IT Optimization for Datacenters Under Renewable Power Constraint

Stéphane Caux, Paul Renaud-Goud, Gustavo Rostirolla, Patricia Stolf

IT Optimization for Datacenters Under Renewable Power Constraint

Stephane Caux[1], Paul Renaud-Goud[2], Gustavo Rostirolla[1,2(✉)], and Patricia Stolf[2]

[1] LAPLACE, Université de Toulouse, CNRS, Toulouse, France
{caux,gustavo.rostirolla}@laplace.univ-tlse.fr
[2] IRIT, Université de Toulouse, 31062 Toulouse, France
{paul.renaud.goud,stolf}@irit.fr

Abstract. Nowadays, datacenters are one of the most energy consuming facilities due to the increase of cloud, web-services and high performance computing demands all over the world. To be clean and to be with no connection to the grid, datacenters projects try to feed electricity with renewable energy sources and storage elements. Nevertheless, due to the intermittent nature of these power sources, most of the works still rely on grid as a backup. This paper presents a model that considers the datacenter workload and the several moments where renewable energy could be engaged by the power side without grid. We propose to optimize the IT scheduling to execute tasks within a given power envelope of only renewable energy as a constraint.

Keywords: Cloud computing · Renewable energy · Scheduling

1 Introduction

Datacenters are now known to be one of the biggest actors when talking about energy consumption [1]. In 2006, particularly, datacenters were responsible for consuming 61.4 billion kWh in the United States [2]. In another study [3], datacenters are in charge of consuming about 1.3% of world's electricity consumption. Datacenters are currently consuming more energy than the entire United Kingdom, and our needs are increasing.

Supplying datacenters with clean-to-use renewable energy is therefore essential to help mitigate climate change. The vast majority of cloud provider companies that claim to use green energy supply on their datacenters consider the classical grid, and deploy the solar panels/wind turbines somewhere else and sell the energy to electricity companies [4], which incurs in energy losses when the electricity travels throughout the grid. Even though several efforts have been conducted at the computing level in datacenters partially powered by renewable energy sources, the scheduling considering the variations in the power production without the grid can still be widely explored. In this paper we consider a datacenter powered only with renewable energy.

https://doi.org/10.1007/978-3-319-96983-1_24

Since energy efficiency in datacenters is directly related to the resource consumption of a computing node [5], performance optimization and an efficient load scheduling is essential for energy saving. Today, we observe the use of cloud computing as the basis of datacenters, either in a public or private fashion. The task management is first optimized by Virtual Machine (VM) management [6], where a task should be placed considering an energy consumption model to describe the task's consumption, depending on the resource description (processor and memory power characteristics) and task's demand (resources usage) while respecting the Quality of Service (QoS - in our case their due dates).

To address the IT load scheduling while considering the renewable energy available we propose Renewable Energy Constrained Optimization (RECO). RECO is a module to schedule batch tasks, which are characterized by their release time, due date and resource demand, in a cloud datacenter while respecting a power envelope. This envelope represents an estimation which would be provided by a power decision module and is the expected power production based on weather forecasts, states of charge of storage elements and other power production characteristics. We also highlight that this RECO module is intended to be used as part of the ANR Datazero project[1]. RECO aims at maximizing the Quality of Service with a constraint on electrical power. There are several possible power envelopes which could be generated using only renewable energy sources and the different moments when storage elements can be engaged. This interaction between datacenter electrical consumption and electrical power sources part is fundamental to profit as much as possible from the renewable energy sources. We propose and evaluate this RECO module with a comparison between classical greedy algorithms and meta-heuristics constrained by power envelopes.

The remainder of this article will present the classical approaches on scheduling with and without renewable energy sources in Sect. 2. In Sect. 3 the problem formulation is presented in details, followed by the resolution in Sect. 4 and the evaluation methodology as well as the results obtained are presented in Sect. 5. Finally, Sect. 6 presents final remarks, highlights the contributions with quantitative data and also directions for future works.

2 Related Work

Several techniques exist to save energy [5,7]. In this section some of these research initiatives are presented, mainly related to the energy aware task scheduling in datacenters. In this sense, several authors tackle this problem using heuristics to schedule tasks trying to reduce the energy consumption in a cloud datacenter, some of which consider also the use of renewable energy. Below we present some initiatives that utilizes green energy to the datacenter in order to maximize the green energy usage.

Goiri et al. proposes GreenSlot [8] which focus on batch jobs and GreenHadoop [9] focused on MapReduce jobs scheduling for a datacenter powered

[1] http://www.datazero.org.

by photovoltaic panels and the electrical grid. The schedulers are based on a predicted amount of solar energy that will be available, and aims to maximize the green energy consumption while meeting the jobs constraints. If grid energy must be used to avoid due date violations, the scheduler finds the cheapest point. Aksanli et al. [10] proposes an adaptive datacenter job scheduler which also utilizes short term prediction but in the case of solar and wind energy production. The aim of the scheduler is to reduce the number of canceled or violated jobs, and improve the efficiency of the green energy usage. Liu et al. [11] investigates the feasibility of powering cloud datacenters using renewable energy. The study focus on geographical load balancing, and the optimal mix of renewable energy using a concept called "follow the renewables" in which the workload is migrated among datacenters to improve the renewable energy usage. Finally, Beldiceanu et al. [4] presents EpoCloud, a prototype aims at optimizing the energy consumption of mono-site cloud datacenters connected to the regular electrical grid and to renewable energy sources, aiming to find the best trade-off between energy cost and QoS degradation using application reconfiguration or jobs suspension along with Vary-On/Vary-Off (VOVO) policy which dynamically turn on/off the computing resources. Sharma et al. [12] presents Blink, a way to handle intermittent power constraints activating and deactivating servers. For example, a system that blinks every 30 seconds is on for 30 seconds and then off for 30 seconds. This approach can be useful for some web applications, but not realistic for the vast majority of applications running in cloud platforms.

As it can be observed, techniques are employed in order to reduce the brown energy consumption [5], such as node consolidation, DVFS (processor voltage and frequency variation) and some authors also take profit of heterogeneity in the datacenter. Nevertheless, with exception of Sharma et al. [12] the authors always consider the grid as a backup and not a datacenter powered only by renewable energy sources, and the fluctuations that could occur in the power production. The scheduling over several possible power profiles allow us to see the impact on metrics such as QoS and the usage of renewable energy. To do so, a module to schedule tasks in a cloud datacenter is proposed in this paper while respecting the several possible power envelopes, minimizing the number of due date violations. For comparison purposes we also explore classical greedy algorithms and meta-heuristics constrained by a provided power envelope.

3 Core Problem Formulation

3.1 The Principles of the RECO Module

IT scheduling problems consist in allocating tasks on the IT resources under constraints depending on the IT platform current state and on energy availability. Several levels of decision are concerned as IT resource management (server switch on/off, process migration, voltage and frequency scaling, etc.). On the other side, we have the power systems where several power profiles could be provided, depending on the moment when the renewable energy is produced and the batteries are engaged for instance.

RECO focuses on integrating both power and computing systems to provide a power constrained optimization using power envelopes, which is applicable in the context of projects such as Datazero. The power envelope is considered as an input of the IT scheduling problem. The objective is to optimize the tasks placement in a cloud datacenter respecting a power envelope provided by the Power Management while maximizing the QoS (in our case, minimizing the due date violations).

RECO can be triggered when a new task arrives or due to some changes in the power envelope. It decides which task will be executed on which resource, when and at which frequency (using DVFS), and also when each node will be turned on or off. RECO ensures that the placement will respect a power envelope engaged by a power module, while minimizing the number of tasks that will be violated (finishing after the due date).

In the next sections, the models for IT and power characteristics and the proposed scheduling approaches are exposed in details.

3.2 IT Management Model

In this work we focus mainly on batch tasks. The IT system receives a set of n tasks $\{\mathcal{T}_j\}_{j\in\{1,\dots,n\}}$, characterized by the following information: et_j represents the execution time of task $\mathcal{T}_j$ running at a reference frequency $F_{1,1}^{(1)}$ (see later), mem_j is the requested memory, rt_j represents the release time of the task (the moment when $\mathcal{T}_j$ can start to be executed), and d_j represents the due date of this task (the moment when $\mathcal{T}_j$ must be finished).

M multi-processor hosts $\{\mathcal{H}_h\}_{h\in\{1,\dots,M\}}$ populate the datacenter, while each host $\mathcal{H}_h$ is composed of C_h processors equipped with DVFS, each of them exposing M_h memory. The power dissipated by $\mathcal{H}_h$ can be computed based on Mudge [13]:

$$P_h = \begin{cases} P_h^{(\text{idle})} + \sum_{h=1}^{C_h} run_{h,p} \cdot P_h^{(\text{dyn})} \cdot (f_{h,p})^3 & \text{if } \mathrm{s_h} = on \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where s_h determines whether $\mathcal{H}_h$ is *on* or *off*, $P_h^{(\text{idle})}$ is the idle power, $run_{h,p}$ is a boolean describing whether there is a task running on the processor, $P_h^{(\text{dyn})}$ is a host-dependent coefficient, and $f_{h,p}$ is the clock frequency of processor p on host h.

Every processor have a set of available frequencies $\mathcal{F}_{h,p} = \{F_{h,p}^{(1)}, \dots, F_{h,p}^{(FM_{h,p})}\}$, in such a way that at any instant, $f_{h,p} \in \mathcal{F}_{h,p}$. Finally, note that under any clock frequency, a power overhead of $P_h^{(on)}$ (resp. $P_h^{(off)}$) is paid during $t_h^{(on)}$ (resp. $t_h^{(off)}$) when $\mathcal{H}_h$ is turned *on* (resp. *off*).

We consider that an external Power Management module sends a set of piecewise power envelopes in a *time window* $[t_{i,\min}, t_{i,\max}]$ where each envelope i is described with time steps $\{t_{i,l}\}_{l\in\{0,\dots,N\}}$ (where $t_{i,0} = t_{i,\min}$ and $t_{i,N} = t_{i,\max}$)

and power values. The available electrical power, constant on each $[t_{i,l}, t_{i,l+1}]$, is given in Watts and N represents the granularity. Here we also loosely call these time intervals as *steps*.

3.3 Objective

The aim is to find when and at which frequency to run every task, *i.e.* to find assignment functions σ_{proc} , σ_{host} and σ_{freq} expressing that $\mathcal{T}_j$ runs on processor $\sigma_{\text{proc}}(j)$ of host $\sigma_{\text{host}}(j)$ at frequency $F^{(\sigma_{\text{freq}}(j))}_{\sigma_{\text{host}}(j),\sigma_{\text{proc}}(j)}$, and a starting point function st expressing that $\mathcal{T}_j$ starts at time $st(j)$. We denote by $ft(j)$ the finish time of $\mathcal{T}_j$, hence, for all j:

$$ft(j) = st(j) + \frac{F^{(\sigma_{\text{freq}}(j))}_{\sigma_{\text{host}}(j),\sigma_{\text{proc}}(j)}}{F^{(1)}_{1,1}} \cdot et_j. \tag{2}$$

The problem can then be formulated as follows: minimize $\sum_j \max(0, ft(j) - d_j)$, while fulfilling memory and power constraints.

4 Core Problem Resolution

Finding a mapping of the tasks onto the processors such that no due date constraint is violated is an NP-complete problem, while DVFS is not enabled and memory is not taken into account, even with two processors. In this way, we focus on approximation methods. More specifically, we explore Greedy Heuristics (GH) and Genetic Algorithms (GA) as a way to validate our proposal. GH can provide locally optimal decisions, and in general have a short execution time. On the other hand, the combinations of choices locally optimal do not always lead to a global optimum. The second approach (GA), can provide a large number of adapted solutions and also makes possible to approach a local minimum starting from an existing solution. Nevertheless, the problem of GA methods can be the execution time on large scale problems. In this work we propose a time window approach. More specifically, an off-line resource allocation problem is considered with a fixed set of tasks that have constant resource needs.

The difference from regular scheduling algorithms is that in this case we need considering the power envelope as a constraint. To do so, the implemented algorithms use a power check function which is responsible for evaluating if a task can be scheduled in a given processing element on the desired time interval. It returns how much power would be consumed to schedule the task using a specific processor and frequency. Hereafter, two different approaches that provide scheduling possibilities are presented but this model is not limited to it and new approaches could be used as long as they rely on the presented function.

For GH, we considered three versions of the Best Fit, where we use different sort task functions. It tries to fit the tasks in the node that presents the smallest power consumption, respecting the power envelope and resource constraints, and three versions of the First Fit algorithm which schedules a task at the first

available node which can finish the task before the due date. The difference among the three versions of each algorithm is the way that the tasks are sorted: (i) Due date, closest task first; (ii) Arrival time, first task that arrives is the first to be scheduled; and (iii) Task size, longest one first. Even though the changes occur only in the task ordering, the impact on the results can be significant. All considered GH algorithms must respect the power envelope, meaning that if there is not enough power in a given time step to power a machine, this task will be delayed until the next time step in which a possible solution is found (increasing the start step).

Regarding the GA we propose two variations, the first one where the fitness function consists only in reducing the number of due date violations, and the second one uses a weight based approach, also trying to minimize the power consumption in a Mixed Objective (hereafter called MPGA - MultiPhase Genetic Algorithm and MPGA-MO - MultiPhase Genetic Algorithm Mixed Objective, respectively). Equation 3 is used to normalize all metrics considered for each chromosome $\mathcal{C}_k$, described below, where $M^{(\max)}$ is the maximum value for a given metric, $M^{(\min)}$ is the minimum, and M_k is the value of the k^{th} chromosome. The normalized values are then inputs in Eq. 4 where DD_k is the normalized due date violations and E_k is the normalized energy consumption. The metrics should be weighted using α, depending on the importance of the objective (for MPGA the only metric considered is the number of due date violations, *i.e.* α is equal to 1).

$$M_k^{(\text{norm})} = \frac{M^{(\max)} - M_k}{M^{(\max)} - M^{(\min)}} \tag{3}$$

$$\mathit{fitness}_k = \alpha \times DD_k + (1 - \alpha) \times E_k \tag{4}$$

In both cases each chromosome represents a scheduling possibility for the given power profile. Figure 1 presents an example of crossover operation (Algorithm 1) where each gene represents a task and the value is the node where it will be executed. For the crossover operation we consider two points crossover since it allows the change of a higher number of genes in a single operation, and the selection consists in tournament selection, which allows the best fitted genes to survive. After that, the processor, frequency and time are assigned using a greedy algorithm. To improve the execution time of both GAs (the verification of the power available occurs for each step in the power envelope) we also use two different power envelopes, the first one provides a rough scheduling based on an aggregation of the initially provided envelope, reducing in this case the number of *steps*. After obtaining an initial placement, a fine grained power envelope (smaller steps) is used to absorb power peaks and respect the given power envelope.

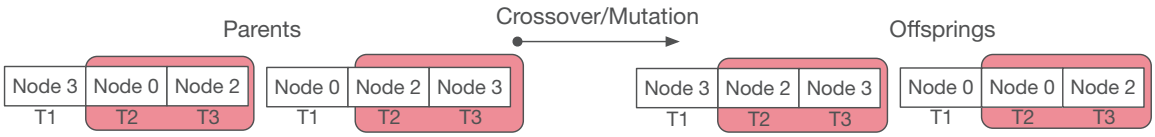


Fig. 1. Genetic algorithm chromosome representation and crossover example.

A pseudocode of the GA used is presented in Algorithm 1 where it can be seen the generation of the simplified envelope in line 2 (assigned to individuals in line 4), the first execution from line 6 to 11, and the execution with the detailed power envelope and the respective stopping criteria from line 12 to 19. The stopping criteria for the MPGA, since it only considers the number of due date violations, is when it has at least one chromosome that has no violation the execution can be stopped, or the maximum number of generations is reached. For the second algorithm (MPGA-MO) the stopping criteria is only the number of generations, since the minimum energy to schedule the tasks in advance cannot be defined easily.

Algorithm 1. Multiphase genetic algorithm pseudocode.

```
   input : Set of tasks in queue, set of resources available, power envelope for the
           window, selection method, population size, number of generations first phase,
           number of generations second phase, number of simplified steps, mutation
           probability, crossover probability
   output: Tasks scheduled, actions to be performed in nodes, QoS metrics, power
           consumption estimation
1  begin
2      simplifiedPowerEnvelope = generateSimplifiedEnvelope(powerEnvelope,nSteps);
       /* First Phase - Simplified Power Envelope                          */
3      foreach Individual i in population do
4          i.setPowerEnvelope(simplifiedPowerEnvelope.copy);
5      end
6      generateInitialPopulation();
7      for (g=0; g < generationsFirstPhase; g++) do
8          scheduleAndCheckConstraints(individuals);
9          calculateFitness(individuals);
10         selectionMethod.select(individuals);
11     end
       /* Second Phase - Detailed Power Envelope                           */
12     foreach Individual i in population do
13         i.setPowerEnvelope(powerEnvelope.copy);
14     end
15     while StopCriteriaNotReached do
16         scheduleAndCheckConstraints(individuals);
17         calculateFitness(individuals);
18         selectionMethod.select(individuals);
19     end
20 end
```

When a set of individuals of a generation is computed, the greedy algorithms is used to perform the time schedule and DVFS adjustment (*scheduleAndCheckConstraints* called in lines 8 and 16). In a simplified manner, how the tasks would be allocated in a processor is presented in Fig. 2 where we illustrate a node with two processors. In (a) we present the scheduling after

the greedy algorithm that defines the time and processor inside a node is executed. The aim of this greedy algorithm is to align the execution of the processors of the same node to be able to switch it off. First we populate an associative array with all the tasks and the time intervals where they can be scheduled. After, we get the first unscheduled task and compare if there is another task which the time to be schedule intercepts this time interval. The algorithm evaluates then, what is the earliest start step in which the tasks can be allocated and not violated. Finally, the algorithm finds a free processor inside the node and schedule the tasks in parallel (as illustrated in (b) by $\mathcal{T}_1$ in Processor 1 and $\mathcal{T}_2$ and $\mathcal{T}_3$ in Processor 2. We also highlight that the algorithm always verifies the power envelope and resources constraints.

In Fig. 2(c) we show a per processors DVFS where we reduce the frequency of Processor 2 in this case, to reduce the power consumption, and consequently increasing the execution time of tasks $\mathcal{T}_2$, $\mathcal{T}_3$. The frequency in this case is only reduced if the due date is not violated. This DVFS control does not impact the idle power consumption of a node, allowing an easy consolidation of nodes where more energy saving can be obtained. In this sense, at the end of the task placement and DVFS adjustment we also calculate when each node can be turned off in order to reduce the power consumption without impacting the system performance.

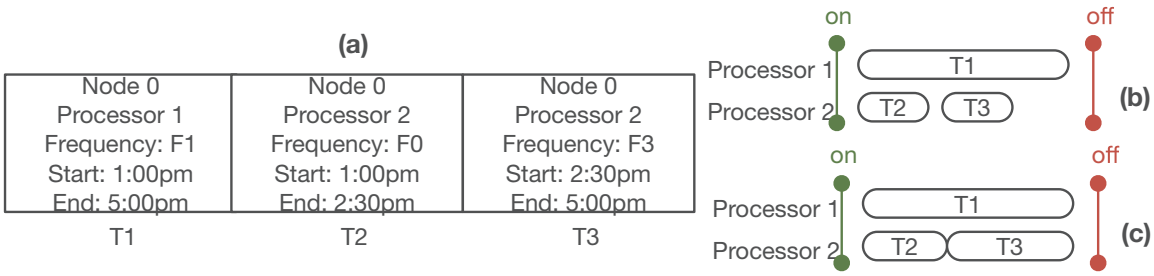


Fig. 2. Tasks allocation inside a node with two processing elements using greedy scheduling inside GA (a), and DVFS adjustment where (b) is before DVFS and (c) after DVFS adjustment.

5 Evaluation Methodology and Results

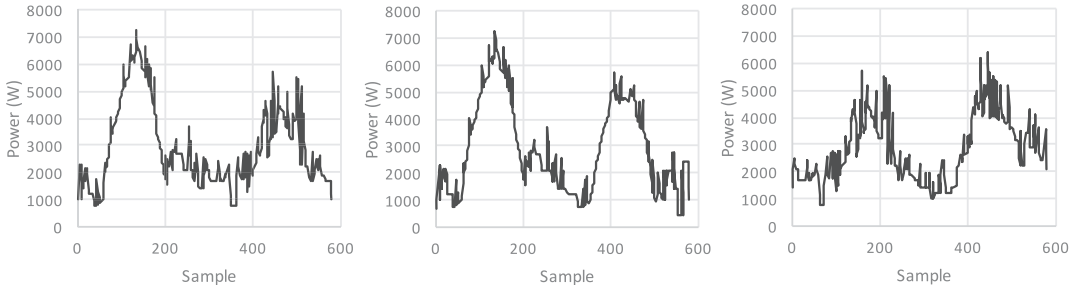
5.1 Methodology

To validate RECO we simulated an IT and Power production infrastructure based on the prototype presented in the previous section. The DCWoRMS simulator and the other modules are executed on the same machine. The IT infrastructure inside the simulator is based on Villebonnet et al. [14], more specifically we are using 30 hosts (15 of each kind) and the power consumption values of Paravance and Taurus clusters from Grid5000[2]. We consider $P^{(\mathrm{dyn})} = 4.725\,W \cdot s^3$

[2] https://www.grid5000.fr/.

(see Eq. 1) and $P^{(\text{idle})} = 69.9\,W$ for Paravance and $P^{(\text{dyn})} = 5.255\,W \cdot s^3$ and $P^{(\text{idle})} = 95.8\,W$ for Taurus. For Paravance we considered $P^{(on)} = 112.91\,W$ over $t^{(on)} = 189\,s$ and for $P^{(off)} = 65.7\,W$ over $t^{(off)} = 10\,s$. For Taurus we considered $P^{(on)} = 125.78\,W$ over $t^{(on)} = 164\,s$ and for $P^{(off)} = 106.63\,W$ over $t^{(off)} = 11\,s$.

Regarding the GA, we bound the number of generations to 100 (resp. 400) with the simplified power envelope (resp. with the original power envelope) and the population size to 100 individuals. The probabilities for crossover and mutation are 0.9 and 0.3 respectively. For the MPGA-MO we consider $\alpha = 0.9$ where the main objective is minimize the due date violations. For the Google based workload [15] generator we use a two-day window (i.e. all the tasks have to be executed inside this interval) to generate 3 different workloads with 234, 569 and 1029 tasks. Each workload is scheduled with 3 different power profiles as observed in Fig. 3. Profile I with peak production of 7249 W and average of 2879W, Profile II peak production of 7249 W and average of 2893 W and Profile III with peak production of 6387 W and average of 2756W. Even though the values are similar, the moment in which the power is delivered is different, as observed in Fig. 3.



(a) Profile I. (b) Profile II. (c) Profile III.

Fig. 3. Graphical representation of the three power profiles.

5.2 Results Evaluation

In Fig. 4 we present the number of due date violations (a) the total time violated (b) and energy consumption (c) for all the proposed workloads for best fit and genetic algorithms (first fit is presented only in text for better visualization), considering the three different power profiles.

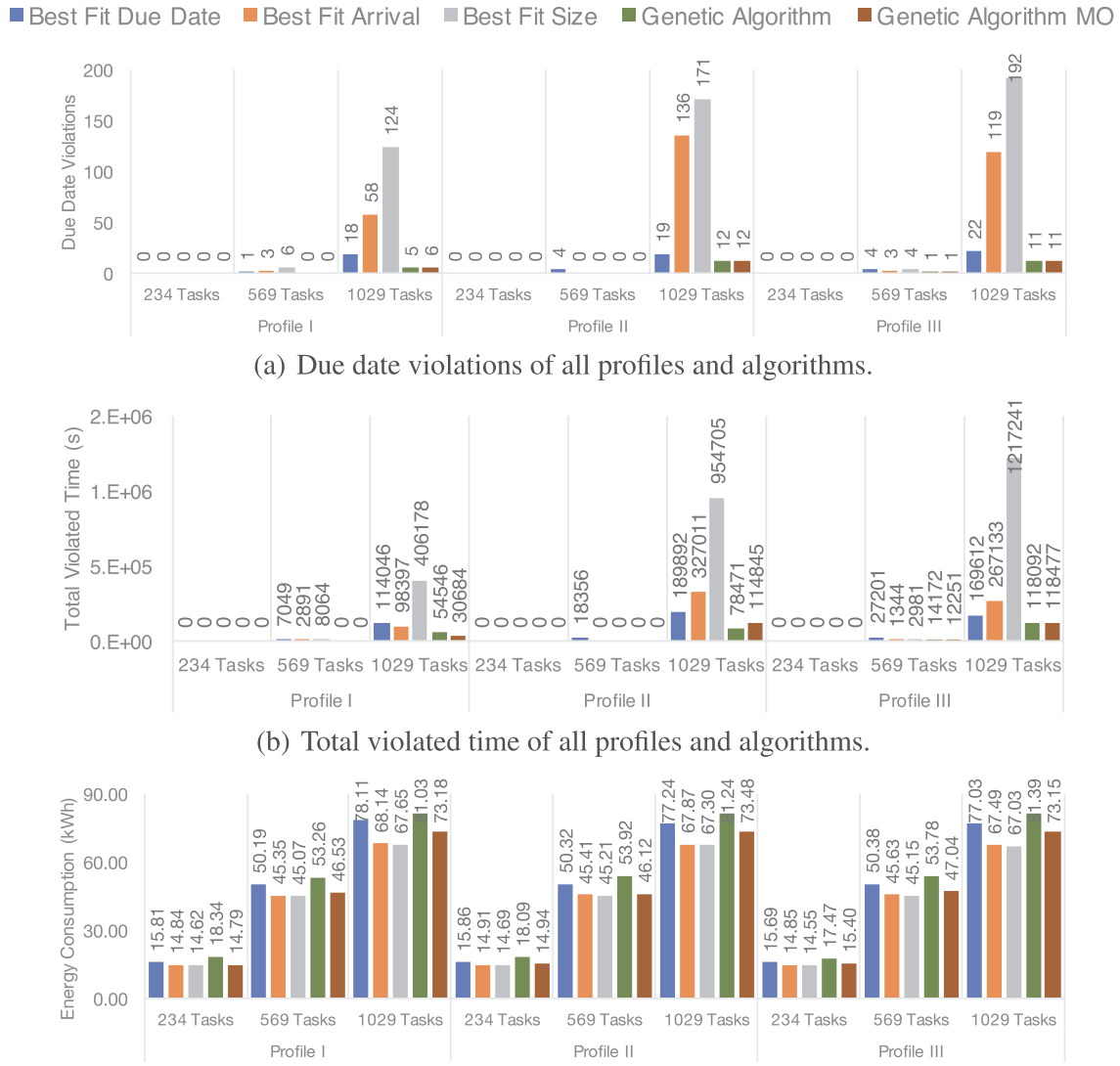
Considering the three power profiles with only 234 tasks, almost all algorithms, even with the power constraint, can reduce the number of violations to 0 and keep the energy consumption around 15 kWh. The exceptions in this case are the first fit algorithms which have a higher energy consumption (around 18 kWh) and one violation (198s of total time violated) with Profile II. As the number of tasks increases an expected degradation of performance of both first fit and best fit algorithms is observed when compared to the GA. When considering 1029 tasks we have in Profile I 18 due date violations (114046 s)

for the best fit algorithm against 5 (545446 s) and 6 (30684 s) of the two genetic algorithms variations, which also obtained a reduction of 6.3% in the energy consumption. In Profile ii we observed the same behavior, reducing from 19 (189892 s) to 12 (78471 s and 114845 s) due date violations with a reduction of 4.9% in the energy consumption. The same goes for Profile iii which reduced from 22 (169612 s) to 11 (118092 s and 118477 s) due date violations with an economy of 5% in energy. The values for the total time violated of the tasks may seem high but we need to consider that the scheduling is constrained by a power envelope, and in this case the tasks need to be delayed for the next moment with enough power available (if we consider only solar energy for instance, this may take a whole day).

In Fig. 5 we present the power produced and consumed for Profile i. These results were obtained when using the Best Fit Due Date (a) and MPGA-MO (b) scheduling planners with Profile i and 1029 tasks. We can observe that in some points (such as in the first 100 samples) the power consumption can be similar for both algorithms due to the high number of tasks that needs to be scheduled and so reaching the maximum power available. This justifies why we have different number of due date violations with the same workload under different power profiles: at some points we have too many tasks to be scheduled, and they lack flexibility (time between release and due date) to wait the next moment where enough power will be available (samples 100–200). This highlights the importance of the generation of multiple power envelopes when considering renewable energy sources and storage elements engagement. We could not only save energy but also provide a better QoS; this behavior can be observed by comparing the results obtained with Profile i against the two others, which have a higher number of violations and in case of Profile ii also a higher energy consumption.

The results become even more significant if we consider the long term impact that it could provide. For Profile i, displayed in Fig. 5, in a period of 2 days we could save 164.98 kWh using the MPGA-MO, instead of 155.35 kWh and 160.04 kWh for first fit and best fit due date respectively. This energy could be stored and used in the generation of the next scheduling windows improving the results, or sold to the grid power provider.

In Fig. 6 the average execution time of all the algorithms (with minimum and maximum values in the bars) is presented. Despite of the smaller number of due date violations and lower energy consumption, as expected, the Genetic Algorithm can have an execution time exponentially higher than the greedy ones. Nevertheless, if the scheduling requested is not a reactive action, this execution time is not prohibitive (around 12 min in the worst case for two days scheduling). We also highlight that it is possible to improve even more the execution time by improving the stopping criteria, but this will have an impact of the quality of the schedule.



(a) Due date violations of all profiles and algorithms.

(b) Total violated time of all profiles and algorithms.

(c) Energy consumption of all profiles and algorithms.

Fig. 4. Power available and consumed in the power profiles using best fit and genetic algorithm scheduling plan.

(a) Best Fit Due Date

(b) MPGA-MO

Fig. 5. Power available and consumed in the power PROFILE I considering two different algorithms and 1029 tasks.

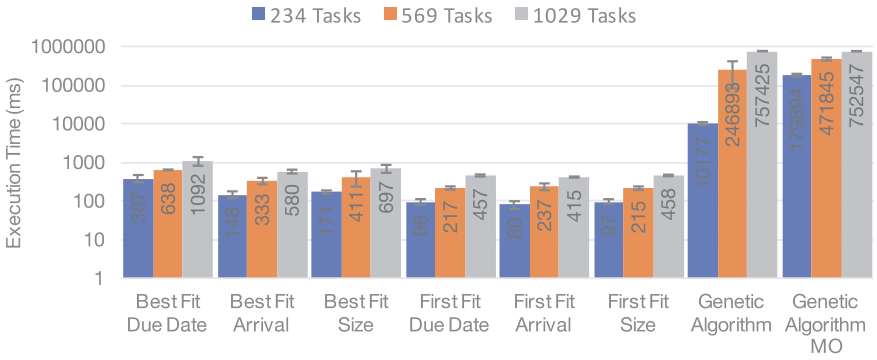


Fig. 6. Execution time of the different algorithms with different number of tasks with Profile i.

6 Conclusion

This article focused on presenting and evaluating an optimization module called RECO that aims to schedule tasks in a cloud datacenter while respecting the possible power envelopes.

We presented different algorithms that try to minimize due date violations while respecting power and resource constraints. The proposed genetic algorithm approach (MPGA and MPGA-MO) was able to reduce from 304 (First Fit) to 11 due date violations, in the best scenario, while also reducing the energy consumption from 78.7 kWh to 73.15 kWh respecting one of the power envelopes provided by a power manager. We have also presented an evaluation of the impact the power envelopes can have in the task scheduling, and concluded that more power does not necessarily means better QoS for the IT part, but it is more important to know when this power is delivered.

Finally, we intend to continue our research extending RECO to support real time task arrival, services (not only batch tasks), and variations in the amount of resources that are consumed by the applications. We also intend to connect RECO's generic interface through a message queue with an electrical middleware to receive the power envelopes.

Acknowledgments. The work presented in this paper was supported by the French ANR DATAZERO project ANR-15-CE25-0012. For source characterization, the experimental database has been obtained thanks to the financial support of several LAPLACE projects, France (leaders Christophe TURPIN, Eric BRU)

References

1. Khan, Z., Kiani, S.: A cloud-based architecture for citizen services in smart cities. In: 2012 IEEE Fifth International Conference on Utility and Cloud Computing (UCC), pp. 315–320, November 2012
2. Le, K., Bilgir, O., Bianchini, R., Martonosi, M., Nguyen, T.D.: Managing the cost, energy consumption, and carbon footprint of internet services. SIGMETRICS Perform. Eval. Rev. **38**(1), 357–358 (2010)

3. Koomey, J.: Growth in data center electricity use 2005 to 2010. In: A report by Analytical Press, completed at the request of The New York Times, p. 9 (2011)
4. Beldiceanu, N., et al.: Towards energy-proportional clouds partially powered by renewable energy. Computing **99**(1), 3–22 (2017)
5. Orgerie, A.-C., Assuncao, M.D.D., Lefevre, L.: A survey on techniques for improving the energy efficiency of large-scale distributed systems. ACM Comput. Surv. **46**(4), 1–31 (2014)
6. Borgetto, D., Stolf, P.: An energy efficient approach to virtual machines management in cloud computing. In: 2014 IEEE 3rd International Conference on Cloud Networking (CloudNet), pp. 229–235, October 2014
7. Deng, W., Liu, F., Jin, H., Li, B., Li, D.: Harnessing renewable energy in cloud datacenters: opportunities and challenges. IEEE Netw. **28**(1), 48–55 (2014)
8. Goiri, I., et al.: GreenSlot scheduling energy consumption in green datacenters. In: 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC), pp. 1–11, November 2011
9. Goiri, I., Le, K., Nguyen, T.D., Guitart, J., Torres, J., Bianchini, R.: GreenHadoop: leveraging green energy in data-processing frameworks. In: Proceedings of the 7th ACM European Conference on Computer Systems, EuroSys 2012, pp. 57–70. ACM, New York (2012)
10. Aksanli, B., Venkatesh, J., Zhang, L., Rosing, T.: Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. In: Proceedings of the 4th Workshop on Power-Aware Computing and Systems, HotPower 2011, pp. 5:1–5:5. ACM, New York (2011)
11. Liu, Z., Lin, M., Wierman, A., Low, S.H., Andrew, L.L.: Geographical load balancing with renewables. SIGMETRICS Perform. Eval. Rev. **39**(3), 62–66 (2011)
12. Sharma, N., Barker, S., Irwin, D., Shenoy, P.: Blink: managing server clusters on intermittent power. SIGARCH Comput. Archit. News **39**(1), 185–198 (2011)
13. Mudge, T.: Power: a first-class architectural design constraint. Computer **34**, 52–58 (2001)
14. Villebonnet, V., Costa, G.D., Lefevre, L., Pierson, J.M., Stolf, P.: Energy aware dynamic provisioning for heterogeneous data centers. In: 2016 28th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD), pp. 206–213, October 2016
15. Da Costa, G., Grange, L., Courchelle, I.D.: Modeling and generating large-scale Google-like workload. In: 2016 Seventh International Green and Sustainable Computing Conference (IGSC), pp. 1–7, November 2016