Assessing Power Needs to Run a Workload with Quality of Service on Green Datacenters

Louis-Claude Canon, Damien Landré, Laurent Philippe, Jean-Marc Pierson, Paul Renaud-Goud

Assessing power needs to run a workload with quality of service constraint on green datacenters

Damien Landré Louis-Claude Canon Laurent Philippe Jean-Marc Pierson
Paul Renaud-Goud
Université de Franche-Comté, CNRS, institut FEMTO-ST, F-25000 Besançon, France
email: {lccanon,damien.landre,lphilipp}@femto-st.fr
IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, Toulouse, France
email: {Damien.Landre,jean-marc.pierson,paul.renaud.goud}@irit.fr

Abstract

Datacenters are an essential part of the internet, but their continuous development requires finding sustainable solutions to limit their impact on climate change. The Datazero2 project aims to design datacenters running solely on local renewable energy. In this paper, we tackle the problem of computing the minimum power demand to process a workload under quality of service constraint in a green datacenter. To solve this problem, we propose a binary search algorithm that requires the computation of machine configurations with maximum computing power. When machines are heterogeneous, we face the problem of choosing the machines and their DVFS state. A MILP (Mixed-Integer Linear Programming), to find the optimal solution, and four heuristics that give satisfactory results in a reasonable time are proposed. The bests reach an average deviation from the optimal solution of 0.03% to 0.65%.

Keywords

Green datacenter, Power consumption, Optimization

1 Introduction

Since a decade datacenters have become an essential part of the internet, either being at the edge or at the center, and their number and size are continuously increasing, as their global energy consumption. These datacenters represented in 2018 1% of the global energy consumption, that is to say 6% more than in 2010 [14]. It is estimated [4] that, by 2025, energy consumption will have multiplied by 2.9 and greenhouse gas emissions by 3.1. To reduce the datacenter impact on climate change several research works propose solutions to optimize their energy consumption [6], [18]. These solutions are essential on the way to efficiency, but cannot achieve a drastic reduction of the carbon footprint. Other projects and research works claim to reduce their brown energy consumption [10], [2]. The objective of the Datazero project [21] (2015-2019) and Datazero2 (2020-2024) is to investigate the solutions to design and operate a datacenter only fueled by renewable energies. By design, this project builds on a negotiation [24] between the electrical management (power production and storage) and the IT management (workload processing) to choose a power command that will be applied in the next time window, typically the coming 72 hours. A power command refers to the control commands that are asked to the electrical side, so that the needed power is provided along the time window.

The negotiation is developed as a process based on game theory that loops on asking for IT consumption and power production predictions over the time window to converge after several iterations on an acceptable solution for both the IT and electrical management. The result of the negotiation is a power command for the time window, represented as a time series of power values for each time interval, called a power profile. This power command is then applied on the infrastructure. The negotiation therefore needs predictions of the power needs during the time window. In this article, we tackle the problem of computing the minimum power profile required to process a workload forecast. We do not address the problem of workload forecast that has been widely studied already [15]. Rather, we investigate the problem of transforming a workload prediction to an optimized usage of a given infrastructure, called a plan,

that minimizes the electrical power needs. This plan is made for how the machine will be switched-on and off based on the negotiated power profile. It is sent to an online module, which applies it and adapts it to events, due to the uncertainty of the plan [8]. Since the negotiation process is interactive, this computation must last a reasonable time [24]. It must be noted already that we consider a consolidated workload, and not individual jobs. Therefore, a workload represents the total amount of work units that have to be processed along time. A workload possibly aggregates the work units of several jobs that may concurrently use the infrastructure and share each of the machines of the infrastructure.

This paper contributes with multiple variants of an algorithm that computes a minimized power profile. The algorithm realizes this computation in steps. The main step iterates on each time interval of the time window. For each time interval, a binary search algorithm is used to find a minimized power value. Last, for each power value, the algorithm computes the maximum processing capacity that can be reached using the datacenter machines and tries to schedule the workload under quality of service (QoS) constraints, using the processing capacity. We propose several solutions, a MILP and different heuristics, to compute the maximum computing capacity for a power value.

In the following, related work is detailed in Section 2. Section 3 formally defines the problem of computing a capping value from a workload forecast and maximizing a processing capacity within a given power value. Section 4 and Section 5 respectively present the binary search algorithm and the heuristics proposed to solve the problem. Section 6 presents experiments and results. Finally, Section 7 summarizes the paper, highlighting the main conclusion.

2 Related Work

In order to minimize energy or power consumption while possibly meeting another criteria, different online approaches are considered in the literature. In [27], an online method manages energy consumption and workload scheduling for hybrid geo-distributed datacenters. Several variables are taken into account such as the variation of the electricity price, the power consumption of the cooling, machines, or constraints on renewable energy. In a similar way, in [20] a genetic based online algorithm optimizes energy consumption, performance degradation as well as power grid cost for multiple hybrid datacenters. Zhang et al. [26] propose PoDD, an online power-capping algorithm to maximize performances of homogeneous servers for dependent application workloads. The applications are composed of a front-end and a back-end, and the front-end produces the data consumed by the back-end. A similar method is also proposed for heterogeneous nodes of a datacenter [7]. In [25], different online algorithms are introduced to minimize the performance degradation and the total energy consumed: LmsReg, based on regression, which detects overloaded servers and migrates virtual machines, and MuP, which addresses the trade-off between power consumption, number of migrations, server performance and the total number of servers that have been switched-off in the selection of virtual machines. These algorithms migrate virtual machines from over-loaded servers to under-loaded servers. Other methods using virtual machine allocation and migration are proposed [16], [12], [17]. In [13], an online holistic approach schedules virtual machines to minimize the total energy consumption of the datacenter by considering the datacenter as a whole and not trying to divide it into several parts to be treated separately from each other. But these works focus only on the objective of reducing online energy consumption by allocating and migrating virtual machines. In our case, we seek to minimize a forecast of power demand.

In [17], an online multi-objective algorithm optimizes energy consumption, by taking into account QoS, energy, number of active servers and number of virtual machine migrations on servers. A similar method in [9] is used by considering DVFS (Dynamic Voltage and Frequency Scaling), temperature-dependent task scheduling, dynamic resource provisioning, and cooling management. But the optimization is also done online, with a non-clairvoyant approach.

In [11], in the context of cloud datacenters, an online clairvoyant method for predicting the total energy consumption of the datacenter is proposed to improve the datacenter energy management that controls and coordinates all the equipment. This method evaluates the importance of the variables of the equipment to make the prediction. Then a neural network computes the prediction on the total energy consumption, a single value for the coming 20 minutes. Finally, an online module is in charge of updating the model based on the forecast errors.

In [8], the authors develop a complementary approach to ours. An online module is based on offline decisions, adapting them to real-time events via different compensation policies, to stay as close to

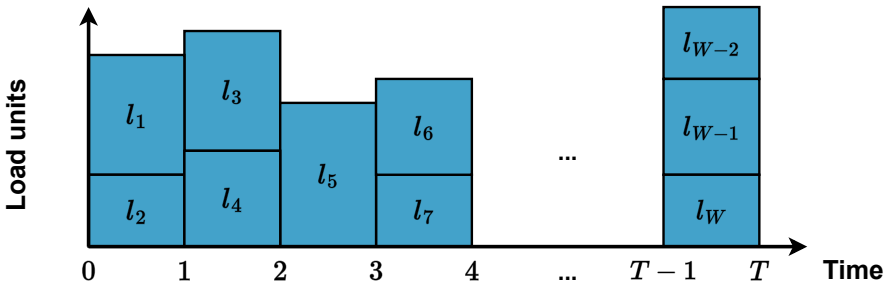


Figure 1: Load parts and amount of operations of a workload forecast.

the offline plan. The module is responsible for compensating energy utilization, scheduling and servers in a green datacenter. Different compensation policies are evaluated according to five metrics. The results indicate that compensation are necessary and simply following the plan is not enough, due to the uncertainty of the plan.

Finally, all these studies address the online problem and, to the best of our knowledge, there is no work addressing the offline minimization of power consumption in the case of homogeneous or heterogeneous machines with different amount of work to process, under the constraint of deadline violations.

3 Problem definition, model and objective

The power management system, running solely on renewable energy, needs to plan its power sources and storage usage to correctly fuel the machines. Due to technical constraints [21], the power command delivered to the machines must be constant over a minimum time duration, a time interval, that ranges from 15 minutes to one hour. To give the power usage over a time window, we thus need to define a power profile that gives, for each time interval, a constant power need value. Since the power supply only relies on intermittent renewable energies, saving as much energy as possible, and storing it, is essential to operate the datacenter during periods of underproduction. The problem we face is hence to compute the minimum power profile needed to process a given workload.

On the other hand, the time intervals are independent. For this reason, in the following we concentrate on finding the minimal value P for one interval, then the same method is applied to each time interval of the time window. Note that the problem is tackled as an offline problem, but it is also constrained by the interactivity of the negotiation, which requires several exchanges before converging to a solution. The problem must hence be resolved in a reasonable time. In the following, we first define the problem and its associated model, then we define the objectives.

In the context of this paper, the workload forecast is an input of the problem and the solution must be able to handle any workload, whatever its characteristics. Since the temporalities are different between the power and the load variations, a time interval is subdivided into multiple time steps and the workload gives the variation of the load on time steps, which duration typically ranges from 1 second to one minute. Formally, we denote by T the number of time steps, and we normalize the time axis such that the t^{th} time step begins at time $t-1$, for $t \in \mathcal{T} = \{1, \ldots, T\}$. We define Δt as the duration of a time step in seconds.

The workload is composed of several load parts, each arriving at a given time step. We define the total workload as a set of W load parts, l_k for $k \in \mathcal{W} = \{1, \ldots, W\}$. A load part l_k is defined by its release time r_k (*i.e.* the time step when l_k arrives), its amount of operations to be processed p_k and a deadline d_k such that, if the load part cannot be processed before d_k, it is killed. For instance, on the first time step of Fig. 1, the workload is composed of two load parts, l_1 and l_2. p_1, the amount of operations of l_1, is two times larger than p_2. The deadline d_k is defined as a duration. It enforces that load parts arriving in the same time step may have different QoS constraints. A load part l_k, which arrives at the time step $t = r_k$ with a deadline d_k must be finished no later than $r_k + d_k$. All the operations of a load part l_k have the same deadline d_k. For instance, in Fig. 1 if load part l_1 has a deadline $d_1 = 2$ steps, then it can be processed partially or completely during time steps 1 and/or 2.

The datacenter is composed of M machines which are noted machine i with $i \in \mathcal{M} = \{1, \ldots, M\}$.

Considering the power consumption, machine i dissipates a power $static_i$ when it is idle. Each machine i can be set in S_i different DVFS states [9]. The DVFS state of a machine is noted $j \in \mathcal{S}^{(i)} = \{0, \ldots, S_i\}$. The set of machines with their DVFS states defines the configuration of the datacenter. We note $\mathcal{S}$ the set of DVFS states of all machines, $\mathcal{S} = \{\mathcal{S}^{(1)}, \ldots, \mathcal{S}^{(M)}\}$. A DVFS state j defines $g^{(i)}_{\max_j}$, the maximum amount of operations per second that machine i can process, and $power^{(i)}_j$, the consumed power per operation per second. For the sake of simplicity, we consider an average value for this consumed power. The model could be extended to consider different power consumption for different operations, since consumption can vary depending whether the workload is CPU, I/O, memory or network intensive. If machine i is switched-on, it computes $g^{(i)}$ operations per second with $0 \leq g^{(i)} \leq g^{(i)}_{\max_j}$ while dissipating $power^{(i)}_j$ power per operation per second. Therefore, if a machine i computes several load parts l_k in state j during a time Δt with an amount of operations to be processed $g^{(i)}\Delta t = \sum_{k \in \mathcal{W}} p_k \leq g^{(i)}_{\max_j}\Delta t$, it consumes a power of $P_i = static_i + g^{(i)} \times power^{(i)}_j$. We assume that, when a machine is off, its DVFS state is $j = 0$ and it does not consume any power, nor does it process any operation. Last, the power consumption P of a configuration is $P = \sum_{i \in \mathcal{M}} P_i$ and its maximum available computing power $w^{(p)}$ is $w^{(p)} = \sum_{i \in \mathcal{M}} g^{(i)}\Delta t$.

The objective of this optimization problem is thus to find a machine configuration (off, on and other DVFS states) that delivers enough computing power to process the workload while consuming the lowest power P. In addition, to give the users a flowtime guarantee, we define opk as the total amount of operations missing their deadline and the ratio D as the amount of operations killed over the amount of operations to be processed during the time interval (1). This ratio must not exceed a fixed threshold $D_{\max}$ (2) to meet the flowtime guarantee.

$$D = \frac{opk}{\sum_{k \in \mathcal{W}} p_k} \qquad (1) \qquad\qquad D \leq D_{\max} \qquad (2)$$

4 Determining the minimum power value

To address the problem of minimizing the power value for a time interval under the deadline violation constraint we propose a binary search algorithm (Algorithm 1). For a given power value P, the algorithm first computes a machine configuration that maximises the computing power and then schedules the workload to determine the deadline violation ratio D of the time interval.

Algorithm 1: Binary search algorithm to minimize the power need to run a workload on a configuration violation ratio

Data: $\mathcal{M}, \mathcal{S}, \mathcal{W}, \mathcal{T}, \epsilon, D_{\max}, \Delta t$
Result: minimize P

```
1  begin
2      P_min ← 0; P_max ← Σ_{i∈M} (static_i + power^(i)_{S_i} × g^(i)_{max_{S_i}})
3      while P_max − P_min ≥ ε do
4          P ← (P_min + P_max)/2
5          w^(p) ← config(M, S, P, Δt)
6          opk ← 0; W̄ ← W
7          for t ∈ T do
8              w̄^(p) ← w^(p)
9              W̄, opk ← schedule(W̄, opk, w̄^(p), t)
10         D ← opk / Σ_{k∈W} p_k
11         if D ≤ D_max then P_max ← P else P_min ← P
```

The dichotomy is initiated (line 1) by setting the maximum power $P_{\max}$ to the case where all the machines are used to their maximum capacity and the minimum power $P_{\min}$ to 0. Then the algorithm iterates until the difference between the two power values is lower than ϵ, the stopping criterion of the

algorithm. At each iteration the algorithm computes, with the *config* function [22] (line 5), the machine configuration with the largest possible computing power $w^{(p)}$ for the power value P of the current iteration. The *schedule* function is then used for each time step t of the time interval $\mathcal{T}$ (lines 7 to 9) to determine the schedule and the *opk* value, the number of killed operations. The *schedule* function simply uses EDF (Earliest Deadline First) algorithm to schedule the load parts on the time steps. Then, if the ratio of violated deadlines D does not exceed the threshold $D_{\max}$ (line 11), it means that the computing power is sufficient and the power value $P_{\max}$ can be decreased to P. Otherwise, D exceeds $D_{\max}$ and the power $P_{\min}$ must be increased to P.

Several propositions of the *config* function are given in the following section.

5 Maximizing the computing power

The *config* function computes the most powerful machine configuration that can be fueled with the power P, fixed by the binary search algorithm. This computation obviously depends on the machine characteristics. The simplest case is when machines are homogeneous with only two DVFS states (switch-on or -off) since machines are undifferentiated and it is sufficient to calculate how many machines can be powered with P to provide the most powerful configuration. If homogeneous machines have several DVFS states there is already a decision to take between switching-on a new machine or setting an already started machine in a higher DVFS state. In the heterogeneous case, several configurations are possible for a given power value, but all of them do not provide the same computing power. It is therefore important to improve the power efficiently by determining an optimal machine configuration. The difficulty lies in the choice of the machines to be switched-on and their DVFS state. In the following, we concentrate on the case of heterogeneous machines with multiple DVFS states, which includes the homogeneous case.

From the complexity view point, the problem of computing the maximum computing power $w^{(p)}$ with heterogeneous machines is at least as difficult as the partition problem and is thus NP-Hard. The corresponding decision problem is trivially in NP as we can verify a solution from the decision variables $g^{(i)}$ in polynomial time. Besides, any instance of the partition problem can be directly reduced to our problem: for each integer z_i, we consider a machine such that $static_i = 0$ and $g^{(i)}$ has only two possible values, *i.e.* 0 or $g^{(i)}_{\max_j} = z_i$. Note that, in the general case, the $g^{(i)}$ are coded in a discrete variable that ranges from 0 to $g^{(i)}_{\max_j}$. In this particular case, we just give the lowest possible value to $g^{(i)}_{\max_j}$. In that case, the power consumed by a computing machine becomes constant, $P_i = static_i + g^{(i)}_{\max_j} = z_i$. Furthermore, we set the total power available $P = \frac{1}{2}\sum_i z_i$. There is a schedule with maximum computing power $w^{(p)} = P$ if and only if the partition problem has a valid solution. Since this problem is NP-Hard, we first designed a MILP (Mixed-Integer Linear Programming).

Mixed Integer Linear Programming: We define the decision variable $x_{i,j}$ to determine the machines to be switched-on or -off and their DVFS state. For each machine i and for each DVFS state j, $x_{i,j} = 1$ if the DVFS state j of machine i is selected, otherwise $x_{i,j} = 0$. These variables hence define the machine configuration. For a machine, we consider that only one DVFS state can be selected and remains the same for the entire duration of the time interval.

The MILP is described in (3). The objective function is to maximize the computing power of the machines. Using the binary decision variable $x_{i,j}$, the first constraint states that a machine i, for all $i \in \mathcal{M}$, must have a single DVFS state j among all possible DVFS states of the machine, from 0 to S_i (including the switched-off state $j = 0$). Depending on the selected DVFS state we express, for all $i \in \mathcal{M}$, that the computing power must not exceed the maximum computing capacity of the machine, with the second constraint. Then, knowing the DVFS state and the computing power of the machine, the third constraint bounds the power consumption of the machine, for all $i \in \mathcal{M}$. Finally, the fourth constraint imposes that the total power consumption of the machines must not exceed the power P value given by the binary search algorithm.

$$\begin{cases} & \text{maximize } \sum_{i=1}^{M} g^{(i)}, s.t.: \\ \sum_{j=0}^{S_i} x_{i,j} & = & 1 \\ g^{(i)} & \leq & \sum_{j=0}^{S_i} (x_{i,j} \times g^{(i)}_{\max_j}) \\ P_i & = & \sum_{j=1}^{S_i} x_{i,j}(static_i + g^{(i)} power^{(i)}_j) \\ \sum_{i=1}^{M} P_i & \leq & P \end{cases} \quad (3)$$

Under the following constraints.

$$\begin{cases} \forall i \in \mathcal{M}, \forall j \in \mathcal{S}^{(i)} & x_{i,j} \in \{0,1\} \\ \forall i \in \mathcal{M} & g^{(i)} \geq 0 \\ \forall i \in \mathcal{M} & P_i \geq 0 \end{cases}$$

As shown by the experiments in section 6, the MILP calculation takes 2.83 seconds in average and up to 50 seconds in complex cases. This calculation has to be repeated for each iteration, usually more than 15, of the binary search algorithm. Its runtime, including the configuration computation and the scheduling, then varies from 42 seconds to more than 100 seconds, depending on the stopping criterion ϵ. Hence, the total runtime ranges from 50 minutes to more than 2 hours to determine a power profile. As previously explained the power profile is used in the negotiation process to anticipate the power which makes several iterations before taking a decision. Although based on offline calculations, the MILP is therefore used in an interactive process for which waiting one hour for a proposition does not make sense. For this reason, we propose in the following heuristics providing solutions in a shorter time.

Random Choice heuristic: A first trivial heuristic proposal is to randomly choose the type of machine to switch-on. When a machine is switched-on, it is allocated the power needed to provide the maximum computing power. The DVFS state chosen is the one maximizing the computing power, according to the remaining power. This step is repeated until the power is insufficient and/or there are no more machines to switch-on. The advantage of this heuristic is its fast execution time, but it provides unsatisfactory results compared to the other presented in the following in the heterogeneous case.

Balance Power-Performance (BPP) heuristic: The BPP heuristic evaluates the most suitable machines and its DVFS states to switch-on according to two metrics: (i) computing power and (ii) performance ratio. The computing power criteria is chosen since the objective is to maximize the total computing power of the machines $w^{(p)}$. The performance ratio criteria is chosen to minimize the power consumed per unit of computing power.

These two metrics are used to compute a normalized score depending on a given power and an α parameter. The α parameter (with $0 \leq \alpha \leq 1$), given as input, controls the trade-off between computing power and performance ratio. The nearer alpha is to 0, the more weight is given to the computing power in the choice of the machine to be switched-on, and inversely. Depending on the alpha parameter value, the configurations proposed by BPP can be different. For this reason, several alpha values are assessed in order to produce different machine configurations and the algorithm returns the one maximizing the total computing power. BPP switches-on the machine with the highest score.

This heuristic is very efficient in the homogeneous and heterogeneous case with a very satisfactory execution time.

Best State Redistribute Without Static (BSRWS) heuristic: The BSRWS heuristic focuses on the performance ratio of the machines. This heuristic switches-on as many machines with the best performance ratio as it is possible without exceeding the given power. If no more machine can be switched-on and there is power left, either because all the machines are on or because there is not enough power to switch-on more machines, the remaining power is redistributed to the switched-on machines. This redistribution increases the DVFS state of the switched-on machines and thus their computing power.

The advantage of this heuristic is its accuracy with a satisfactory execution time, which however increases with power. In the heterogeneous case, some solutions deviate from the optimal because it switches-on too many machines.

Best State Redistribute Without Static And Removing (BSRWS-AR) heuristic: The BSRWS-AR heuristic focuses on the performance ratios of the machines and explore more machine configurations. This heuristic is an improvement of the BSRWS heuristic. The latter is run several times and, at each

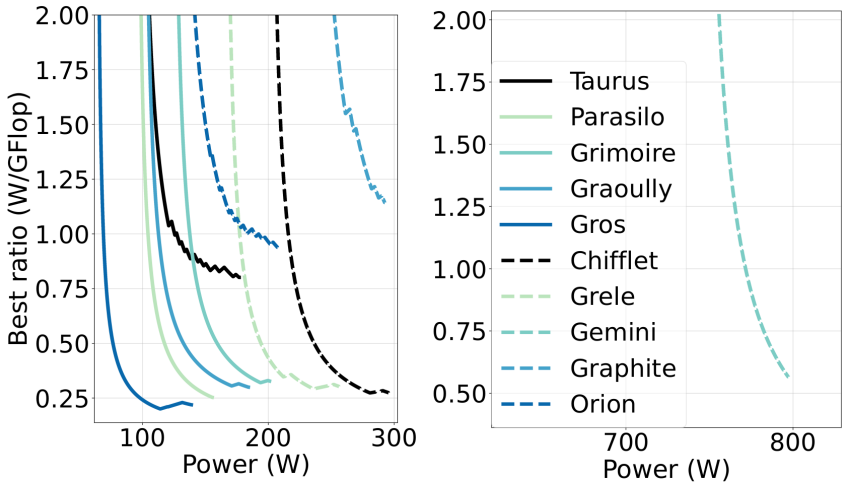


Figure 2: Best performance ratio in W/GFlop depending on power.

iteration, it removes a machine of the configuration in order to test configurations with fewer switched-on machines. More power can be redistributed to increase the DVFS state and the computing power of the remaining machines.

The advantage of this heuristic is its accuracy compared to BSRWS in the homogeneous and heterogeneous case. However, its execution time is much higher and increases strongly with power.

6 Experiment and Results

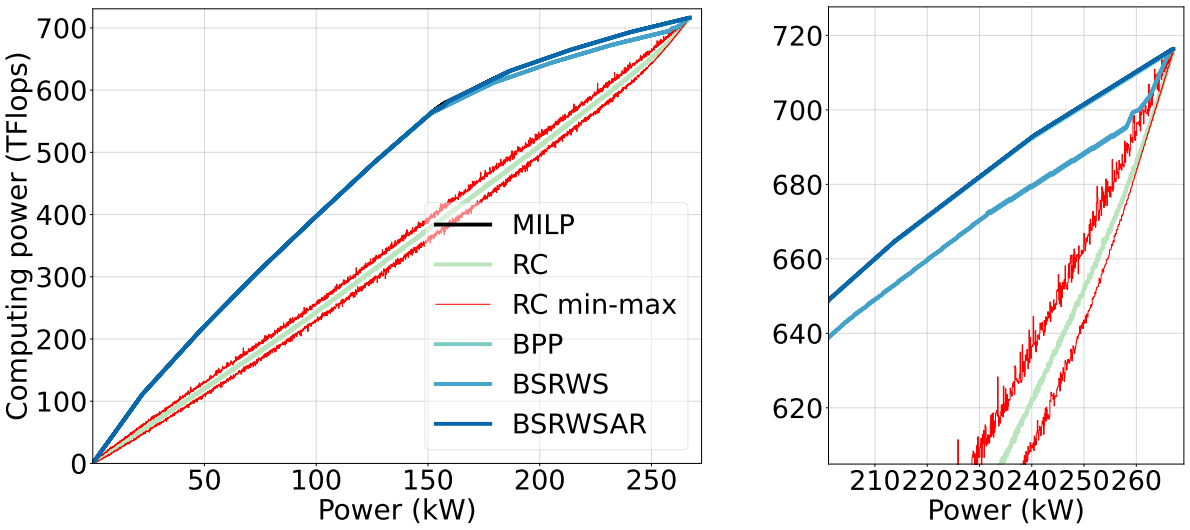
We present here an experiment that considers the example of a medium-sized datacenter of 267 kW [21] and 10 machine types. Note that, due to the paper length constraint, we only present this example but different size of datacenter are experimented and given in the research report [22] to completely assess our heuristics. The machine types are taken from the GRID5000 platform [3]. We have implemented the MILP and the heuristics in Python [1] [2]. We remind the reader that on this part there is no related work in the literature to compare to. Input data includes 1241 machines divided into the 10 types. Note that no workload is used for the experiment, as we only assess the quality of the solutions of the heuristics compared to the MILP in the determination of the computing power. Data on the characteristics of the machines are known in advance [19], [23]. For each type of machine, we know all the data described in section 3. All data are average values coming from experiments on Grid5000 performed in the context of the ANR ENERGUMEN [1] project.

To illustrate the characteristics of the set of machines, Fig. 2 shows the best performance ratio of the 10 machine types in W/GFlop depending on power, taking into account static power of the machines (the leftmost of each curve) and their DVFS states. It is worth noticing the continuity of the performance ratio curves, even when changing the DVFS state, for almost all machines. The Gemini machines, having the highest static power, are shown on the right figure. The other machines are grouped on the left. The lower the static power of a machine and the better the performance ratio, the more advantageous it is to switch-on this machine, depending on power. For instance, on a simple case, if the available power is 1000 W, the best is to use 9 Gros machines than any other combination (visually and confirmed by the MILP solution), if we have these 9 Gros machines at hand. Otherwise, with less Gros machines, a different combination has to be used involving Gros and other types of machines. Also note that some performance ratios of machine types cross others.

Fig. 3 shows the maximum computing power given by the MILP and the heuristics for different power values, from 63 W (the minimum power required to switch-on a machine) to 267 kW (the maximum power that can be required by all machines when running at maximum frequencies) by steps of 100 W.

[1] The source code in zip file is available here

[2] Experiments have been run on Ubuntu 22.04.1 LTS, Intel Core i7-11850H processor, 32.0 Go of memory, Python 3.10 and PulP 2.6.0 with Gurobi 9.5.1 solver.



(a) (b)

Figure 3: (a): Comparison of the maximum computing power computed by the MILP and heuristics depending on power value from 63 W to 267 kW by steps of 100 W. (b): Zoom on 200 kW to 267 kW by steps of 100 W.

Table 1: Average and median relative deviation in percentage of heuristics from optimal solution.

	MILP	RC	BPP	BSRWS	BSRWS-AR
Avg. dev. (%)	-	31.84	0.12	0.65	**0.03**
Median dev. (%)	-	34.57	0.04	0.09	**0.00**
Avg. Exec. Time (s)	2.83	1.15×10^{-3}	9.07×10^{-3}	$\mathbf{1.03 \times 10^{-3}}$	1.61

The RC (Random Choice) heuristic is run 100 times to show the dispersion of the solutions and the average computing power for each power value. The minimum and maximum computing power are shown in red.

The RC heuristic significantly deviates from the optimal solution with an average deviation of 31.84% (Table 1). This is not surprising. Since the choice of the machine type is random, the RC heuristic may switch-on the least efficient machines. This occurs mostly in the heterogeneous case, according to our experiences. The BPP (Balance Power-Performance) heuristic with alpha from 0 to 1 by steps of 0.05 for each power value and BSRWS-AR heuristic (Best State Redistribute Without Static And Removing) are the closest to the optimal solution. Their average deviation from the optimal is 0.12% and 0.03% respectively (Table 1). Note that the BSRWS-AR heuristic performs better than the BSRWS (Best State Redistribute Without Static) heuristic, since it explores more configurations. Also, BPP outperforms BSRWS and BSRWS-AR in other cases of heterogeneity. From 150 kW to 260 kW, the deviation from the optimal is more significant for BSRWS (Fig. 3). But our different experiences show that this is not always the case. The BSRWS heuristic has an average deviation of 0.65%. Note that from 260 kW, the BSRWS heuristic reduces its deviation from the optimal because there is enough power to switch-on all the machines and redistribute the remaining power to increase their DVFS state. In terms of accuracy, BPP and BSRWS-AR are the most satisfying heuristics, but BPP is significantly faster than BSRWS-AR. This is mostly the case in our experiments.

Fig. 4 gives the runtimes of the MILP and the heuristics depending on power. Note that the y-axis is plotted on a logarithmic scale. There is a general trend for all the runtimes to increase with power. This is intuitive since the more power, the more machines the heuristics have to consider. Compared to the MILP that has an average runtime of 2.83 s per power value, the heuristics are more time efficient to find a configuration. The runtime of the BPP heuristic is of the order of milliseconds and increases slightly depending on power. While the BSRWS-AR heuristic runtime increases dramatically: from 0.4 ms to more than 4 s depending on power. This is partly due to the fact that the more machines are switched-

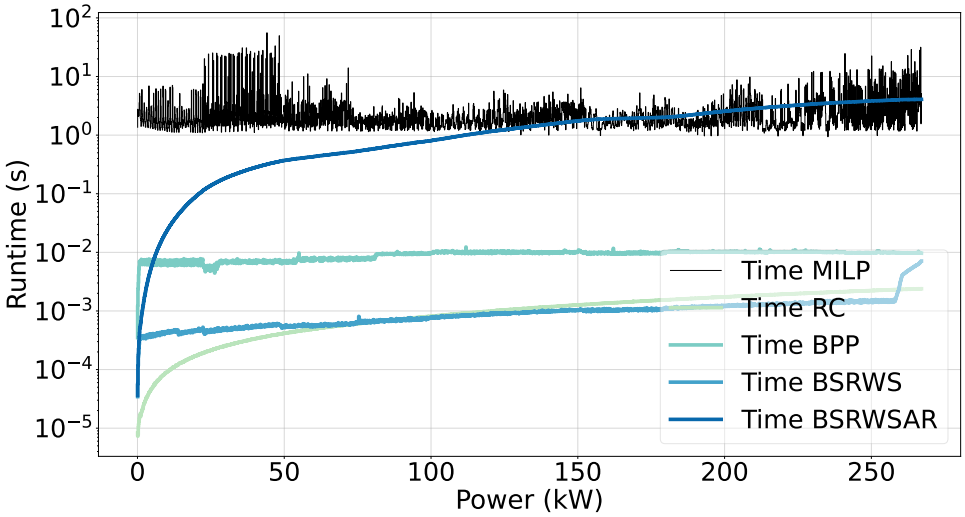


Figure 4: Runtime of the MILP and the heuristics depending on power

on, the more configurations are explored. Note that the use of the *redistribute* function in BSRWS heuristic when all the machines are switched-on explains the increase of the runtime when the power is approximately 260 kW. When determining the power required over a time interval, the runtime of the binary search algorithm varies from 0.11 seconds to 2.75 seconds, using the BPP heuristic and depending on the stopping criterion ϵ.

7 Conclusion

In this paper, we tackle the problem of minimizing a power value to switch-on just enough machines to process a workload over a time interval while respecting quality of service constraints. We propose a binary search algorithm to solve this problem with multiple variants. This algorithm uses two functions, one that computes the maximum computing power that is obtained knowing a given power, and another that schedules the workload on the switched-on machines. Since computing the maximum processing power is NP-Hard in the heterogeneous case, we propose a MILP and 3 non-trivial heuristics and compare their performance and runtime. Heuristics give satisfactory results in a reasonable time, with an average relative deviation from optimal solution of 0.12%, 0.65% and 0.03%. Looking at the results and runtime, the BPP (Balance Power-Performance) heuristic seems the most suitable to solve this problem in a reasonable time.

These different approaches show that using DVFS states in a heterogeneous environment allows approaching the optimal configuration of the machines and thus efficiently using energy. In future work, we plan to compare the BS approach to an integrated solution, then we will take the switching-on and switching-off and their consumption into consideration to integrate the cost of changing a configuration between two time intervals. Last, we plan to introduce uncertainty in the machine choice to better anticipate the workload variations.

Acknowledgments and Data Availability Statement

This work was supported by the DATAZERO2 project (“ANR-19-CE25-0016”) and by the EIPHI Graduate school (“ANR-17-EURE-0002”). We would like to thank the ENERGUMEN project(“ANR-18-CE25-0008”) and Grid5000 for providing some of the data used in this article. The scripts and instructions necessary to reproduce and analyze our result are available in a Figshare repository [5].

References

[1] Energy saving in large scale distributed platforms – Energumen. https://anr.fr/Project-ANR-18-CE25-0008, 2018. [Online; accessed 20-November-2022].

[2] Apple: Environmental Progress Report (2022). https://www.apple.com/environment/, 2022. [Online; accessed 20-May-2022].

[3] Clusters Grid5000. https://www.grid5000.fr/, 2022. [Online; accessed 21-October-2022].

[4] F. Bordage. The environmental footprint of the digital world. *GreenIT*, 2019.

[5] L.-C. Canon, D. Landré, L. Philippe, J.-M. Pierson, and P. Renaud-Goud. Artifact and instructions to generate experimental results for euro-par'2023 paper: Assessing power needs to run a workload with quality of service on green datacenters. https://doi.org/10.6084/m9.figshare.23544939, 2023.

[6] P. Chaithra. Eco friendly green cloud computing. *Journal of Research Proceedings*, 1(2):41–52, 2021.

[7] T. Ciesielczyk, A. Cabrera, A. Oleksiak, W. Piątek, G. Waligóra, F. Almeida, and V. Blanco. An approach to reduce energy consumption and performance losses on heterogeneous servers using power capping. *Journal of Scheduling*, 24, 2021.

[8] I. Fontana de Nardin, P. Stolf, and S. Caux. Mixing offline and online electrical decisions in data centers powered by renewable sources. In *IECON–48th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE, 2022.

[9] Q. Fang, J. Wang, Q. Gong, and M. Song. Thermal-aware energy management of an hpc data center via two-time-scale control. *IEEE Transactions on Industrial Informatics*, 13(5):2260–2269, 2017.

[10] Greendatanet research project. http://www.greendatanet-project.eu/, 2013–2016. Accessed: 2021-05-28.

[11] Y.F. Hsu, K. Matsuda, and M. Matsuoka. Self-aware workload forecasting in data center power prediction. In *2018 18th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 321–330. IEEE, 2018.

[12] H.A. Kurdi, S.M. Alismail, and M.M. Hassan. Lace: A locust-inspired scheduling algorithm to reduce energy consumption in cloud datacenters. *IEEE Access*, 6:35435–35448, 2018.

[13] X. Li, P. Garraghan, X. Jiang, Z. Wu, and J. Xu. Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy. *IEEE Transactions on Parallel and Distributed Systems*, 29(6):1317–1331, 2017.

[14] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey. Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986, 2020.

[15] M. Masdari and A. Khoshnevis. A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing*, 23(4):2399–2424, 2020.

[16] S. Mazumdar and M. Pranzo. Power efficient server consolidation for cloud data center. *Future Generation Computer Systems*, 70:4–16, 2017.

[17] B. Nikzad, B. Barzegar, and H. Motameni. Sla-aware and energy-efficient virtual machine placement and consolidation in heterogeneous dvfs enabled cloud datacenter. *IEEE Access*, 10:81787–81804, 2022.

[18] A. Pahlevan, M. Rossi, P. Garcia del Valle, D. Brunelli, and D. Atienza Alonso. Joint computing and electric systems optimization for green datacenters. Technical report, Springer, 2017.

[19] K. Pedretti, R.E. Grant, J.H. Laros III, M. Levenhagen, S.L. Olivier, L. Ward, and A.J. Younge. A comparison of power management mechanisms: P-states vs. node-level power cap control. In *Int. Parallel and Distributed Processing Symposium Workshops*. IEEE, 2018.

[20] Y. Peng, D.K. Kang, F. Al-Hazemi, and C.H. Youn. Energy and qos aware resource allocation for heterogeneous sustainable cloud datacenters. *Optical Switching and Networking*, 23:225–240, 2017.

[21] J.M. Pierson, G. Baudic, S. Caux, B. Celik, G. Da Costa, L. Grange, M. Haddad, J. Lecuivre, J.M. Nicod, L. Philippe, V. Rehn-Sonigo, R. Roche, G. Rostirolla, A. Sayah, P. Stolf, M.T. Thi, and C. Varnier. Datazero: Datacenter with zero emission and robust management using renewable energy. *IEEE Access*, 7, 2019.

[22] Research report: Assessing power needs to run a workload with quality of service constraint on green datacenters. `https://members.femto-st.fr/Laurent-Philippe/sites/femto-st.fr.Laurent-Philippe/files/content/articles/rr-1-2023.pdf`, 2023.

[23] Standard performance evaluation corporation. `http://spec.org/`. Accessed: 2023-02-28.

[24] M.T. Thi, J.M. Pierson, G. da Costa, P. Stolf, J.M. Nicod, G. Rostirolla, and M. Haddad. Negotiation Game for Joint IT and Energy Management in Green Datacenters. *Future Generation Computer Systems*, 110:1116–1138, 2020.

[25] R. Yadav, W. Zhang, K. Li, C. Liu, M. Shafiq, and N.K. Karn. An adaptive heuristic for managing energy consumption and overloaded hosts in a cloud data center. *Wireless Networks*, 26:1905–1919, 2020.

[26] H. Zhang and H. Hoffmann. Podd: Power-capping dependent distributed applications. In *Proceedings of the Int. Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–23, 2019.

[27] M. Zhao, X. Wang, and J. Mo. Workload and energy management of geo-distributed datacenters considering demand response programs. *Sustainable Energy Technologies and Assessments*, 55:102851, 2023.