constraint Programming

1. INTRODUCTION

Data centres are large facilities which purpose is to host information processing and telecommunication services for scientific and/or business applications. Until recently, research on data centres has been focused only on improving metrics like performance, reliability, and availability. However, due to the rise in service demands together with energy costs, the energy efficiency has now been added as a new key metric for data centres. Indeed, the prices for electricity are constantly getting higher and carbon emissions to the environment are increasing every year. Energy-aware strategies are beginning to be integrated inside the data centre resource manager. In practice, a Virtual Machine (VM) placement algorithm considers the data centre and the workload characteristics to place the VMs among the servers in the most efficient way, considering performance and energy consumption. This placement must be done respecting the requirements of the Service Level Agreement (SLA) existing between the data centre and its clients.

In parallel to reducing the overall energy consumption, the current trend is to foster the use of renewable energies. Renewable energies have the problem to be very variable and time-dependent: for example solar power is available only during the day, and is subject to variations due to the meteorological conditions. Thus, data centre operators must try to shift the workload of running applications in time, to match it with the availability (or forecast availability) of renewable energy.

In this paper we present a research plan to make data centres more energy aware. The goal is two-fold:

- Reduce the overall energy consumption of a data centre.
- Allow the data centre to better use the renewable energies.

As preliminary work, we present Plug4Green, an energy aware VM manager based on Constraint Programming (CP) [1]. The use of CP allows to attain a great flexibility and extensibility: indeed data centres are evolving permanently and new use cases are added regularly. We demonstrate how the flexibility realized in our framework can address new requirements arriving in a data centre. We propose and implement 23 VM placement constraints to address common concerns such as hardware compatibilities, performance, security issues, and workload instability. We also propose 2 objectives: the first one allows to reduce the overall energy consumption while the second one allows to reduce the greenhouse gas emission. The usage of CP makes placement constraints, objectives, and algorithms independent from each other: new concerns can be added in the VM manager without changing the existing implementation. We evaluate our approach in an industrial test bed, to show that it is both efficient and scalable:

- Using our framework in a realistic cloud data centre environment allowed 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.
- 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.

In the remainder of this paper, we will first perform a survey of the related works in Section 2 and define the research goal and plan in Section 3. We will then present our preliminary

results in Section 4 and finally conclude in Section 5.

2. RELATED WORK

The problem of consolidating and rearranging the allocation of VMs in a data centre in an energy efficient manner is described in [2]. In the heuristics proposed in [2], the algorithm computes, for each VM to be moved, the appropriate server that leads to minimizing the current overall power consumption of a data centre. This is similar to the First Fit Decreasing algorithm which has been used in previous works [3, 4, 5], with the addition of power-awareness for choosing the server. In [6], 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.

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 will be detailed in Section 4.2.1, Plug4Green leverages on Btrplace [7, 9]. 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 [1] and Watts, to the end-users.

Similar modular consolidation manager adopting CP paradigm is presented in [10]. 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 manager scalability is effective for 32 servers and 128 VMs. 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 manager scalability is effective for 32 servers and 128 VMs.

A hybrid system proposed in [11] solves a resource reallocation problem. This system includes Business Rules Management System (BRMS) and CP. A user can customize both business rules and constraints. 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 which resolves it within seconds. In contrast to our manager, both the systems presented in [10] and [11] are not addressing energy-efficiency problems.

Nefeli [12] 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. As opposed to Plug4Green, Nefely has not been evaluated in terms of scalability.

Some preliminary theoretical and practical aspects of Plug4Green were investigated in [13]. Compared with this work, 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 the 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 [13], 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.

3. RESEARCH GOAL AND PLAN

In this section we introduce the problem that this research proposal will try to solve. We then present our plan to attain this goal, including development, trial and dissemination activities.

3.1 Research Problem

As already mentioned, this research proposal aims at finding solutions to reduce the overall energy consumption of a data centre and to increase the usage of the renewable energies. A great challenge of efficiently using the renewable energies in a data centre is to be able to schedule correctly the workload of the applications. Indeed, the availability of renewable energy can have a great variation in time, with comparison to brown energies. To increase the use of renewable energies with respect to brown energies, it is necessary to shift in time the workload of some applications in the data centre. This shows the importance of being able to know the workload an application will have to run at a certain point of time, to understand under what conditions it can be shifted or delayed, and *in fine* to schedule it correctly.

Yet, currently most of the applications running in data centres are unaware of their self workload: they are unable to predict how much computing power they will require and when. In data centres, the knowledge of the require-

[1]PUE and CUE are defined by The Green Grid Consortium: http://www.thegreengrid.org/

ments of an application in terms of resources is still "meta-knowledge", i.e. the knowledge of the data centre operators. It is the role of the data centre operator to provision sufficient resources for an application, and this provision is often done in a static way. For example, in data centres, database indexing maintenance operations are usually performed at night, to minimize the impact on the overall performance. However, in a data centre using primarily solar power, it would be interesting to shift this task during the lunch break, when the sun is shining. The knowledge that this particular task, "database indexing", can cope with a 12 hour shift, and that it takes approximately half an hour, belongs to the operator's knowledge. It is a very coarse grained and subjective knowledge. This advocates the need for:

- a standardized format and protocol for applications to advertise in real-time their own needs in term of resources, including possible performance trade-offs and uncertainty ranges (this format is part of the so-called "application profile"),
- a library and programming methodology to ease the extraction of application profiles from the application source code itself, at run-time or compile-time,
- a data centre management framework and algorithms able to read the application profiles and use them to consolidate and schedule the applications on the servers in the appropriate way, in order to minimize a given utility function,
- a library and programming methodology to allow an external process to control the application load to some extent.

The problem of deriving an application profile is very accurate in many load optimization and prediction problems. More generally, automatic derivation of program macroscopic properties is a topic that has a great number of applications in the ICT field, especially with the emergence of Cloud Computing. For example, a smart phone application might want to "off-load" part of its workload to the Cloud, or on the contrary, a service running in the Cloud might be relocated in the local device for performance reasons. These application or service migrations must be controlled by a decision framework, which must know the exact applications profile in order to take the right decisions. We need to research and develop the algorithms, methodologies and tools to make applications "self aware" to a certain extent. In practice very seldom applications are able to know "what" workload they have to perform and when it needs to be done. They just "do it", in some sense. There is no internal representation of this workload, or when there is one (like in Hadoop or some database management framework), it is not general enough and standardized.

3.2 Development

The development plan includes developing the application profile format and libraries as detailed in the previous section. Furthermore, we will extend the existing Plug4Green prototype to include:

- an extended model to represent the renewable energies,
- a model to read, represent and interpret the applications profiles,
- additional constraints, scheduling oriented,
- search heuristics able to reduce the search duration.

The development plan includes early releases to be able to detect problems and correct them as soon as possible.

3.3 Trial

The developed solutions will be trialled in the staging area of the main APSS data centre. APSS is a public institution with 7500 employees, 2 main hospital and 5 minor hospitals that governs all the healthcare system of the Trentino Province. The ICT department of the APSS manages the 3 data centres owned by APSS and all the ICT infrastructures needed by the healthcare system. In particular the data centre of the APSS, where the experimentation will take place, provides services to the APSS employees (healthcare operators) and to the citizens of the Trentino Province. In the context of the Trial, the energy is provided by Dolomiti Energia[2]. The Dolomiti Energia Group is a multi utility with more than 1300 employees, leader in the sectors of energy, gas, water, garbage collection in the Trentino Province. Its subsidiaries, Dolomity Energia and SET Distribuzione, are active in the production and distribution of energy respectively. Around 90% of the energy produced is renewable and comes from hydroelectric plants. Dolomiti Energia can provide all the energy needed by the Trentino Province.

The APSS data centre offers critical services that cannot undergo any disruption. For this reason, the first experimentation will be conducted in a dedicated and isolated environment. Three representative services, albeit non critical, has been selected among the ones provided by the data centre:

- GRU: web application for ticketing management. It manages the interventions on ICT related requests.
- TINCA: SLA calculation on trouble tickets related to support requests handled by external companies (service providers).
- SISWeb: reporting tool on some indicators like quality of water, quality of meat etc. It is oriented to risk prevention.

These three applications have been chosen also because they do not handle sensible data and because the source code is available for modifications. For the trial, they will be deployed in a staging area provided by APSS, along with the upgraded version of Plug4Green.

3.4 Dissemination

Communication is an important part of the life of a researcher. As such, we will disseminate research results in papers for conferences or journals. Here is a selection of the possible peer reviewed conferences and workshops:

- E-Energy - International Conference on Future Energy Systems
- ISCIS - International Symposium on Computer and Information Sciences
- ICSE - International Conference on Software Engineering

[2]www.gruppodolomitienergia.it

- CP - International Conference on Principles and Practice of Constraint Programming
- ICFP - International Conference on Functional Programming
- E2DC - International Workshop on Energy-Efficient Data Centres
- EuroEcoDC - International workshop on European actions towards eco-friendly data centers

We also target top quality journals, such as:

- Elsevier Future Generation Computer Systems journal
- Elsevier Journal on Sustainable Computing
- Springer Journal on Cloud Computing
- Springer Journal of Internet Services and Applications
- IEEE Transactions on Cloud Computing

Those journals periodically makes special issues that are focused on a specific subject, which might also be interesting considering.

Dissemination of knowledge about renewable energy awareness in data centres will also be performed locally, through our network of partners (APSS, Trentino Network, Dolomiti Energia) and the local and national press.

4. PRELIMINARY RESULTS

We present in this section the preliminary results obtained with the Plug4Green prototype, in particular the design selected (see Section 4.1). We also give some insight into the implementation (see Section 4.2). As a preliminary validation for our approach, an early version of Plug4Green has already been tested[3], some of result will be presented here (see Section 4.3).

4.1 Design

By design, Plug4Green is extensible. The architecture chosen allows to easily extend the engine by adding new concerns, without modifying the underlying algorithms. In particular, new constraints can be added straightforwardly, as we showed by implementing 23 constraints commonly encountered in data centres, including energy-oriented ones. As can be seen in Figure 1, Plug4Green has the following inputs:

- The *SLAs*
- The *data centre configuration*
- A *Single Allocation* request
- Or a *Global Optimisation* request

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 consists of a set of actions such as *powering on*, *powering off*, *waking up* and *putting in idle mode* a server, and *migrating* a VM, that satisfies all the constraints and minimizes the current objective. The objective can be to minimize either the power consumption

[3]The experimentation have been carried out within the EU FP7 project FIT4Green: www.fit4Green.eu

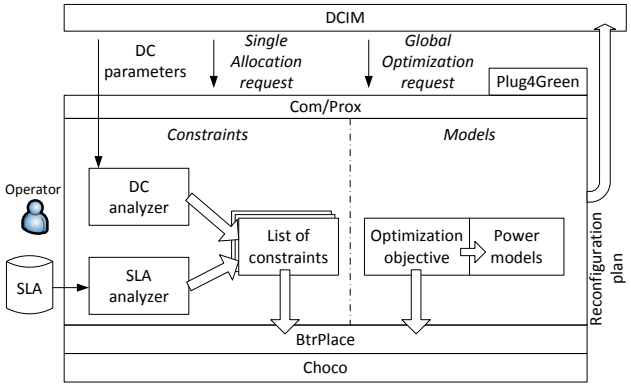

Figure 1: Plug4Green architecture

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 (every ten minutes in our experimentation) and Plug4Green will return a reconfiguration plan. In manual mode, the data centre operator has the possibility to accept or reject this reconfiguration plan, while in automatic mode, it is enacted automatically. Plug4Green will then execute the reconfiguration plan in order to reduce the overall consumption of the data centre (either power consumption or gas emission) while also respecting the SLAs. The *Com/Prox* layer ensures that Plug4Green can be plugged easily to different existing DCIM: its the only part that must be updated when adapting the software for a new DCIM. Currently, Plug4Green can be integrated into VMWare[4], Eucalyptus[5], and HP Matrix Operating Environment[6] infrastructures. Plug4Green is based on the flexible consolidation manager BtrPlace [8].

A huge number of SLAs contracts exists in data centres. Furthermore, those SLAs are more and more extended with energy concerns. It is thus difficult to create a management framework that would come with all those SLAs predefined and hard-coded, without possibility of adding new semantics easily. 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.

[4]`http://www.vmware.com`
[5]`http://eucalyptus.com`
[6]`http://h18004.www1.hp.com/products/solutions/insightdynamics/overview.html`

Category	Constraint	The Constraint enforces...
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 physical CPU
	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)
	MaxServerAvgvRAMperPhyRAM	Same as MaxVRAMperPhyRAM 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

SLAs often comes 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 needs to translate it into our SLA schema. The SLA schema is a format in XML allowing the CPT to use the predefined constraints detailed in Table 1. Once the SLA XML file is ready, it can be submitted to Plug4Green. The SLA constraints will then be translated automatically to lower level CP constraints, with the process shown in Figure 2.

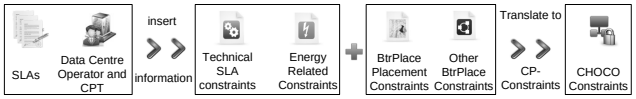

Figure 2: Translation of the SLA contract into technical SLAs and then to Constraints

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.

4.2 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.

4.2.1 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. 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. By using CP, we achieve the important goal of separating two different concerns: 1) the development of a placement objective and 2) the development of constraints that will specialise the objective.

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:

```
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*

Using the CP operators *eq*, *plus* and *mult* we first define

Energy related variables	
P	Future global power consumption of the data centre federation
$P(s)$	Future power consumption of a server s
P_{net}	Power consumption of the network
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

the power of a server P (one of the variables in Table 2). α and β are defined in the Section 4.2.3. We then create the constraint that this power P must be less or equal than a threshold $maxServerPower$. This constraint is posted to the *model*, which will collect all the constraints. As we can see, the definition of the constraint does not include *how* we want to solve it, only *what* we desire. This is a clear separation of concerns: we were not obliged to revise the solving algorithm to define this constraint. Furthermore, it is very short: it took some hours for an engineer to create and test it.

4.2.2 From SLA to constraints

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 (in Watts) that a server or a group of servers can consume. 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, through the constraints *DelayBetweenMove* and *VMPaybackTime*. 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.

4.2.3 Optimisation Objectives

In this section, we present the implementation in Plug4Green of two different objectives:

- $minEnergy$ to reduce the data centre energy consumption as much as possible
- $minGasEmission$ to reduce the data centre CO_2 emission as much as possible

Plug4Green relies on the power prediction models described in [14] and [15]. 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:

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

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:

$$P = \sum_{d=1}^{D} (PUE_d \times \sum_{i=1}^{n} P(s_i^k)) \quad (2)$$

To reduce energy consumption, migrating VMs to consolidate them is a common solution. However, migrating a

VM has an energy cost: this effect need to be integrated in the power objective function of Plug4Green. This way, Plug4Green will be able to avoid migrating 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 [16]. We also include an energy penalty for switching on and off a server: $E_{onoff}(j)$. 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: this is thus lost energy.

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 (every 10 minutes in our experimentation) and to take into account changes in the data centre like VM termination, Plug4Green will recompute the power objective. Using equation (2), Plug4Green can compute 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:

$$E_{tot} = (P_{bef} - P_{aft}) \times T_{reconf} - \sum_{i=1}^{p} E_{move}(i) - \sum_{j=1}^{n} E_{onoff}(j) \quad (3)$$

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.

4.2.4 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 4.2.2) 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 1) the position of the VMs on the servers and 2) the state of the servers.

For both presented objectives, their search heuristics suggest to the solver 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. For the *minEnergy* objective, the PUE of the facility containing the server is considered. For the *minGasEmission* objective, the CUE of the facility is considered.

In practice, the servers are sorted in ascending order depending on their load. In case of equivalence, their energy efficiency is used using the power consumption prediction models. The list of VMs running on these servers is then created accordingly. To choose a value to try for each VM placement variable, the list of servers is browsed backward and the first satisfying node is used. Once the new placement for each VM is computed, the heuristics makes the solver try to turn off unused servers.

4.3 Framework Evaluation

In this section we demonstrate the impact of Plug4green on the power consumption or the gas emission of an heterogeneous data centre federation running an industrial workload. Secondly, we evaluate within a simulator the scalability of Plug4Green for a data centre with up to 2500 servers.

4.3.1 Efficiency of Plug4Green

To evaluate the practical efficiency of Plug4Green in an environment as realistic as possible, a trial has been performed by the HP Innovation Centre Italy with a state-of-the-art cloud stack running two workloads derived from industrial traces.

The cloud test bed simulates an heterogeneous data centre federation. It is composed of two racks (see Table 3), each embedding an 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*.

	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	24GB	24GB

Table 3: Characteristics of the Racks/Enclosures

Plug4Green has been evaluated against 2 synthetic workloads derived from real traces inside the private cloud of a corporation in Italy.

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. A *ad-hoc* heuristic deploys the VMs on servers with a load-balancing placement objective. Idle servers are not turned off and VMs are not be migrated. In the "P4G same PUE" scenario,

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

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 takes the benefit from servers in DC1 that are more energy-efficient. Figure 3 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.

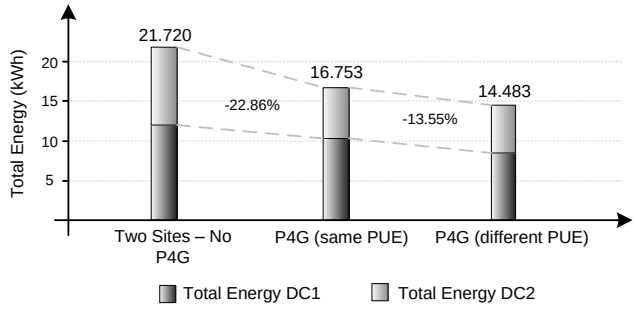

Figure 3: Energy consumption of two data centres with different PUE values

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

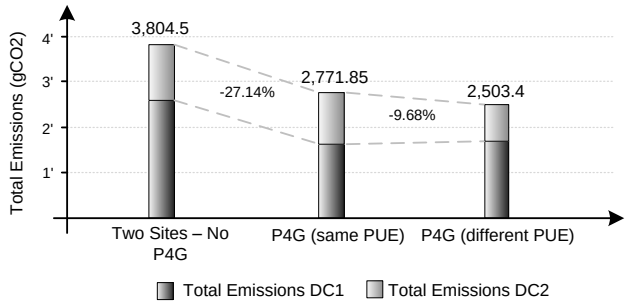

Figure 4: Energy consumption of two data centres with different CUE values

4.3.2 Scalability of Plug4Green

The scalability of Plug4Green is determined by the number of VMs and servers composing the configuration but also the placement constraints that specialize the algorithm. 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. The amount of VMs in each configuration equals five times the number of servers, according to a consolidation ratio observed in industry [17]. Finally, their initial placement is computed randomly but ensures their SLA is satisfied.

We generate 5 sets of configurations that are composed by 500 to 2,500 servers. Each set is composed by 50 configurations that differ in the template used by the VMs and their initial placement. To evaluate the impact of the data centre size and the placement constraints, we run Plug4Green on each configuration with the constraints evaluated in Section 4.3.

Figures 5 and 6 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 and will thus be prevented to migrate.

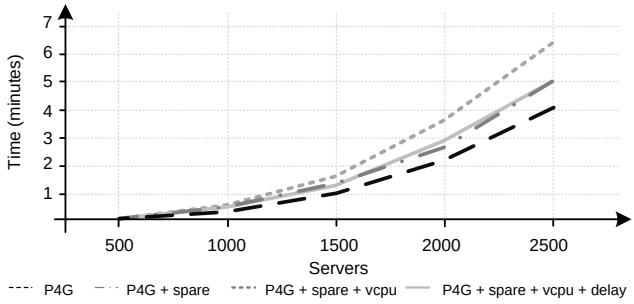

Figure 5: Solving duration to compute the improved configurations

Figure 5 shows the solving time increases exponentially with regards to the data centre size. This is expected as the problem addressed by Plug4Green is NP-hard. We observe Plug4Green computes an improved configuration in one minute with a data centre of up to 1,500 servers running 7,500 VMs. Doubling the size of the data centre requires approximately 4 times more time. Below 1000 servers, 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. However, the *SpareCPUs* and the *MaxVCPUperCore* constraints increase the computation time by 25% each.

Figure 6 shows the energy consumption of the improved configurations. We first observe that the *SpareCPUs* and the *MaxVCPUperCore* constraints do not alter the quality of the improved configurations. For the *SpareCPUs* constraint,

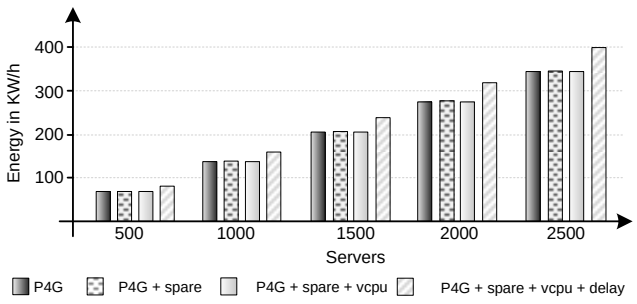

Figure 6: Energy consumption of the improved configurations

Plug4Green was able to keep free the requested amount of PCPU capacity without having to turn on additional servers.

5. CONCLUSION

In this paper we presented a research plan to create an energy aware infrastructure for data centres. There are two aspects to this objective: first, reduce the overall energy consumption in data centres, and second, increase the usage of renewable energies. We presented our plan to enhance Plug4Green, a flexible energy-aware VM manager, in particular to allow it to increase the usage of renewable energies in data centres. Thanks to constraint programming, Plug4Green can be easily specialized to support various combinations of SLAs, but also different power models and energy policies. The flexibility of Plug4Green has been verified through the implementation of 23 meaningful SLAs and 2 energy policies. Its practical effectiveness for saving energy has been evaluated on an industrial testbed. The default version of Plug4Green reduced the power consumption and the greenhouse gas emission by 33% and 34% respectively. Finally, a scalability experiment in a simulated environment showed the ability of Plug4Green to compute an improved placement for 7,500 VMs running on 1,500 servers in a minute, while respecting their SLA.

Acknowledgments & Availability

The author would like to thanks the University of Trento, the EU FP7 projects FIT4Green and DC4Cities, and the Create-Net research centre. Evaluations presented in this paper were carried out by HP Innovation Centre Milan and INRIA.

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

6. REFERENCES

[1] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.

[2] Dang Minh Quan, Robert Basmadjian, Hermann de Meer, Ricardo Lent, Toktam Mahmoodi, Domenico Sannelli, Federico Mezza, Luigi Telesca, and Corenten Dupont. Energy efficient resource allocation strategy for cloud data centres. In Erol Gelenbe, Ricardo Lent, and Georgia Sakellari, editors, *Computer and Information Sciences II*, pages 133–141. Springer London, 2012.

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

[4] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin 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, pages 229–242, Berkeley, CA, USA, 2007. USENIX Association.

[5] Akshat Verma, Puneet Ahuja, and Anindya Neogi. Power-aware dynamic placement of hpc applications. In *Proceedings of the 22nd annual international conference on Supercomputing*, ICS '08, pages 175–184, New York, NY, USA, 2008. ACM.

[6] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, May 2012.

[7] Fabien Hermenier, Sophie Demassey, and Xavier Lorca. Bin repacking scheduling in virtualized datacenters. In *Proceedings of the 17th international conference on Principles and practice of constraint programming*, CP'11, pages 27–41, Berlin, Heidelberg, 2011. Springer-Verlag.

[8] Fabien Hermenier, Julia Lawall, and Gilles Muller. Btrplace: A flexible consolidation manager for highly available applications. *IEEE Transactions on Dependable and Secure Computing*, 10(5), 2013.

[9] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller, and Julia Lawall. Entropy: a consolidation manager for clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '09, pages 41–50. ACM, 2009.

[10] E. Bin, O. Biran, O. Boni, E. Hadad, E.K. Kolodner, Y. Moatti, and D.H. Lorenz. Guaranteeing high availability goals for virtual machine placement. In *Distributed Computing Systems (ICDCS), 2011 31st International Conference on*, pages 700–709, 2011.

[11] Roman Krogt, Jacob Feldman, James Little, and David Stynes. An integrated business rules and constraints approach to data centre capacity management. In David Cohen, editor, *Principles and Practice of Constraint Programming âĂŞ CP 2010*, volume 6308 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010.

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

[13] Corentin Dupont, Thomas Schulze, Giovanni Giuliani, Andrey Somov, and Fabien 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, pages 4:1–4:10. ACM, 2012.

[14] Robert Basmadjian, Hermann de Meer, Ricardo Lent, and Giovanni Giuliani. Cloud computing and its interest in saving energy: the use case of a private cloud. *Journal of Cloud Computing*, 1(1), 2010.

[15] Robert Basmadjian, Nasir Ali, Florian Niedermeier, Hermann de Meer, and Giovanni 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, pages 1–10, New York, NY, USA, 2011. ACM.

[16] William Voorsluys, James Broberg, Srikumar Venugopal, and Rajkumar 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.

[17] Virtualization penetration rate in the enterprise. Technical report, Veeam Software, 2011.