Sufficiency power consideration to run a workload on renewable energy operated datacenter

Damien Landré[1,2], Laurent Philippe[1], Jean-Marc Pierson[2]

Damien.Landre@irit.fr, laurent.philippe@femto-st.fr, Jean-Marc.Pierson@irit.fr
[1] Université Marie et Louis Pasteur, CNRS, institut FEMTO-ST, F-25000 Besancon, France
[2] IRIT, 118 Route de Narbonne, 31062 Toulouse, France

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 datacenter operated with renewable energy. 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 (Dynamic Voltage and Frequency Scaling) 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. simulations show that the best heuristics reach an average deviation from the optimal solution of 0.03% to 0.65%. The binary search algorithm is challenged against a real workload to assess the impact of flexibility on the quality of service.

Keyword: Green datacenter, Power consumption, Optimization

1 Introduction

Datacenters are one of the most power consuming part of the ICT and their workload has been multiplied by 10 [1] in the past decade. Despite huge efforts to limit their potential energy consumption explosion (improving the efficiency of electronic components, of cooling, of the power leakage), their consumption still represents 1 to 1.5% of the global energy consumption. Even if it has (only) increased by 6% since 2010 [2], there is no evidence that these limitation efforts will be able to absorb the continuing growth in internet use, since efficiency improvements and load migration from traditional, energy consuming, datacenters to hyper-scale datacenters will reach a limit. At the same time, the datacenter industry is under pressure to limit its impact on climate change, in particular its greenhouse gas emissions. To face these challenges, datacenter operators invest in REC (Renewable Energy Certificates) that compensate their carbon footprint but do not guarantee that their facilities are supplied with green power. Several datacenter operators have also announced plans to integrate low-carbon electricity from renewable sources. However, the integration of renewable energy sources is a challenge due to the intermittency of their production: wind turbines only produce power when there is wind and solar panels do not produce during the night. This implies the addition of storage devices, as batteries or fuel cells, and a complex orchestration of the use of all these power devices together with the IT machines.

The DATAZERO project ([3] (2015-2019) and DATAZERO2 (2020-2024)) project investigate solutions for the design and operation of a datacenter only fueled by renewable energy. The orchestration is based on a negotiation process [4] between the power and energy management (power production and energy storage) and the IT management (workload processing and servers states). This process determines a power command, based on workload and power production forecasts, that will be applied in the next time window, typically the coming 72 hours. The result of the negotiation process is a power command that dynamically controls the flow of power during the time window. It is a time series of power values for each time interval and is called a power profile. In order to operate, the negotiation process needs predictions of the IT power needs during the given time window. The problem is tackled as an offline problem, but it is constrained by the negotiation that iterates with several exchanges between the IT and the power managers before converging to a solution. Since the negotiation process is iterative, the computation of the power profile must be time efficient.

In this work, we tackle the problem of computing the minimum power profile required to process a workload forecast. Note that we do not address the problem of workload forecast that has already been widely studied [5]. Rather, we investigate the problem of transforming a workload prediction into a power command that minimizes the electrical power needs for the next time window in a offline algorithm. This problem is addressed in two steps: finding the maximum processing capacity given a power value and finding the lowest power needed to process an amount of load. Finally, the resulting plan is sent to an online module, which applies it and adapts it to unexpected events, due to the uncertainty of the plan [6].

This paper first contributes with a binary search algorithm that computes a minimal power profile. A minimal power profile is defined as the series of minimal power values that needs to be provided to the machines during the time window. A second contribution is the proposition of multiple variants of an algorithm that, from each power value found by the binary search algorithm, finds the maximum processing capacity that can be reached using the datacenter machines is deduced, in order to schedule the workload under quality of service (QoS) constraints. We propose a MILP and different heuristics to compute the maximum computing capacity for a power value. Simulations are conducted to decrease the runtime of the heuristics by evaluating them against different machine sets (more or less machines, more or less heterogeneous). The approach is then applied to a real workload over a longer period of one year, using different heuristics, which are compared to each other and to the MILP in terms of power consumption. We finally study the impact of assumptions we took about the workload model on our results.

In the following, the context and a motivating example are given in Section 2, related work is detailed in Section 3. Section 4 formally defines the problems of computing the minimum power value from a workload forecast and maximize a processing capacity within a given power value. Section 5 and Section 6 respectively present the binary search algorithm and the heuristics proposed to compute the processing capacity. Section 7 and Section 8presents the simulations conducted to assess these heuristics and their results. Finally, Section 9 summarizes the paper, highlighting the main conclusion.

Note that this paper is an extended work from [7]. It gives more details on the heuristics and presents a wider study on their tuning and performance compared to the original paper. Simulations with a real workload regarding power consumption and QoS and the impact of deadlines, using the MILP and heuristics, are added.

2 Context and motivating example

The context of this work is the Datazero and Datazero2 projects [3] in which we tackle several issues related to the fueling of datacenters with renewable energies, as online scheduling of tasks [8] energy management [6] and dimensioning [9]. We investigate here the problem of transforming a workload prediction into a power command that minimizes the electrical power needs. Instead of considering individual tasks and their needs, which could be hard to predict, we prefer to concentrate on a general load indicator for the workload forecast. A load value hence represents the total amount of work units to be processed during a time duration: it aggregates the work units of several jobs that may concurrently use and share the infrastructure. A workload forecast, the load variation over the coming time window, is represented by a series of load values.

Since the power supply relies exclusively on renewable energy sources, the power management system needs to plan its power sources and storage usage to correctly fuel the machines that will process the workload. Due to technical constraints [3], the power command delivered to the machines must be constant over a minimum time duration, called a time interval, that ranges from 15 minutes to one hour. To give the power usage over a time window (e.g. 72 hours), we thus need to define a power profile that gives, for each time interval, a constant power need value.

To compute a minimal power profile (see Section 5), the proposed algorithm, iterates on each time interval of the time window to find a minimal power value that allows to process the workload. This algorithm relies on the computation of the maximum processing capacity. Note that the time intervals are independent. For this reason, in the following we concentrate on finding the minimal power value for one time interval. Then the same method is applied to each time interval of the time window to determine the whole power profile.

For each time interval, from the minimal power value, an algorithm computes the maximum processing capacity that can be reached using the datacenter machines (see Section 6) . This processing capacity is obtained by switching-on machines of the datacenter and settings them in an adequate DVFS state (Dynamic Voltage and Frequency Scaling). In the following, we call configuration to the set of states of the machines of the infrastructure, *i.e.* their on/off and DVFS states. Last, the algorithm tries to schedule the workload under quality of service (QoS) constraints, using the given processing capacity.

In the datacenter the machines may be heterogeneous [10], [11] hence different configurations provide different computing capacities and have different power consumption. Since the power command must be maintained over a time interval, there is no need to change the configuration during a time interval. The problem is hence to compute, for each time interval, a configuration that is able to process the workload without exceeding the QoS constraint. Then a given configuration defines the power consumption over one time interval and the configurations of the machines over all time intervals define the power profile over the entire time window.

Figure 1 gives a motivating example of the problem. It shows a workload forecast with the number of operations expected to be processed for the coming 5 hours. Note that this workload varies at an higher frequency (a few seconds to a few minutes) than the electrical power and that time intervals are composed of several time steps (in this example, 5 time intervals of one hour, each composed of 12 time steps of 5 minutes). The computing load is defined on time steps, while the power values are defined on time intervals. Note also that the two y-axis with different units on the figure.

The red power profile on Figure 1 is obtained with successive configurations, one for each time interval, that are able to process the load of their interval. It is however preferable to propose a power profile that minimizes the power demand at each time interval in order to save energy, to anticipate periods of under-production in the future and because the energy production forecast is uncertain. The machine configuration is therefore computed with the objective to reduce power demand while keeping the necessary computing power. The green power profile is obtained by minimizing power demand at each time interval, *i.e.* finding the less consuming machine configuration whose computing power is sufficient to process the load. This turns out to be a complex optimization problem as proven in Section 6.

3 Related Work

Our work is positioned in the IT scheduling field, halfway between the online and the offline approaches. Offline scheduling approaches forecast and plan actions without taking feedback from the infrastructure into account and are usually not time-constrained. On the opposite, online approaches do not use forecasts but interact with the infrastructure and are time constrained. Some of them, the online clairvoyant approaches, nevertheless use forecasting methods to guide their decisions. We differ from strict offline approaches in that we are time-constrained and we differ from online ones because we only interact with the infrastructure at each time window.

In order to minimize energy or power consumption while possibly meeting other criteria, different online approaches are considered in the literature. An online approach operates dynamically on the IT infrastructure. In [12], in the context of hybrid geo-distributed datacenters, an online approach manages energy consumption and workload scheduling, using the variation of the electricity price, the power consumption of the cooling and machines, and renewable energy constraints to take decisions. In [13] the authors optimize energy consumption and performance degradation as well as power grid cost for multiple hybrid datacenters, using a genetic based online algorithm. Zhang et al. [14] propose an online power-capping algorithm (PoDD) to maximize performances of homogeneous servers. A proposition for heterogeneous nodes of a datacenter is given in [15]. In [16], the authors minimize the performance degradation and the total energy consumed with different online algorithms. The first is based on regression. It detects overloaded servers and it migrates virtual machines. The second addresses the trade-off between several server metrics such as power consumption, number of migrations and performance. Finally, they migrate virtual machines from over-loaded servers to under-loaded servers. Other methods are proposed to minimize the total energy consumed using a similar method [17], [18], [19], [20], [21]. In [22], an online holistic approach schedules virtual machines to minimize the total energy consumption of the datacenter by considering the datacenter as an ensemble and not trying to divide it into several parts to be treated separately from each other.

In [19], an online multi-objective algorithm optimizes energy consumption, by taking into account QoS, energy, the number of active servers and the number of virtual machine migrations on servers. A similar method in [23] is used by considering DVFS (Dynamic Voltage and Frequency

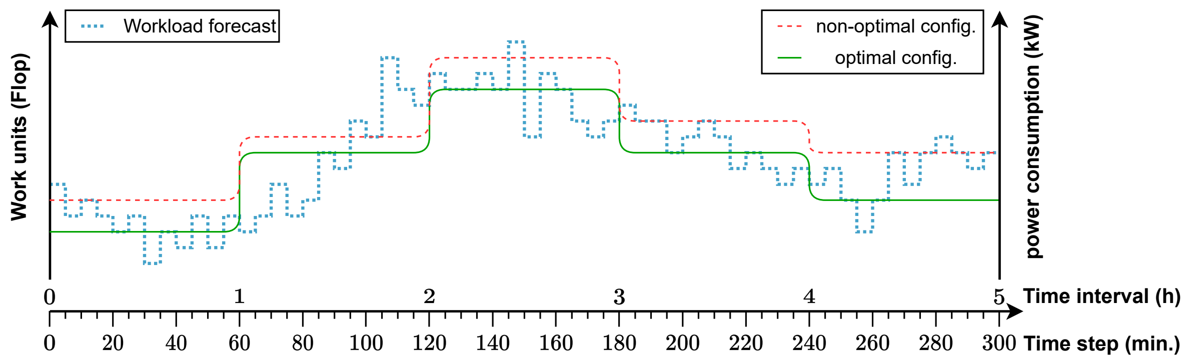


Figure 1: A workload forecast depending on time. Each time interval is composed of several time steps where the workload differs. For each time step the total amount of work units in Flop to be processed is represented. Different machines configurations are possible to process the workload over a time interval. The green line shows an optimal configuration to process the workload while consuming minimum power. The red line gives a configuration that consumes more power.

Scaling), temperature-dependent task scheduling, dynamic resource provisioning, and cooling management. The optimization is also done online, with a non-clairvoyant approach.

In the context of cloud datacenters, in [24], an online clairvoyant method for predicting the total energy consumption of the datacenter is proposed to improve the datacenter energy management. This method uses PCA (Principal Component Analysis) to evaluates the importance of the variables of the equipment. Then a neural network computes the prediction on the total energy consumption, a single value for the coming 20 minutes, the time necessary for the air conditioner to reach a desired temperature. Finally, an online module updates the model based on the forecast errors. In [25], an online clairvoyant method manages servers by switching them -on or -off to deal with the load (QoS) and save energy (power). A forecasting method (Holt Linear Method) of the number of requests per second is used for short-term predictions to find a new server configuration. Compared to an optimized solution, their method shows an energy savings up to 5.4% with better QoS and fewer missed requests. In [6], the authors develop a complementary approach to ours, in the context of a green datacenter. Offline decisions are used by an online module that adapts them to real-time events via different compensation policies, to stay as close as possible to the offline plan. It is responsible for compensating energy utilization by acting on scheduling and servers. Different compensation policies are evaluated according to five metrics about jobs and energy storage sources. Due to the uncertainty of the plan, the results indicate that compensation are necessary and simply following the plan is not enough.

In [26], the authors propose another approach complementary to ours. A proactive offline system minimizes carbon footprint by delaying flexible workloads (simulations, machine learning, data processing) over the next 24 hours for several Google datacenter clusters (typically 20 datacenters on 4 continents consuming more than 15.5 TW). Demand and carbon intensity forecasting models are used to define hourly resource usage limits. More specifically, these hourly resource usage limits are computed from the flexible and inflexible workload forecasts with their respective uncertainties, the hourly carbon intensity forecast, and several constraints related to energy suppliers and commercial and environmental objectives. Forecasts are computed at different times of the day using Exponential Weighted Moving Average (EWMA), whose parameters are determined by minimizing the Mean Absolute Percentage Error. Their results show 1-2% reduction in energy consumption when carbon intensity is the highest. In contrast to us, they use a real-time scheduler-agnostic load shaping mechanism that computes resource usage limits to constrain the scheduler.

Finally, all these studies address the online and offline problem to reduce energy or power consumption and carbon footprint. To the best of our knowledge, there is no work addressing the offline power-capping time-constrained minimization in the case of homogeneous or heterogeneous machines with different amount of work to process, under the constraint of avoiding deadline violations.

4 Problem definitions, model and objectives

In the following, we first formally define the global problem and its associated model, then we define the problems and 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 timescales 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. A time step 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 $t \in \mathcal{T} = \{1, \dots, T\}$. We define Δt as the duration of a time step in seconds.

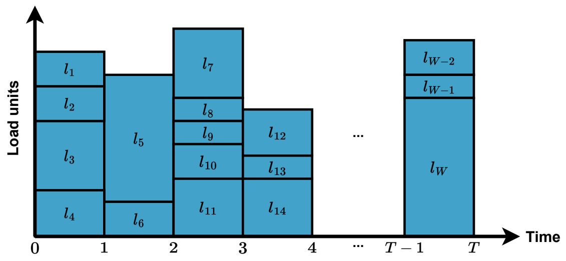


Figure 2: Load parts of a workload forecast. The height of a part represent its amount of operations to be processed (in load units)

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 $k \in \mathcal{W} = \{1, \dots, W\}$. A load part $l_k = (r_k, p_k, d_k)$ 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 Figure 2, the workload is composed of four load parts, l_1, l_2, l_3 and l_4. p_1 and p_2, the amount of operations of l_1 and l_2, are two times smaller than p_4. 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 but they can be processed on different time steps. For instance, in Figure 2, 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 denoted machine i with $i \in \mathcal{M} = \{1, \dots, M\}$. 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 [23]. The DVFS state of a machine is denoted $j \in \mathcal{S}^{(i)} = \{0, \dots, S_i\}$. The set of machines with their DVFS states defines the configuration of the datacenter. We denote $\mathcal{S}$ the set of DVFS states of all machines, $\mathcal{S} = \{\mathcal{S}^{(1)}, \dots, \mathcal{S}^{(M)}\}$. 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 [27]. If machine i is switched-on, it computes $g^{(i)}$ operations per second with $0 \le g^{(i)} \le g^{(i)}_{\max_j}$ while dissipating $power^{(i)}_j$ power per operation per second. Therefore, if a machine i computes several load parts in state j during a time Δt such that $g^{(i)}\Delta t \le g^{(i)}_{\max_j}\Delta t$, 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 $w^{(p)} = \sum_{i \in \mathcal{M}} g^{(i)}\Delta t$.

In addition, to give the users a flowtime guarantee, we define the *opk* value, the total amount of operations missing their deadline, and the D ratio, the amount of operations killed over the amount of operations to be processed during the time interval (see equation (1)). This ratio must not exceed a fixed threshold $D_{\max}$ (see equation (2)) to meet the flowtime guarantee.

$$D = \frac{opk}{\sum_{k \in \mathcal{W}} p_k} \tag{1}$$

$$D \le D_{\max} \tag{2}$$

As previously presented, the global problem can be divided into two sub-problems since determining the minimum power value needed for a time interval requires being able 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 the following, we tackle these two problems one after another.

5 Determining the minimum power value

The problem of minimizing the power value for a time interval under the deadline violation constraint is not complex if we assume to have a function, called *config*, that computes the machine configuration providing the largest computing power for a given value P of power. This function and its underlying optimization problem are presented in the next section.

For this problem the objective function is hence the power value P to be minimized, the constraint is the $D_{\max}$ threshold (2) that limits the D ratio (1) and the inputs are the machine characteristics ($\mathcal{M}$, $\mathcal{S}$), the workload ($\mathcal{W}$) and the set of time steps with their size ($\mathcal{T}$, Δt). Note that the $D_{\max}$ value and the workload can be chosen by the user while the machine characteristics are fixed by the datacenter characteristics.

To solve this problem, we propose a binary search algorithm. Given an ϵ value, Algorithm **??** returns a solution not farther than ϵ from the optimal solution. 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.

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 (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 [28]) 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.

6 Maximizing the computing power

The second optimization problem is to find the most powerful machine configuration that can be fueled with the power P, given 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.

The problem of computing the maximum computing power for heterogeneous machines can be defined with the computing power $w^{(p)}$ as the objective function to be maximized. The variables here are the state of each machine. They are constrained by the total power consumption P of the machine and the maximum computing power of each machine. Finally, the inputs of the problem are the machine characteristics ($\mathcal{M}$, $\mathcal{S}$), fixed in a given datacenter, and the time step size Δt. A more formal definition is given by the MILP presented in Equations (3) to (10).

From the complexity view point, this problem 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 $g^{(i)}_{\max_j}$ and the power consumed by a computing machine hence 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. We can hence conclude that the problem of maximum computing power with a given electrical power is in turn NP-Hard.

Since this problem is NP-Hard, we first designed a MILP (Mixed-Integer Linear Programming).

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

The MILP is described by Equations (3) to (10). The objective function is given by Equation (3) and aims to maximize the computing power of the machines. Using the binary decision variable $x_{i,j}$, the first constraint (Equation (4)) states that a machine i, for all $i \in \mathcal{M}$, must have a single DVFS state j among the 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 (Equation (5)). Then, knowing the DVFS state and the computing power of the machine, the third constraint (Equation (6)) bounds the power consumption of the machine, for all $i \in \mathcal{M}$. Finally, the fourth constraint (Equation (7)) imposes that the total power consumption of the machines must not exceed the power P value given by the binary search algorithm. The variable domain constraints are given by Equations (8), (9) and (10). This linear program is able to deal with heterogeneity in power consumption in homogeneous clusters [29] as the variable $x_{i,j}$ refers to individual machines.

Objective function.

$$\text{maximize} \sum_{i=1}^{M} g^{(i)} \tag{3}$$

Under the following constraints.

$$\begin{cases} \sum_{j=0}^{S_i} x_{i,j} & = 1 & (4) \\ g^{(i)} & \leq \sum_{j=0}^{S_i} (x_{i,j} \times g^{(i)}_{\max_j}) & (5) \\ P_i & = \sum_{j=1}^{S_i} x_{i,j}(static_i + g^{(i)} power^{(i)}_j) & (6) \\ \sum_{i=1}^{M} P_i & \leq P & (7) \end{cases}$$

Variable domain constraints.

$$\begin{cases} \forall i \in \mathcal{M}, \forall j \in \mathcal{S}^{(i)} & x_{i,j} \in \{0,1\} & (8) \\ \forall i \in \mathcal{M} & g^{(i)} \geq 0 & (9) \\ \forall i \in \mathcal{M} & P_i \geq 0 & (10) \end{cases}$$

As shown by the simulations in Section 7.2, 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 of the binary search algorithm, that usually execute more than 15 iterations. 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 since it is repeated for each time interval. As previously explained, the power profile is used in the negotiation process 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. So we have to move toward heuristic solutions. Numerous meta-heuristics, presented in the literature to solve various optimization problems [30], could have been applied here to find solutions. These meta-heuristics, by widely exploring the solution set and because of their implementation complexity, however often take a lot of time while we are time constrained. For this reason we preferred to develop ad-hoc heuristics that incrementally construct the solutions in a shorter time. Note that all these heuristics use the same inputs ($\mathcal{M}$, $\mathcal{S}$, P, Δt) to compute the best computing power $w^{(p)}$ that they can.

Random Choice heuristic: A first trivial heuristic randomly choose the machine to switch-on. When a machine is switched-on, it gets the electrical power it needs to provide the maximum computing power, if it remains enough electrical power, and so its DVFS state is set to the one maximizing the computing power. This step is repeated until the electrical power is consumed and/or there are no more machines to switch-on.

The advantage of this heuristic is its fast runtime, but it provides unsatisfactory results compared to the other heuristics presented in the following.machines

Balance Power-Performance (BPP) heuristic: The BPP heuristic (Algorithm **??**) evaluates the most suitable machines to switch-on, and their DVFS states, according to two metrics: (i) computing power and (ii) performance ratio (Algorithm **??**). For each machine and for each DVFS state, these two metrics are scalars contained in a vector. The set $\mathcal{R}$ contains these vectors. 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 corresponds to the total power consumed by the machine over the provided computing power. The ratio hence represents the power consumption of the machine per unit of computing power (11). This ratio is chosen to minimize the power consumed per unit of computing power: the lower the ratio the more efficient the machine.

$$\forall (i,j) \in \mathcal{M} \times \mathcal{S},\ ratio_{i,j} = \frac{static_i + g^{(i)} power_j^{(i)}}{g^{(i)}} \tag{11}$$

These two metrics are used to compute a normalized score depending on a given power and an α parameter (line 9 in Algorithm **??**). The α parameter (with $0 \leq \alpha \leq 1$) controls the trade-off between computing power and performance ratio. The nearer α is to 1, the more weight is given to the computing power in the choice of the machine to be switched-on, and inversely. This score is then normalized according to the highest computing power and lowest performance ratio that can be obtained with the machines. BPP switches-on the machine with the highest score. Depending on the α parameter value, the configurations proposed by BPP can be different. For this reason, several α values are assessed in order to produce different machine configurations (line 3 in Algorithm **??**) and the algorithm returns the one maximizing the total computing power (line 1 in Algorithm **??**. The algorithm ends when all α values are explored (line 3 in Algorithm **??**) and, for each α value, when there is not enough power available to switch-on more machines or when there are no more machines to switch-on (line 5 in Algorithm **??**).

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

Best State Redistribute Without Static (BSRWS) heuristic: The BSRWS heuristic (Algorithm **??**) focuses on the performance ratio of the machines. This heuristic switches-on as many machines with the best performance ratio (Algorithm **??**) 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 pay the static power cost to switch-on a new machine, the remaining power is redistributed to the switched-on machines (Algorithm **??**). This redistribution increases the DVFS state of the switched-on machines and thus their computing power. While there is power left to redistribute, the heuristic computes, for each machine type and for each DVFS state greater than the current DVFS state and that increases the computing power, the performance ratio and the computing power that can be gained. The heuristic first increases the DVFS state of the machines with the highest performance ratios without including static power, since the considered machines for redistribution are the ones that are switched-on. This heuristic differs from the previous one by not focusing on maximizing the computing power.

The advantage of this heuristic is its accuracy with a satisfactory runtime, which however increases with electrical power value. In the heterogeneous case, some solutions deviate from the optimal one. The heuristic switches-on too many machines and too much static power is consumed, whereas the best solution is to switch-on fewer machines and distribute the power to increase the DVFS state of the machines.

Best State Redistribute Without Static And Removing (BSRWS-AR) heuristic: The BSRWS-AR heuristic focuses on the performance ratio of the machines and explore more machine configurations. This heuristic is an improvement of the BSRWS heuristic that avoids to switch-on too many machines. In this heuristic BSRWS is run several times and, at each iteration, a machine of the configuration is switched-off. The remaining power is then redistributed to the remaining machines to increase their DVFS state and so their computing power. This heuristic allows to test configurations with fewer switched-on machines. Removing a machine may indeed limit the expense in static power.

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

7 Tuning and assessing the heuristics

All the previously presented heuristics implement their own method to find, as well as they can, solutions to the optimization problem of maximizing the computing power, given the machine characteristics ($\mathcal{M}$, $\mathcal{S}$) and the available power P. They are possible implementations of the *config* function used in the binary search algorithm to solve the minimum needed power. The objective of this section is to assess the quality of the proposed heuristics, i.e. how good is the returned configuration (the higher the $w^{(p)}$ value returned, the best the solution) and how much time they take to find these solutions, outside of the scope of the binary search algorithm.

After tuning the heuristics, we compare them against each other and with the MILP to conclude which heuristic is the most suitable for the initial problem.

7.1 Improving heuristic runtimes

Since the BPP and BSRWS-AR heuristics use parameters in their definition, we study their impact on their results. By design, these heuristics explore several machines configurations to propose the most powerful configuration considering a given input power consumption which could be time consuming. We seek to limit their runtime while maintaining a good efficiency.

The BPP (Balance Power-Performance) heuristic is computed with α values ranging from 0 to 1 by steps of 0.05 for each electrical power value to propose several machine configurations. The heuristic then selects the most powerful machine configuration. As with the BPP heuristic, BSRWS-AR explores several machine configurations and returns the one that maximizes the computing power. This contributes to their efficiency, but increases their runtime. These heuristics are executed for each iteration of the binary search algorithm and for each time interval of the power profile (*i.e.* more than 1080 times): this results in a longer runtime for the heuristics and therefore a longer runtime for the construction of the power profile. For this reason, they are assessed in this section to minimize their runtime, while maintaining their efficiency as far as possible. Several simulations with more or less homogeneous and more or less machines are conducted to assess results and ensure robust optimization.

7.1.1 Settings

For all the heuristic assessment we preferred to use the characteristics of real machines to position our work in a realistic case. The machine types and characteristics used are taken from Grid5000 platform [31]. Note that we have also run tests with randomly chosen machine characteristics that produced the same results.

The heuristics are compared in the context of a medium-sized datacenter of 267 kW [3] that includes 1241 machines divided into 10 machine types. Data on the characteristics of the machines are known in advance [32], [33]. For each type of machine, we know all the data described in Section 4. All data are average values coming from real simulations performed on Grid5000 in the context of the ANR ENERGUMEN [34] project. Note that such data are seldom available for datacenters since they require laborious practical measures to be obtained.

A total of 300 different machine sets are investigated to assess the heuristics. They all differ from each other in terms of number of machines and number of machine types to ensure robust optimization. The 300 machines sets are distributed into 30 groups, each group being composed of 10 machine sets. These 30 groups differ in the number of machines: 60, 120, 180, 240 and 300; and in the number of machine types: 1, 2, 3, 4, 5 and 6. In each group, the 10 machine sets differ in the selection of the machine type(s) considered. For example, in the group composed of 10 machine sets, each with 180 machines and 3 machine types (see Figure 3), the only difference is in the machine types considered. The first machine set is composed of 60 Taurus, 60 Parasilo and 60 Grisou. The second is 60 Parasilo, 60 Grisou and 60 Grimoire. etc... For each simulation on the machine set investigated, the power value varies by steps of 1 W from the minimum power required to switch-on a machine to the maximum power required by all machines of the set.

For the simulations, the MILP and the heuristics have been implemented in Python. The source code, the data sets and comparisons are available here (zip file) and an artifact is published here artifact. Note that all simulations in the remainder of the paper 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.

After running the simulations, Two Wilcoxon-Mann-Whitney tests [35] are used to evaluate the hypothesis that the distributions of the computing power provided by, respectively, the heuristics and their runtime are similar. A P-value lower than an threshold of 0.05 is considered to denote a statistically significant difference between the results.

To illustrate the characteristics of the machine set, Figure 3 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 whatever 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 use this machine, depending on power. For instance, if the available power is 1000 W, the best is to use 9 Gros machines than any other configuration (visually and confirmed by the MILP solution), if we have these 9 Gros machines at hand. Otherwise, with less Gros machines, a different configuration has to be used involving Gros and other types of machines. Note that some performance ratios of some machine types cross others, which means that the machine choice depends on the available power, and that the performance of some machines decreases before reaching their maximum power consumption, which means that powering a machine at its maximum is not always the best choice.

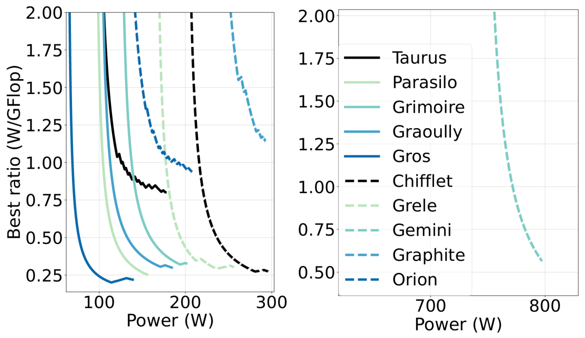
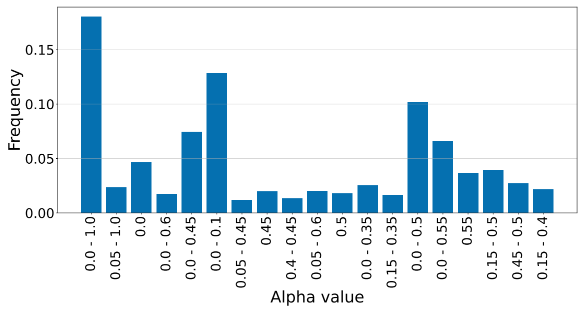


Figure 3: Best performance ratio in W/GFlop depending on power, whatever the DVFS state of the machine. The lower the ratio the more efficient the machine.

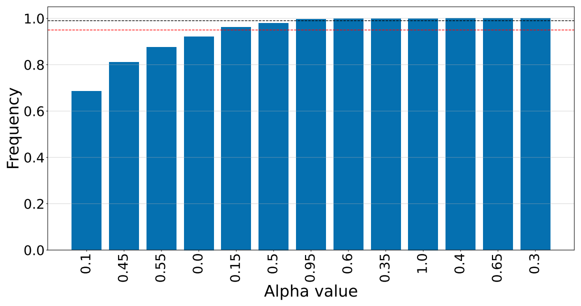
In the following, the BPP and BSRWS-AR heuristics are assessed with different parameter values, to observe the gain in runtime and, to a lesser extent, the loss in efficiency.

7.1.2 BPP heuristic

The objective here is to determine the most relevant α values to obtain the best configuration of the machines, whatever the machine set in the infrastructure and the power value. Reducing the number of α values means reducing the number of configurations explored and the runtime of the heuristic. For each power value, the α values giving the configuration that maximizes computing power are stored. This is represented by a list of α values, called α value combination.



(a) Frequency of occurrence of α value combinations that all find the best machine configuration.



(b) Cumulative frequency of occurrence of best α values that find the best machine configuration. The red and black lines represent a cumulative frequency of 95% and 99%, respectively.

Figure 4

Figure 4a shows the frequency of occurrence of α value combinations that all find the best machine configuration. This figure is based on 14 million executions resulting from the 300 machine sets and power values tested. α value combinations with frequencies below 1% are not displayed for better visibility. The x-axis represents the α value combinations for which the best machine configuration has been found. In over 17% of cases, all the α values find the best configuration (*i.e.* the combination $0.0 - 1.0$, α values from 0 to 1 by steps of 0.05). The α value combination $0.0-0.1$ (i.e. α values 0, 0.05 and 0.1) find the best configuration in over 12% of cases, while other α values do not find any better configuration. Note that some α values are included in a large majority of α value combinations. For example, this is the case for the values 0 and 0.1. These two values are able to provide, in the vast majority of cases, the best configuration that can be found with the BPP heuristic. It is also noteworthy that values above 0.6 are not represented in the figure except in the cases $0.0 - 1.0$ where all the α value combinations find the best configuration, and $0.05 - 1.0$. This means that they are the only ones to find the best configuration in few cases (about 1% of all cases). This suggests that giving more weight to the performance ratio (i.e. with α values lower than 0.5) leads more often to the best configuration.

Figure 4b shows the cumulative frequency of occurrence of best α values that find the best machine configuration. In other words, this is the frequency of occurrence of α values in the combinations shown in Figure 4a. The contribution of α values are highlighted by sorting them from most to least significant. In 69% of cases, an α value of 0.1 is sufficient to find the best machine configuration. Adding an α value of 0.45 increases the number of cases by 11%. The red and black lines shows the number of α values to be tested to obtain the best configuration in, respectively, 95% and 99% of cases, respectively. To maintain at least 95% efficiency, 5 α values are required, and 7 for 99%.

Table 1 shows the results of the BPP heuristic before and after the reduction to the 5 and 7 best α values shown in Figure 4b. When reducing the number of α values, the runtime of the heuristic is divided by a factor of 3 to 5. In terms of computing power, the results are very similar according to the Wilcoxon-Mann-Whitney test. In both cases (BPP with 5 values and BPP with 7 values), the P-values with BPP with all α values are 0.99 . There is hence no significant difference in computing power. It is however significant in terms of runtime, with P-values of 0.00 in both cases.

7.1.3 BSRWS-AR heuristic

As explained in Section 5, the BSRWS-AR heuristic explores more machine configurations by removing machines one by one from the initial configuration found by the BSRWS heuristic. In doing this, from a specific number of removed machines, the computing power of the configuration decreases and cannot reach maximum computing power. There is so no benefit to explore configurations after a specific number of machines removal. We seek here to find this specific number, to reduce the number of explored configurations, and therefore reduce the heuristic runtime while limiting the impact on its efficiency.

Table 2 shows the maximum percentages of machine removals to reach the best configuration for each of the conducted simulations. This table summarizes the results of the 300 simulations. Each cell of the table is the maximum percentage obtained from 10 simulations. For example, the cell corresponding to one machine type and 60 machines is the result obtained from 10 simulations, which were conducted with 60 machines and 1 machine type. They differ in the type of machine used. In the same way, The cell corresponding to 3 machine types and 180 machines is the result obtained from 10 simulations with 180 machines by modifying the 3 machine types in the 10 simulations.

From the results we can see that increasing the number of machines does not influence the maximum percentage of machines to be removed, but increasing machine heterogeneity does. Beyond these percentages of removed machines, the machine configurations explored are less good. Finally, imposing a limit of 20% of machines removed from the configuration seems reasonable to reduce the number of explored configurations, while maintaining maximum efficiency.

Table 3 shows the results of the BSRWS-AR heuristic before and after application of the limit of 20% of removed machines in the context of a medium-sized datacenter. Reducing the number of machines in the configuration allows to divide the heuristic runtime by a factor 3. In terms of computing power, the results are very similar according to the Wilcoxon-Mann-Whitney test. The P-value between BSRWS-AR and BSRWS-AR with the limit on the number of removed machines is 1.00. While there is no significant difference in computing power, it is significant in terms of runtime, with P-values of 0.0. Note however that the reduced runtime is still too long, for the same reason as explained in Section 5 for the MILP.

7.2 Evaluation of the heuristics

The objective of this section is to evaluate and compare the heuristics against each other and with the MILP in terms of efficiency and runtime. In the case of the BPP and BSRWS-AR heuristics, we use the optimized version of the heuristics. For the BPP heuristic, we use only the 7 alpha values determined previously. For the BSRWS-AR heuristic, we use the limit of 20% of machines removed.

7.2.1 Settings

As in Section 7.1, we consider in this simulation the example of a medium-sized datacenter with the same machine set and characteristics. We remind the reader that on this part there is no related work in the literature to compare to, hence we only provide our heuristic results. A Wilcoxon-Mann-

Table 1: Results for the BPP heuristic before and after application of the reduction in the number of α values considered.

	BPP	BPP with 7 α values	BPP with 5 α values
Avg. dev. (%)	0.12	0.12	0.12
Median dev. (%)	0.04	0.04	0.04
Max dev. (%)	3.15	3.15	5.47
Avg. Runtime (s)	9.07×10^{-3}	2.61×10^{-3}	1.97×10^{-3}

Table 2: Maximum percentage of machines to switch-off to reach the best configuration (in terms of computing power), depending on the number of machines and the number of machine types.

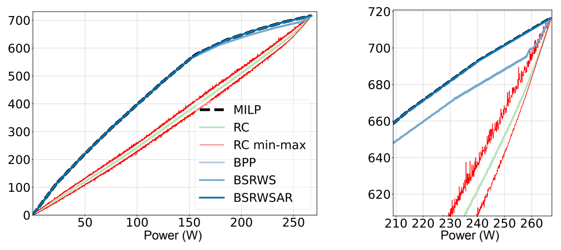
	1 type	2 types	3 types	4 types	5 types	6 types
60 machines	20%	20%	20%	16.67%	16.67%	12.5%
120 machines	20%	20%	20%	16.67%	16.67%	12.5%
180 machines	20%	20%	20%	16.67%	16.67%	12.5%
240 machines	20%	20%	20%	16.67%	16.67%	12.5%
300 machines	20%	20%	20%	16.67%	16.67%	12.5%

Whitney test [35] is then used by considering that a P-value lower than a threshold of 0.05 denotes a statistically significant difference between the results.

7.2.2 Evaluation

Figure 5 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 of our medium-sized datacenter when running at maximum frequencies.

The RC (Random Choice) heuristic is run 200 times to show the dispersion of the solutions and the average computing power for each power value. The sample size for each power value is justified using the central limit theorem [36] from which the formula for minimum sample size is derived [37]. We considered the maximum standard deviation obtained (7200.87) to estimate the mean value with a 95% confidence interval of half-width of 1 TFlops. The minimum and maximum computing power are shown in red.



(a) (b)

Figure 5: (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.

The RC heuristic significantly deviates from the optimal solution with an average deviation of 31.84% (Table 4). 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 the results of other simulations. The BPP heuristic with seven α values for each power value and BSRWS-AR heuristic are the closest to the optimal solution. Their average deviation from the optimal is 0.12% and 0.03% respectively (Table 4). 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 not presented here. From 150 kW to 260 kW, the deviation from the optimal is more significant for BSRWS (Figure 5). But different simulations show that this is not always the case.

Table 3: Results for the BSRWS-AR heuristic before and after application of the limit in the number of machines switched-off.

	BSRWS-AR	BSRWS-AR with a limit of 20%
Avg. dev. (%)	0.03	0.03
Median dev. (%)	0.00	0.00
Max dev. (%)	2.04	2.04
Avg. Runtime (s)	1.61	0.55

The BSRWS heuristic has an average deviation of 0.65%. Note that from 260 kW onward, all the heuristics reduce their deviation from the optimal, because there is enough power to switch-on all the machines and to increase their DVFS state and with 267 kW the solution is trivial since all the machines can be used at their maximum capacity.

As previously, we verify if the solutions of the heuristics significantly differ from the MILP by using a Wilcoxon-Mann-Whitney test. In terms of computing power, the RC heuristic is the only one to provide significantly different solutions from the MILP with a P-value of 0.00. BSRWS heursitcis provide solutions that barely pass the test with a P-value of 0.09. BPP and BSRWS-AR heuristics provide solutions statistically very close to the MILP with P-values of 0.87 and 0.97, respectively.

Figure 6 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-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. In terms of runtime, all heuristics are statistically significantly different from the MILP, with P-values of 0 in all four cases.

Considering the whole process, 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.

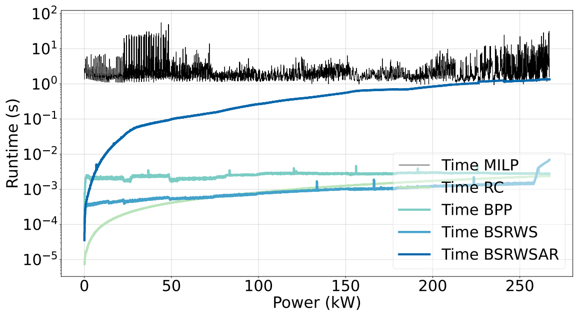


Figure 6: Runtime of the MILP and the heuristics depending on power

8 Simulation with a real workload

As previously said, the problem of finding the minimal power value for one time interval, without taking the configuration computation done by the *config* function into consideration, can be solved with Algorithm **??**, with a solution not more far than ϵ from the optimal solution. There is hence no need to assess it alone. On the other hand, the use of different heuristics in the *config* function may impact the global energy consumption on several time steps. In this section, we challenge the binary search algorithm, using the MILP or one of the the heuristics as the *config* function, with a real workload. We first analyze the ranges of power consumption that are most demanded by the workload. Then we compare the heuristics in terms of total energy consumed by the machine configurations required to process the workload, including the energy needed to switch-on and switch-off the machines. Then, we analyze the impact of deadlines and computing power on the quality of service. With these simulations we seek to show how the algorithms behave in a real context.

8.1 QoS and power consumption

In this simulation we compare the binary search algorithm with the MILP and each heuristic, according to several metrics: total energy consumed and

Table 4: Average, median and maximum relative deviation in percentage and computing power 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**
Max dev. (%)	-	52.43	3.15	3.15	**2.04**
Avg. Runtime (s)	2.83	1.16×10^{-3}	2.61×10^{-3}	$\mathbf{1.03 \times 10^{-3}}$	0.55

energy consumed by switching-on/off machines.

8.1.1 Settings

For this simulation, we assess the binary search algorithm with the Metacentrum workload [38], [39].We made this choice because cloud workloads as Google Traces [40], [41] or Alibaba Traces [42], [43] do not last enough time. We want a large processing period to maximize the number of time intervals and observe the maximum number of changes from one time interval to the next. Moreover, the machine characteristics of these cloud workloads (CPU, number of cores, memory, etc...) are unknown, while knowing these characteristics is necessary to provide realistic data on their performance and power consumption. The Metacentrum workload, running for over 2 years, is composed of more than 5 million jobs processed on machines with known characteristics. These jobs were run in a heterogeneous environment system including 33 clusters that evolved over the period. The submit time, wait time, runtime and number of allocated cores are given for each job. The environment system and job data are used to set up our simulation under the following assumptions:

- Only Metacentrum non-exotic nodes [44] are selected to be closer to a cloud datacenter. An exotic node is specific in terms of architecture or hardware (CPU, GPU, Memory). Only non-exotic partitions 7, 9, 11, 14, 15, 18 and 19 are kept for a total of 305 machines. These partitions have servers with 12 to 16 CPU cores and 12 GB to 134 GB of memory. For the simulation, we consider a heterogeneous set of Grid5000 machines similar in number and CPU core to those in the selected Metacentrum partitions since the data on Metacentrum machines (power consumption, computing power) are unknown. In particular, we take 113 Taurus, 112 Parasilo, 26 Grisou, 47 Grimoire and 7 Graoully to provide a heterogeneous set similar to Metacentrum's machines (the number of cores in this machine set is the same as the selected machine partitions in Metacentrum), for a maximum computing power of 142.92 TFlops and a maximum power consumption of 52.90 kW.

- For the simulation, we consider the period from July 30, 2013 to September 19, 2014, totalling 7406 hours and more than one million jobs to process. During this period, the set of selected partitions remains the same [39].

- Load parts are derived from Metacentrum jobs and are defined by a submit time, a deadline and a number of operations to be processed. The submit time of load parts is the same as for jobs. As there are no deadlines defined for each job in Metacentrum, the deadline is considered to be the sum of the wait time and the runtime of Metacentrum jobs. Lastly, the number of operations to be processed is determined by the product of the number of cores allocated to the job, its runtime and a factor determining the number of Flops per core per second. For the simulation, the factor used is the computing power for one core in the lowest DVFS state of the Taurus machine from Grid5000 [31] , *i.e.* 9.72 GFlops, because the characteristics of this machine are the closest to the ones of Metacentrum. Figure 7 shows the number of Metacentrum jobs and the number of operations to be processed in Flops along time, as previously described and applied to each load part. The number of jobs and the number of operations to be processed are not correlated, as the job runtime is also considered when computing the number of operations. The time intervals composed of a large number of jobs, typically more than 4000, is not the time intervals composed of the largest number of operations to be processed. These time intervals are mostly composed of jobs with runtime that are relatively short compared to other time intervals. The number of operations to be processed over a time interval rarely exceeds 2 000 000 TFlops dispersed over the 3600 time steps of a time interval.

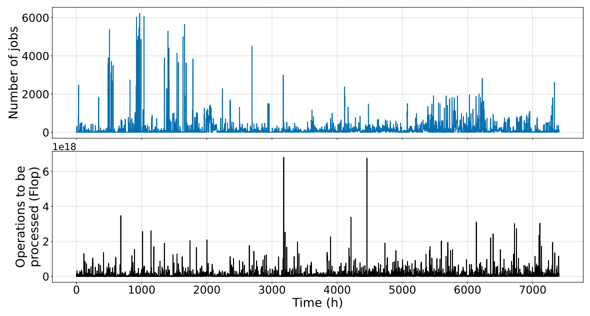


Figure 7: Top: Number of Metacentrum jobs along time. Bottom: Number of operations to be processed in Flop along time.

We initiate the simulation with a threshold $D_{\max} = 1$ in the binary search algorithm **??** to impose the processing of all operations. Note that processing all the operations may still be impossible if the maximum machine configuration is not sufficient to process all the operations over a time step. We set $\epsilon = 0.01$ to obtain a reasonably accurate power value.

While the energy consumed by switching-on and -off machines is not considered in the model, it is included to provide an additional point of comparison between the heuristics. From a time interval to another, the machines to be switched-on and switched-off may differ. A machine is switched-on at the end of the previous time interval if it is required at the current time interval. Similarly, a machine is switched-off at the beginning of the current time interval if it is on and is no longer required.

8.1.2 Results

Figure 8 shows the number of operations to be processed (same as in Figure 7), the number of delayed operations and the rate of processed operations depending on time. The middle figure highlights time intervals during which operations are killed. Overall, the rate of operations killed is 2.48×10^{-9}. The scheduler domino effect [28] and load parts with short deadline but large number of operations intensify this effect. Time intervals with the highest number of operations are not necessarily those with the highest number of operations killed. Despite imposing the processing of all operations, the rate of operations processed is greater than 0.99. The quality of service is met for 98.42% of time intervals, i.e. all operations are processed during these time intervals. For 3 time intervals, the quality of service is less than 0.9975, but remains greater than 0.99. Whatever the method used in the binary search algorithm to determine a configuration and provide computing power, the result for processing operations is the same, since all methods (MILP and heuristics) provide with this workload the same computing power, but not necessarily the same machine configuration and power consumption.

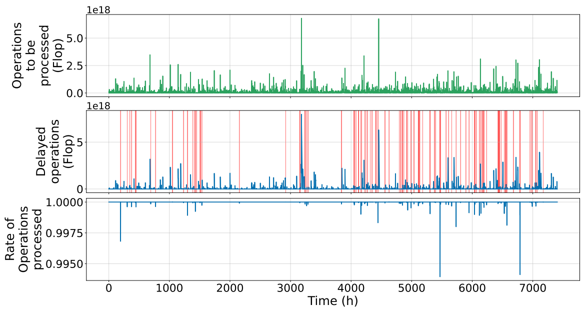
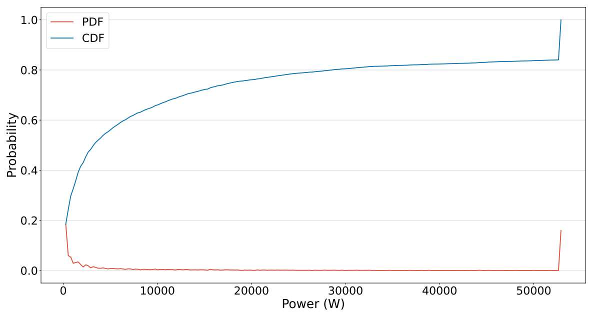


Figure 8: Top: Number of operations to be processed along time. Middle: Number of delayed operations (blue) and time intervals where operations are killed (red). Bottom: Rate of operations killed.

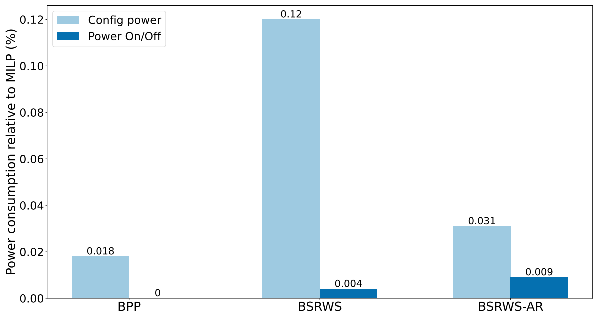
Figure 9a shows the Probability Density Function (PDF) and the Cumulative Density Function (CDF) of the power consumed to process load parts over the entire period. For more than 18% of the time intervals, power consumption is zero and all machines are switched-off and for 50% of the time intervals, the power consumption is less than 3.3 kW, i.e. 6% of the maximum power that can be requested by the machines. The maximum power required by the machines is reached for 16% of the time intervals. This is not surprising, since we have imposed the processing of all operations, which minimizes the number of operations delayed over the next time intervals. We remind the reader that we use Metacentrum HPC workload that is more sporadic than what can be observed in a cloud workload [45].

The energy consumed to process load parts on all time intervals with the binary search algorithm using MILP is 101.79 MWh. Over one time interval, depending on the heuristic used, the binary search algorithm may require more power to provide a configuration for the same computing power as the MILP, as shown by Figure 5 in Section 7.2. Figure 9b shows the percentage of total excess power required to process load parts by the binary search algorithm using the BPP, BSRWS and BSRWS-AR heuristics over the entire period compared to the MILP. BPP consumes the least excess power with 0.018% (*i.e.* 18 kW). BSRWS and BSRWS-AR respectively consume 6.7 and 1.7 times more than BPP.

Figure 9b also shows that heuristics can provide configurations that are more or less power-intensive in terms of switch-on and switch-off power consumption. Over the entire period, MILP consumed 1.80 MWh to switch-on and -off machines, i.e. 1.73% of all energy consumed. On the other hand, the heuristics consumed 41 Wh to 9232 Wh more than the MILP to switch-on and -off machines. Overall, BPP is the least power-intensive and the fastest heuristic to provide a satisfying configuration.



(a) Probability Density Function (PDF) and the Cumulative Density Function (CDF) of the power consumed to process load parts.



(b) Percentages of total excess power required by the binary search algorithm using different heuristics (Config power) and of total excess power required to switch-on/off machines (Power on/off) versus MILP.

Figure 9

8.2 Impact of load part deadlines

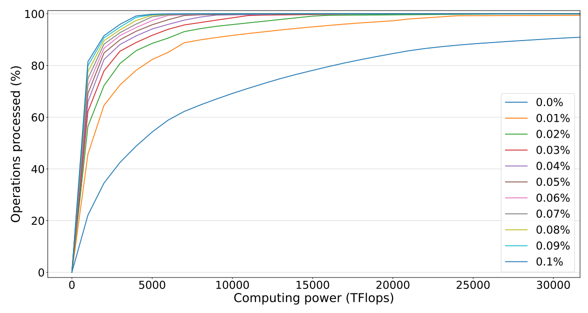
The deadline chosen in Section 8.1 (sum of wait time and runtime) has an impact on the number of operations processed, delayed and killed. Load part deadlines and the machine set used, and hence computing power, have a significant and decisive impact on the number of operations processed. For these reasons, in this section we investigate the number of operations processed (and consequently the number of operations killed) depending on the deadline assumption selected and the computing power.

8.2.1 Settings

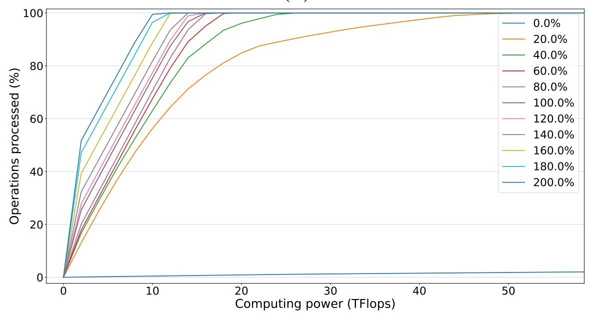
We focus on two cases: in the first one, for all load parts, deadlines are uniformly fixed; in the second one, a multiplicative factor is applied to the assumption presented in Section 8.1 In the case where a fixed deadline is applied to all load parts, we simply vary the value of the deadlines in seconds. The same principle is applied in the other case, where the multiplicative factor varies. In this simulation, we take 10 days (from August 9, 2013 to August 19, 2013) of the Metacentrum workload to reduce the runtime of the simulation, as we run the scheduler for different values of deadline and computing power.

8.2.2 Results

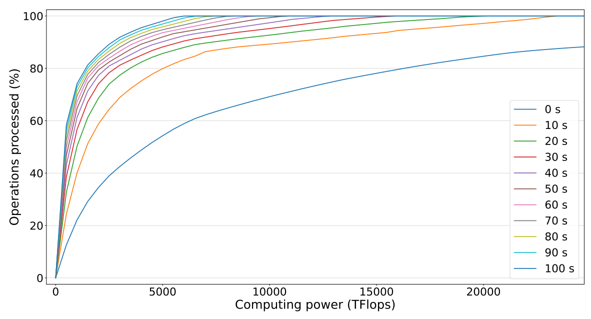
Figure 10 shows the impact the deadlines on the percentage of operations processed over a time interval, depending on computing power. Unsurprisingly, in all cases, the lower the deadlines for the operations, the greater the computing power required to process all of them. When deadlines are very small (just a few seconds), the computing power required is on the order of ten thousand TFlops. On the opposite, when the flexibility of the workload is very large, the computing power required to process the maximum number of operations is on the order of only a hundred TFlops. A minor variation in the workload flexibility results in a major variation in the computing power required. For example (Figure 10a and Figure 10b) to reach 100% of processed operations, 10 TFlops are needed when deadlines are set to 200% of the real execution, 16 TFlops for 100% and 52 TFlops for 20%. With fixed deadlines (Figure 10c and Figure 10d), with a computing power of 45 TFlops and deadlines of 1000s, 40% of the operations are processed. This value drops to 30% for deadlines of 500s. We also noted that without flexibility (deadlines at 0), 176.000 TFlops would be needed. We can therefore see that flexibility has a significant impact on QoS, and can be an efficient way to reduce the need for computing power, and by extension the need for electrical power. On the other hand, flexibility has a direct impact on user satisfaction, which can be a limiting factor.



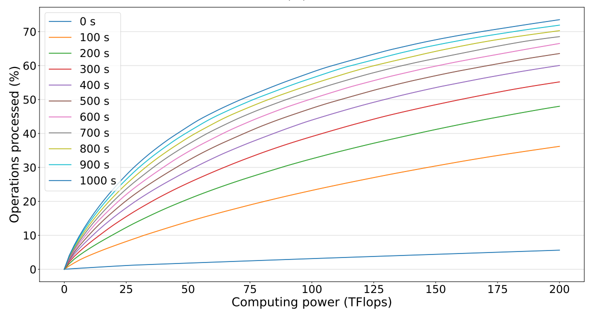
(a)



(b)



(c)



(d)

Figure 10: (a) and (b): Percentage of operations processed depending on computing power by applying a multiplicative factor. (c) and (d): Percentage of operations processed depending on computing power by applying fixed deadlines (in seconds).

Depending on computing power, more or less operations can be processed. A given computing power value can be obtained with different machine sets. Figure 11 shows the possible power consumption for a given computing power, depending on the machine set by using machines and data from Grid5000. The most power-efficient sets are the Gros machines, with the lowest static power and a good performance ratio. Inversely, the most power-intensive are the Graphite machines, with one of the highest static power and a very poor performance ratio. For example, for a machine set with a computing power of 176 TFlops, the most power-efficient set is the one with 278 Gros machines consuming 38.6 kW, while the set with 688 Graphite machines consumes 201.2 kW. The most and least power-intensive sets are homogeneous. But these sets are not common in a datacenter, since machines may be homogeneous a priori in their technical characteristics, but not in their performance and power consumption [29] and because datacenters regularly update part of their machines (see the Metacentrum example already described with several partitions).

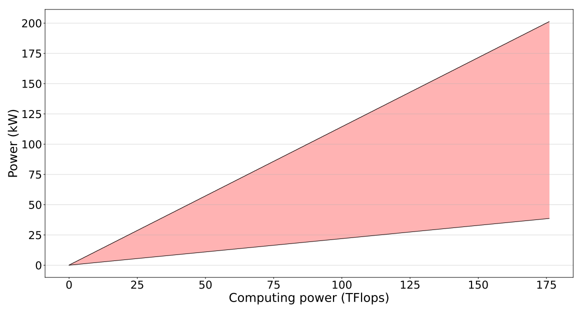


Figure 11: Possible power consumption for a given computing power depending on machine set

9 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 time-efficient heuristics. Two of them explore several possible machine configurations and select the most efficient. In a first step, several simulations are conducted to optimize them to limit their runtime and maintain their efficiency. Second, we assess the quality of the heuristics against the MILP to 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.

In a third step, we assess the binary search algorithm and heuristics with a real workload to analyse the power consumed in a practical case and the impact of deadlines and computing power on the quality of service. Results show that the power consumption is less than 6% of the maximum power that can be requested by the machines half of the time, and maximum power consumption is required 16% of the time. The power consumed for switching-on and -off machines corresponds to 1 to 2% of total consumption. Then, the impact of deadlines and computing power on quality of service is studied. We exhibit that a minor variation in the workload flexibility results in a major variation in the computing power required. However, depending on the machine set in the datacenter, for a given computing power, power consumption varies significantly.

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 implement and test the binary search algorithm with the BPP heuristic in a real environment (DATAZERO2 platform) to assess its resilience against uncertainty. Testing all the heuristics against a wider set of machines is also in our plan, but this requires a hard work of machine characterization, as it was done in the ANR ENERGUMEN [34] project. Last, we have seen that flexibility has a significant impact on QoS, and can be an efficient way to reduce the need for electrical power. However, it can be a limiting factor as it has an impact on user satisfaction. One possible solution to be explored could be to have job classes with variable flexibility.

Acknowledgments and Data Availability Statement

This work was supported by the DATAZERO2 project ("ANR-19-CE25-0016") and 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.

CRediT authorship contribution statement

All: Conceptualization, Methodology, Investigation, Validation, Writing, **Damien Landré**: Software, Formal analysis **Jean-Marc Pierson**: Funding acquisition

References

[1] IEA, Data centres and energy – from global headlines to local headaches?, iea, paris, www.iea.org/commentaries/data-centres-and-energy-from-global-headline-to-local-headaches (2019).

[2] E. Masanet, A. Shehabi, N. Lei, S. Smith, J. Koomey, Recalibrating global data center energy-use estimates, Science 367 (6481) (2020) 984–986. arXiv:https://www.science.org/doi/pdf/10.1126/science.aba3758, doi:10.1126/science.aba3758. URL https://www.science.org/doi/abs/10.1126/science.aba3758

[3] J. Pierson, G. Baudic, S. Caux, B. Celik, G. Da Costa, L. Grange, M. Haddad, J. Lecuivre, J. Nicod, L. Philippe, V. Rehn-Sonigo, R. Roche, G. Rostirolla, A. Sayah, P. Stolf, M. Thi, C. Varnier, Datazero: Datacenter with zero emission and robust management using renewable energy, IEEE Access 7 (2019). doi:10.1109/ACCESS.2019.2930368.

[4] M. Thi, J. Pierson, G. da Costa, P. Stolf, J. Nicod, G. Rostirolla, M. Haddad, Negotiation Game for Joint IT and Energy Management in Green Datacenters, Future Generation Computer Systems 110 (2020) 1116–1138. doi:10.1016/j.future.2019.11.018. URL https://hal.archives-ouvertes.fr/hal-02941028

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

[6] I. F. de Nardin, P. Stolf, 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, IEEE, 2022, pp. 1–6.

[7] L.-C. Canon, D. Landré, L. Philippe, J.-M. Pierson, P. Renaud-Goud, Assessing power

needs to run a workload with quality of service on green datacenters, in: European Conference on Parallel Processing, Springer, 2023, pp. 229–242.

[8] I. F. de Nardin, P. Stolf, S. Caux, Evaluation of heuristics to manage a data center under power constraints, in: 2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC), 2022, pp. 1–8. doi:10.1109/IGSC55832.2022.9969362.

[9] M. Benaissa, G. D. Costa, J.-M. Nicod, Standalone data-center sizing combating the overprovisioning of the it and electrical parts, in: 2022 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW), 2022, pp. 57–62. doi:10.1109/SBAC-PADW56527.2022.00019.

[10] W. Wang, B. Li, B. Liang, Dominant resource fairness in cloud computing systems with heterogeneous servers, in: IEEE INFOCOM 2014-IEEE Conference on Computer Communications, IEEE, 2014, pp. 583–591.

[11] C. Reiss, A. Tumanov, G. Ganger, R. Katz, M. Kozuch, Heterogeneity and dynamicity of clouds at scale: Google trace analysis, in: Proceedings of the third ACM symposium on cloud computing, 2012, pp. 1–13.

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

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

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

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

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

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

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

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

[20] K. Chang, S. Park, H. Kong, W. Kim, Optimizing energy consumption for a performance-aware cloud data center in the public sector, Sustainable Computing: Informatics and Systems 20 (2018) 34–45.

[21] C. Thiam, G. D. Costa, J. Pierson, A decentralized algorithm to revisit the debate of centralization and decentralization approaches for cloud scheduling, in: C. Klein, B. Donnellan, M. Helfert (Eds.), Proceedings of the 7th International Conference on Smart Cities and Green ICT Systems, SMARTGREENS 2018, Funchal, Madeira, Portugal, March 16-18, 2018, SciTePress, 2018, pp. 165–172. doi:10.5220/0006682901650172.
URL https://doi.org/10.5220/0006682901650172

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

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

[24] Y. Hsu, K. Matsuda, 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), IEEE, 2018, pp. 321–330.

[25] C. Santana, J. C. Leite, D. Mossé, Power management by load forecasting in web server clusters, Cluster Computing 14 (2011) 471–481.

[26] A. Radovanović, R. Koningstein, I. Schneider, B. Chen, A. Duarte, B. Roy, D. Xiao, M. Haridasan, P. Hung, N. Care, et al., Carbon-aware computing for datacenters, IEEE Transactions on Power Systems 38 (2) (2022) 1270–1280.

[27] V. S. Shekhawat, A. Gautam, A. Thakrar, Datacenter workload classification and characterization: An empirical approach, in: 2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS), IEEE, 2018, pp. 1–7.

[28] G. Lipari, Earliest deadline first, Scuola Superiore Sant Anna, Pisa-Italy (2005).

[29] M. Diouri, O. Glück, L. Lefevre, J.-C. Mignot, Your cluster is not power homogeneous: Take care when designing green schedulers!, in: 2013 International Green Computing Conference Proceedings, IEEE, 2013, pp. 1–10.

[30] A. Alorf, A survey of recently developed metaheuristics and their comparative analysis, Engineering Applications of Artificial Intelligence 117 (2023) 105622. `doi:https://doi.org/10.1016/j.engappai.2022.105622`. URL `https://www.sciencedirect.com/science/article/pii/S0952197622006121`

[31] Clusters Grid5000, `https://www.grid5000.fr/`, [Online; accessed 17-October-2023] (2022).

[32] K. Pedretti, R. Grant, J. L. III, M. Levenhagen, S. Olivier, L. Ward, A. 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.

[33] Standard performance evaluation corporation, `http://spec.org/`, accessed: 2023-10-17.

[34] Energy saving in large scale distributed platforms – Energumen, `https://anr.fr/Project-ANR-18-CE25-0008`, [Online; accessed 17-October-2023] (2018).

[35] H. Mann, D. Whitney, On a test of whether one of two random variables is stochastically larger than the other, The annals of mathematical statistics (1947) 50–60.

[36] D. Montgomery, G. Runger, Applied statistics and probability for engineers, John Wiley & Sons, 2020, p. 243.

[37] P. Mathews, Sample size calculations: Practical methods for engineers and scientists, Mathews Malnar and Bailey, 2010, p. 4.

[38] D. Klusáček, Š. Tóth, G. Podolníková, Real-life experience with major reconfiguration of job scheduling system, in: Job Scheduling Strategies for Parallel Processing: 19th and 20th International Workshops, JSSPP 2015, Hyderabad, India, May 26, 2015 and JSSPP 2016, Chicago, IL, USA, May 27, 2016, Revised Selected Papers 19, Springer, 2017, pp. 83–101.

[39] The metacentrum 2 log, `https://www.cs.huji.ac.il/labs/parallel/workload/l_metacentrum2/index.html`, accessed: 2023-10-26.

[40] J. Wilkes, More Google cluster data, Google research blog, posted at `http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html`. (Nov. 2011).

[41] J. Wilkes, Yet more Google compute cluster trace data, Google research blog, posted at `https://ai.googleblog.com/2020/04/yet-more-google-compute-cluster-trace.html`. (Apr. 2020).

[42] `https://github.com/alibaba/clusterdata/`, accessed: 2024-01-30 (2017).

[43] `https://github.com/alibaba/clusterdata/`, accessed: 2024-01-30 (2018).

[44] Hpc and htc tutorial, `https://www.grid5000.fr/w/HPC_and_HTC_tutorial`, accessed: 2023-11-23.

[45] S. Di, D. Kondo, W. Cirne, Characterization and comparison of cloud versus grid workloads, in: 2012 IEEE International Conference on Cluster Computing, IEEE, 2012, pp. 230–238.