Characterization of Different User Behaviors for Demand Response in Data Centers

Maël Madon, Georges da Costa, Jean-Marc Pierson

Characterization of different user behaviors for demand response in data centers

Maël Madon[0000−0001−9476−4682], Georges Da Costa[0000−0002−3365−7709], and Jean-Marc Pierson[0000−0001−8948−0474]

IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, Toulouse, France
{mael.madon,georges.da-costa,jean-marc.pierson}@irit.fr

Abstract. Digital technologies are becoming ubiquitous while their impact increases. A growing part of this impact happens far away from the end users, in networks or data centers, contributing to a rebound effect. A solution for a more responsible use is therefore to involve the user. As a first step in this quest, this work considers the users of a data center and characterizes their contribution to curtail the computing load for a short period of time by solely changing their job submission behavior. The contributions are: (i) an open-source plugin for the simulator Batsim to simulate users based on real data; (ii) the exploration of four types of user behaviors to curtail the load during a time window, namely *delaying*, *degrading*, *reconfiguring* or *renouncing* their job submissions. We study the impact of these behaviors on four different metrics: the energy consumed during and after the time window, the mean waiting time and the mean slowdown. We also characterize the conditions under which the involvement of users is the most beneficial.

Keywords: Demand response · User involvement · User-aware · Reproducible research · Parallel workload · Data center

1 Introduction

Digital technologies are increasingly contributing to global warming, for instance through mining of their components, transport along their supply chains or electricity consumed during their use phase. A recent review of estimates [6] puts this impact at 1.0–1.7 $GtCO_2e$ in 2020, i.e., 1.8%-2.8% of global greenhouse gas emissions. The authors also argue that although progress in energy efficiency of these technologies will probably continue, it will likely be outbalanced by growth in usage, leading to an overall increase of the carbon footprint. This so-called "rebound effect" seems difficult to fight within our research area (scheduling and distributed computing) where the focus is on energy optimization that must be effortless to end-users. On the contrary, we argue that users of digital technologies must be brought back into the loop, made aware of their impact and empowered to mitigate it.

Involving the user for environmental-aware scheduling in data centers has two aspects. One is to consider user *requests* for more environment-friendly services (e.g., guarantees, green labels) and try to achieve them. The other is to

consider the users as a *lever* for flexibility in the scheduling, i.e., they accept to compromise occasionally on their quality of service to allow some optimizations. The degradation can be spatial [9] (reducing the amount of resources allocated for the jobs), temporal [16] (delaying their execution) or both [8].

This paper proposes an experimental analysis of such user levers in a context of demand response management by investigating the following question: from the users' perspective, what is the room for maneuver to curtail the load on the data center for a short period of time?

The rest of the article is organized as follows. We start by giving some background and discuss related works in Section 2. Section 3 presents our data center model and lists the user behaviors studied for demand response. Section 4 describes the experimental setup for characterizing these behaviors. The results are presented in Section 5 while Section 6 provides a discussion on the results and the limitations of the study. Finally, we conclude in Section 7 and provide perspectives for future works.

2 Background and related works

Context: demand response Data centers are viewed as good candidates to participate in demand response programs [17]. Large consumers of electricity, they also have a more flexible load than other industrial facilities. Demand response consists of adapting the electricity *consumption* in response to the availability of *production*. For example, some electricity markets have Coincident Peak Pricing programs, where industrial consumers are charged a very high price during the time window when the most electricity is requested overall in the grid. These peak pricing events last typically 15 minutes [18] or one hour [14] but are only known *afterwards*, e.g., at the end of the month. The electricity supplier would only send warnings to the consumer that a peak load event may happen in the next few hours.

Involving the users Among the large body of work on energy-aware scheduling in data centers, some authors have studied strategies involving the users. Some works aim at providing guarantees to their users ("green offers" [7], "green SLA" [10,1]) and commit to fulfilling them by classical methods (self-supply of renewable energy [10], geo-distributed data centers with variable PUE and energy mix [7,1]). More related to this paper, some works study user flexibility as a *lever* for energy efficiency. For example, Guyon et al. [9] give to the users the choice between three execution modes (big, medium, little) for their jobs. Small execution modes request fewer resources but take longer to complete. They achieve gains through spatial consolidation with a bin-packing algorithm. Orgerie et al. [16] save energy through thermal-aware scheduling and smart resource switch off by letting the users choose between different submission times on the basis of energy consumption estimations for each of the alternatives. A combination of both spatial and thermal consolidation is proposed in another work by Guyon et al. [8] or in the All4Green project [2], where user involvement is

leveraged through contracts between the energy supplier, the data center and the user. The latter work, also in a context of demand response, is the closest to our approach. However, it integrates demand response mechanisms affecting the user with mechanisms transparent to them (use of batteries, precooling, geographical workload migration) so much so that the contribution of each user behavior to the final results is difficult to identify.

The originality of our work is to focus on the user behaviors, which allows providing a characterization of them. To the best of our knowledge, we are also the only ones to consider the behavior of simply *renouncing* job submissions. It is a radical behavior, but to be considered in a sufficiency approach.

3 Model

In our model, a demand response event will be represented by a time window of few hours (called "demand response window") during which the objective is to reduce electricity consumption. The event is supposed unknown in advance. In order to characterize the efficiency of different user behaviors to react to such demand response event, we consider a data center to which users can submit their jobs. At the interface between the two is the RJMS (Resource and Job Management System), the scheduler in charge of job placement and resource management. In this section, we describe the different components of our system.

3.1 Data center

In the data center, we only take into account the energy consumption of the multicore homogeneous machines. The power of a machine is P_{off}, P_{son} or P_{soff} if the machine is switched off, switching on or switching off, respectively. When a machine is switched on, its power is equal to $P_{idle} + N * P_{core}$ with P_{idle} the power drawn by an idle machine, N the number of cores in use (i.e., with a job running on it) and P_{core} the power drawn by each core.

A job is completely defined by its *submission time*, *execution time* and number of requested cores that we denote by *size* in the rest of this paper. The scheduler decides the starting time for the job and the machine it will be executed on. Note that the scheduler in our model only execute jobs on single machines. We suppose perfect communication without latency.

3.2 Scheduler

The scheduler is a bin-packing scheduler with shutdown (same as Guyon et al. [8,9]). It is a greedy algorithm trying to schedule ("pack") all the jobs in the least possible machines and shut down idle machines. To do so, it maintains and updates two data structures: a queue of waiting jobs and a list of switched-on machines. The queue of jobs is sorted by *decreasing* size order – and by increasing submission time (first come, first served) in case of a tie. The list of machines is sorted by *increasing* order of available cores. Every time one (or more) job

is submitted or finishes, the scheduler goes through the job queue in order and tries to find for each job the smallest machine where it fits. If no machine is found that way, a new machine (if any) is powered on and the job is scheduled on this machine. After that, we immediately shut down all idle machines.

3.3 Users

During the demand response window, users are asked to make an effort to curtail the load in the data center. They do so by adopting different behaviors described below and illustrated in Fig. 1.

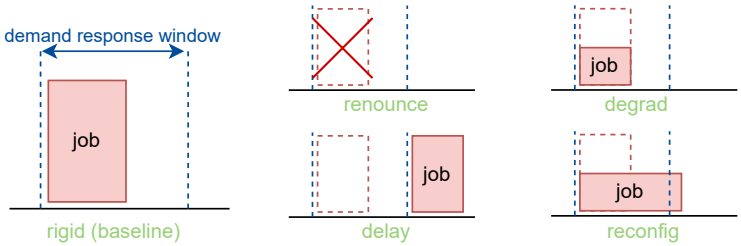


Fig. 1. The five user behaviors studied

- **rigid**: replay jobs as in the original workload; Baseline for comparison.
- **renounce**: do not submit jobs originally submitted during the window.
- **delay**: delay all job submissions to the end of the window.
- **degrad**: divide the size of the jobs by two, rounded up. The execution time stays the same. Note that the rounding implies that when only one core is requested for a job, the job remains unchanged.
- **reconfig**: also divide the size by two, rounded up, but increase the execution time to match the original computing mass. We make the hypothesis of perfect speedup, i.e., a job executing on one core completes in exactly twice the time than on two cores.

4 Experimental setup

4.1 Software used for simulation

To simulate our system, we use Batsim [4], an open-source infrastructure and resource management system simulator[1] based on SimGrid[2]. We implemented

[1] Batsim: https://batsim.org/
[2] SimGrid: https://simgrid.org with the energy plugin https://simgrid.org/doc/latest/Plugins.html?highlight=energy#host-energy

the bin-packing scheduler for Batsim in C++. We also developed a plugin called "batmen" to interact with simulated users and receive their job submissions. For the purpose of this study, users replay an input workload trace except in the demand response window, where they act according to their behavior. We therefore implemented five user classes corresponding to the five behaviors of Fig. 1. Our code is open source[3]. With this simulation tool, we designed and conducted an experimental campaign, whose main details are given below. All scripts are available to reproduce and analyze our results, either in our gitlab[4] or in the Figshare repository [15].

4.2 Workload

We replay a real public workload trace containing the information about the submitting user for each recorded job. We chose the 2-year trace from MetaCentrum (national grid of the Czech Republic), available in the Parallel Workload Archive[5]. The platform is very heterogeneous and underwent majors changes during the logging period [12]. For the purpose of our study, we perform the following selection:

1. We truncate the workload to keep only 6 months (June to November 2014) where no major change was performed in the infrastructure, and we remove all the clusters whose nodes have more than 16 cores;
2. From this truncated workload, we remove all jobs with an execution time greater than one day and all jobs with a size greater than 16. It leaves us with a workload manageable with machines of a usual size, and without more than one day of inertia.

4.3 Platform

The first selection step keeps a total of 6304 cores. The second selection step excludes 2.7% of jobs from the truncated workload, representing 73.7% of the mass (in core-hour). Consequently, we create a simulated platform adapted to this load with $6304 * (1 - 0.737)/16 =$ **104 homogeneous 16-core machines**. Power constants ($P_{idle} = 100W$, $P_{core} = 7.3125W$, $P_{off} = 9.75$, $P_{son} = 100W$ and $P_{soff} = 125W$) for the servers and time to switch on ($T_{son} = 150s$) and switch off ($T_{son} = 6s$) are measurements in Taurus Grid'5000 cluster from existing work [9].

4.4 Experimental campaign

For the evaluation, we consider the following scenario. We imagine a data center functioning at nominal load: some jobs are currently running and users can

[3] batmen repository: https://gitlab.irit.fr/sepia-pub/mael/batmen
[4] experiments repository: https://gitlab.irit.fr/sepia-pub/open-science/demand-response-user/-/tree/europar2022
[5] METACENTRUM-2013-3.swf available at https://www.cs.huji.ac.il/labs/parallel/workload/l_metacentrum2/index.html

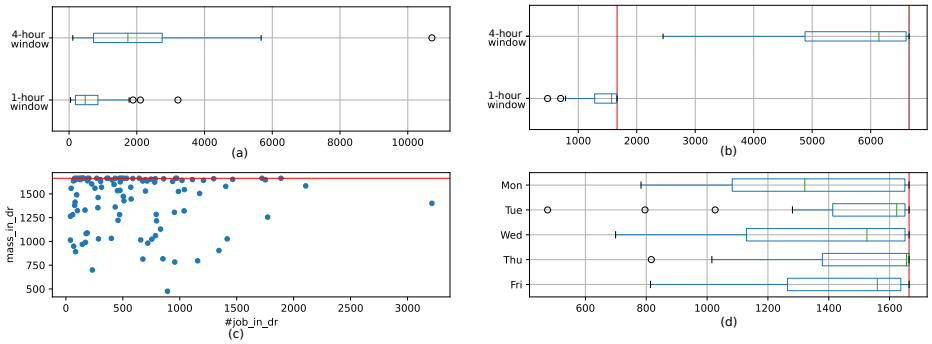


Fig. 2. Descriptive statistics for the 105 experiments. Red lines corresponds to the infrastructure (1664 cores). (a) number of jobs submitted in window; (b) computing mass (in core-hour) in window; (c) computing mass in window by number of submitted jobs (1-hour window); (d) computing mass in window by weekday (1-hour window)

submit new jobs to the scheduler. Suddenly, the electrical grid sends an alert to warn the data center manager that a consumption peak is detected in the grid. The manager forwards this alert to his users who adapt their submission behavior. At the end of the alert, the users return to a normal behavior (after submitting all their delayed jobs, if any).

We conduct an experimental campaign on 105 different input data (the number of weekdays between Jun 1, 2014 and Oct 23, 2014). For each input data, we simulate the aforementioned scenario, assuming that all users adopt the same behavior during the demand response window. On three days of data center operation, we make the demand response event arise at 16:00 on day 2, chosen to be a weekday. This choice is justified by a characterization of 26 years' coincident peak pricing data [14], given that the MetaCentrum trace also displays diurnal and weekday/weekend patterns. We study two lengths for the demand response window: one and four hours. We also tried other starting times (drawn at random) and other window lengths (0.5 and 2 hours) but decided not to report their results here as they are not leading to different conclusions.

The simulation starts one day before the event and stops one day after, to ensure that the infrastructure runs at nominal load on day 2 and has absorbed the event by the end of day 3 (the selected jobs in the workload having an execution time lower than one day). In total, we launch nine simulations per experiment (= input data): the baseline simulation with all users having a "rigid" behavior, and the four other behaviors on the two window lengths. Descriptive statistics on the experiments are displayed in Fig. 2.

The campaign launched in parallel on a 2 x 8-core Intel Xeon E5-2630 v3 machine finished in less than two hours. Launched in France, and ran 2 times in total, this has a carbon footprint of around 50 g CO2e (calculated using https://green-algorithms.org v2.1 [13]).

5 Results

5.1 Energy metrics

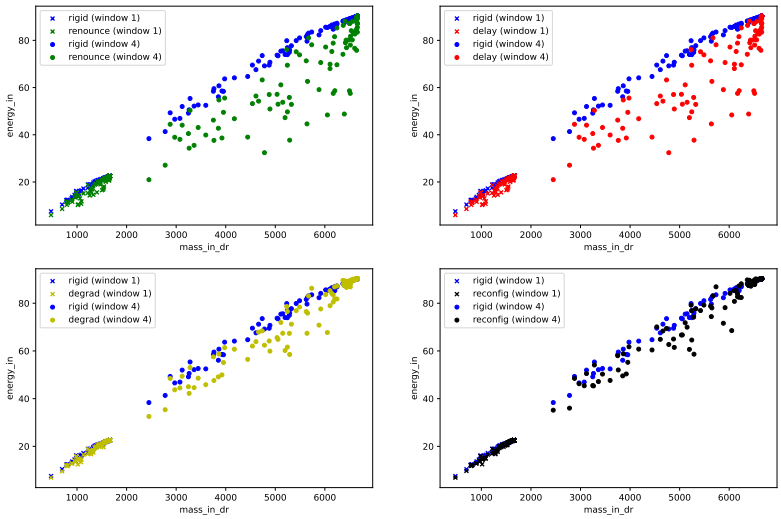
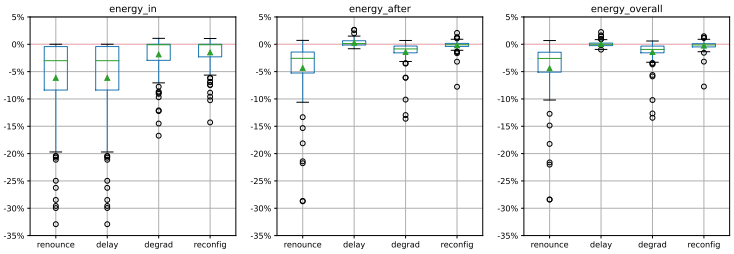


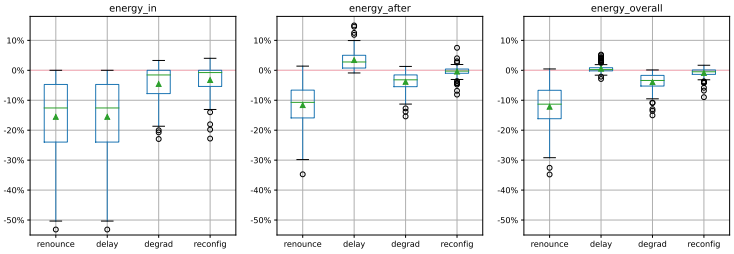
Fig. 3. Energy consumed in each simulation. Y-axis: energy consumed (in kWh) during the demand response window. X-axis: computing mass (in core-hour) during the demand response window for the baseline behavior.

We recall our research question: by intervening only on the user's side, what energy gains can be expected by adapting one's behavior for a few hours? Fig. 3 displays the **energy consumed during the demand response window** for every experiment and every behavior. Values are scattered by the total load of the infrastructure during the window for the baseline behavior. For that behavior, we note an almost linear relationship between infrastructure load and consumed energy. Deviations from the linear line are due to situations favoring a more or less good packing from the scheduler inside the 16-core machines. Behaviors "renounce" and "delay" perform identically for this metric: users of both behaviors stop submitting inside the demand response window, resulting in a lower energy consumption compared to the baseline. This gain is the best we can expect. Behaviors "degrad" and "reconfig" display similar results. In addition, one would expect a positive correlation between the load of the platform and the relative energy gains of the four behaviors compared to the baseline. It would translate into an increasing distance between the colored dots and the blue dots in the graphs, as the load increases. Counter-intuitively, this does not seem to happen.

The experimental campaign showing very scattered results, Fig. 4 displays the relative energy gains for each experiment as box plots. We can read for



(a) 1-hour demand response window



(b) 4-hour demand response window

Fig. 4. Energy metrics per behavior relatively to the baseline behavior. The green triangle in the box plots indicates the mean.

example that "renounce", the most radical behavior, allows energy savings of up to 33% in the window for a one-hour window, and 53% for a four-hour window. The savings do not go up to 100% because jobs that were already there before the window are still running in the infrastructure, which consumes energy.

In addition to the energy consumed *within* the window, Fig. 4 shows the impact of the different behaviors on the energy consumed *after* the demand response event, i.e., from 17:00 or 20:00 on day2 (depending on the window length) to 24:00 on day3. For this second metric, "delay" performs very differently compared to "renounce". All the jobs within the window get postponed, resulting in an extra power consumption at the end of the window: +0.3% (resp. +3.4%) on average for a 1-hour (resp. 4-hour) window. This behavior remains neutral with respect to overall energy consumption (*within* + *after* the window).The behavior "reconfig", which also keeps a constant mass of jobs compared to the baseline, allows some optimizations. Up to 10% overall energy consumption could be saved because the reconfigured jobs "fit better in the holes" left by the available cores in the switched on machines. "Degrad" performs unsurprisingly better in all respects, the users having accepted to reduce the mass of job submitted.

Finally, we notice that **the bigger the window, the better the energy gains**. This is due to inertia of the system: with a longer window, a behavior on the submitted jobs has more time to make a difference compared to the residual jobs that are still running in the infrastructure.

5.2 Perceived impact on the scheduling

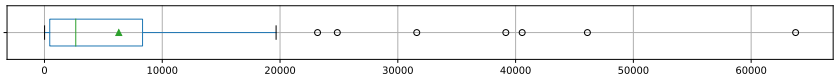
We use two usual metrics: mean waiting time and mean slowdown. Waiting time is the time a user has to wait until her job starts running:

$waiting_{time} = starting_{time} - submission_{time}$

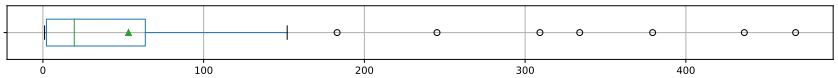
Slowdown divides this extended completion time by the execution time:

$slowdown = (finish_{time} - submission_{time})/execution_{time}$.

For each experiment, we take the average waiting time (resp. slowdown) on all jobs submitted between the beginning of the demand response window and the end of the experiment (same period as metric $energy\_in + energy\_after$). Fig. 5 shows these results for the "rigid" behavior. We observe that for half of the experiments, the mean waiting time is below one hour (3600s) and the mean slowdown below 25. These are experiments with an unsaturated infrastructure and an often empty queue of waiting job. On the other hand, there are also cases of high congestion (e.g., the seven outliers at more than 6h mean waiting time).



(a) Mean waiting time (in seconds)

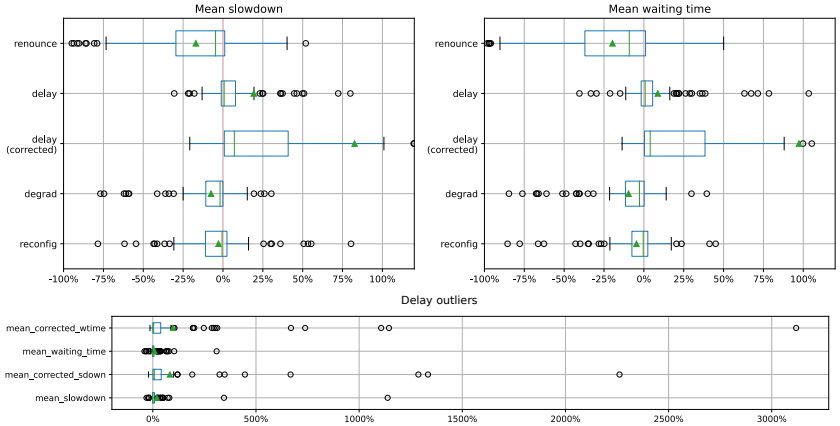


(b) Mean slowdown (dimensionless)

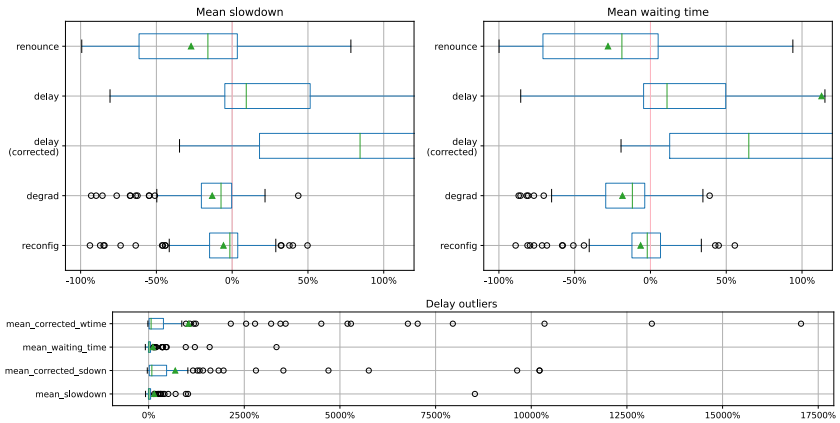
Fig. 5. Scheduling metric distribution for the 105 experiments, baseline behavior.

The results for the other behaviors are plotted in Fig. 6, as a percentage of gain/loss compared to the baseline. Specifically for the behavior "delay", we provide both *corrected* and *uncorrected* metrics. The uncorrected slowdown and waiting time are calculated in relation to the *new* (delayed) submission times, while the corrected ones use the *original* submission times (from the baseline). Note also that for the behavior "renounce" some jobs have been canceled, thus the mean waiting time and slowdown is calculated on a subset of the jobs compared to the other behaviors. From Fig. 6 it can be observed that the behaviors "renounce", "degrad" and "reconfig" (in this order) affect the scheduling positively on average. This is not surprising, as the first two behaviors reduce the total mass of jobs to compute, and the third allows a better packing. Yet, the scheduling gets worsened in a significant number of cases (around 50% for "reconfig" and 25% for "degrad" and "renounce"), due to bad choices of the scheduler.

The behavior "delay" stands out from the others as it affects the scheduling negatively in most cases, even for the uncorrected metrics. It gets even worse when including the extra waiting time from the delayed job in the calculation of the corrected metrics. In fact, it is preferable in terms of waiting time and slowdown that the job submissions are spread out throughout the time.



(a) 1-hour demand response window



(b) 4-hour demand response window

Fig. 6. Scheduling metrics per behavior relatively to the baseline behavior

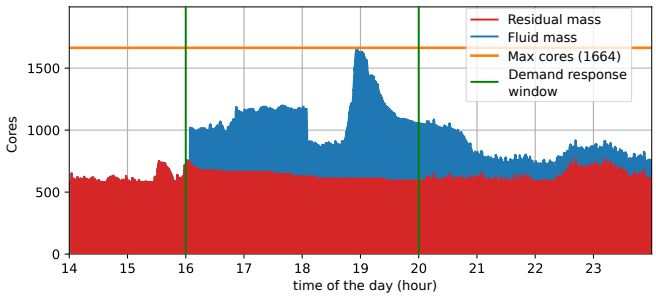


Fig. 7. Example of fluid and residual mass. *(Thursday Jun 26 2014)*

6 Discussion

6.1 The fluid-residual ratio: an explanation of the results

As seen previously in Fig. 3, the achievable energy savings in the demand response window cannot be explained by the infrastructure load during that window. In fact, it is possible that the load is very high because of a large mass of job submitted *before* the window, although the load on which the users have an influence is the mass submitted *during* the window. We call these two quantities the **residual mass**, submitted outside the window, and the **fluid mass**, submitted inside the window (Fig. 7).

Users, by accepting to "renounce" or "delay" their jobs, allow cutting the energy consumption due to the fluid mass, which is roughly proportional to the mass itself, as we saw before. In other terms, **the gains during the window are at most equal to the proportion of fluid mass in that window**. This is exactly what we see in Fig. 8 displaying the energy gains as a function of the fluid-residual ratio. The red line indicates the best possible gains, which are almost achieved by "renounce" and "delay" behavior (the non-linearity of the energy model explaining the gap).

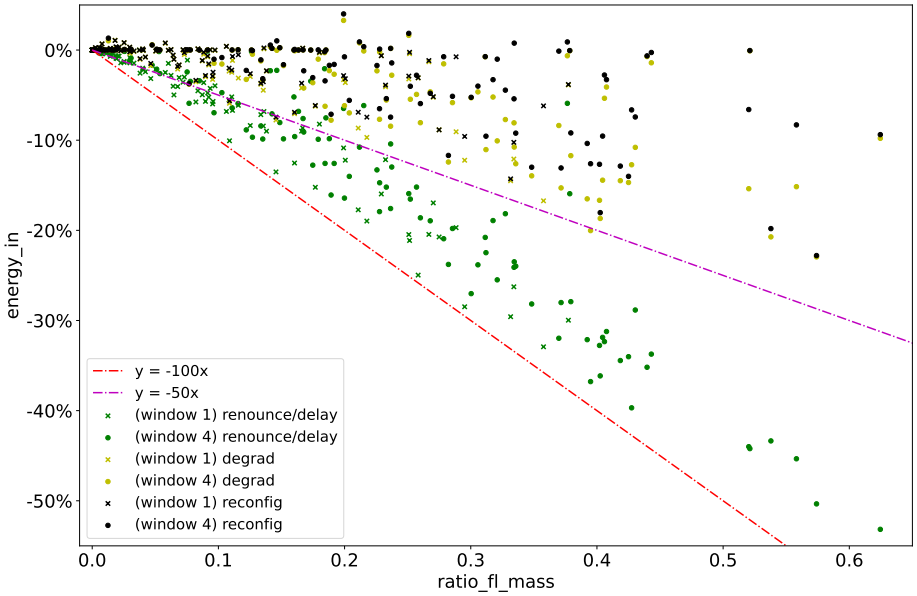


Fig. 8. Energy gains in function of the fluid-residual ratio. Only one plot for the behaviors "renounce" and "delay" because they are identical for this metric.

In some cases, however, these behaviors don't realize that gain: they are cases of **saturation**, when many jobs are waiting in the queue. The removal of the fluid mass is compensated by the execution of the awaiting residual mass.

For the "degrad" behavior, gains are expected to be of half the fluid mass at most, as users divide their submitted mass by two. In practice, the results are even more scattered and further away from their optimal (magenta line). This is partially due to the saturation effect, but also to rounding (e.g., a job with an original size of 3 will be submitted with size 2 and a job with size 1 remains unchanged) and imperfect packing. The analysis for the behavior "reconfig" is similar, with even less expected gains. Some experiments even make negative gains: they are due to the greedy and non-clairvoyant scheduler taking bad decisions for the future, like switching off a machine just before the submission of new jobs.

6.2 Pros and cons of each behavior

behavior	energy in	energy overall	scheduling metrics	acceptability
renounce	1st	1st	1st	4th
delay	1st	4th	4th	2nd
degrad	3rd	2nd	2nd	3rd
reconfig	4th	3rd	3rd	1st

Table 1. Summary ranking of the four behaviors according to their impact on energy consumption and scheduling metrics. The column "acceptability" is opinion-based, it reflects the size of the effort asked from the user.

Building upon what we learned from the results, we provide a summary discussion on the characteristics of each behavior (see Table 1). To begin with, the behavior **renounce** performs the best for all the metrics studied. We saw that it actually reaches the optimal energy gains during the window for unsaturated cases. This rank is not surprising considering the sacrifice required from the user. Yet, we think that such a behavior is often overlooked in similar studies and argue that environmentally aware users or users provided with a proper incentive would do it. Moreover, some jobs running in data centers today might not be indispensable.

On the other end of the spectrum, the behavior **reconfig** seems to be the most acceptable to the users, as it does not decrease the mass initially submitted and provides better waiting time and slowdown than "delay" for both the jobs within and after the window. "Reconfig" is a good trade-off to achieve some optimizations with a low effort from the user, especially in combination with bin-packing schedulers and on/off policies (see [9]).

Delay also keeps the mass constant, which ranks it second behind "reconfig" in terms of acceptability. Same as "renounce", it reaches the optimal energy gains during the window. However, it introduces an overhead in overall energy

consumption and slowdown compared to the baseline behavior. Note that this overhead would probably be less important in real life due to users adapting their behavior if they experience congestion in the infrastructure. This is the limit of blindly replaying past workload traces in simulations, as pointed by Feitelson [5].

Finally, the behavior **degrad** ranks second or third in all the categories of Table 1. It remains an interesting trade-off between simply renouncing a job and reconfiguring it at constant mass. Practical applications of this behavior are optional features in an application that can be cut off if needed (e.g., recommendations for e-commerce, alternative paths for mapping apps).

6.3 Interactions with scheduling systems

The work presented in this paper is rather theoretical and abstracts part of the reality of production systems. For example, in real-world schedulers, it is commun to have job priorities. In that case, the degradation of a low-priority job would be less costly to the user compared to a high-priority job. The priority of a job could be considered along with other criterias (e.g., magnitude of the delay, size of the degradation) to define a *utility* metric in an attempt to quantify the acceptability of a given behavior.

All in all, user submission behaviors remain one lever for energy saving among others. It has the particularity of having some latency, which makes it not optimal in a context of demand response without prediction. Therefore, **taking into account these behaviors inside the scheduler** seems essential to make the best of their potential and go beyond the fluid-residual limit. For example, by allowing the scheduler to kill jobs, checkpoint them [3], or to suspend the waiting queue. Decisions could be taken on behalf of the users, with a mecanism of contract with the data center operator specifying the degradation the user is willing to accept [2]. Nevertheless, it seems crucial for us to make these decisions transparent to the user and involve them as much as possible, as this appears as the main path towards a *sufficient* [11] use of our technologies.

6.4 Limitations

Model simplifications In our data center simulations, we do not take into account the latency and bottleneck effects in the communications. Also, we suppose perfect speedup in the model, i.e., a job executed on two cores will take exactly twice longer than the same job executed on four cores. Finally, we accounted only for the energy consumption of the CPUs, and neglected others like memory, network or cooling. Hopefully, the powerful simulation tools that we use (Batsim and SimGrid) will help us to overcome these simplifications in future works.

Methodological limitations We see three major threats to the validity of our method to answer the research question. First, we study only one scheduler (bin-packing) while results with other common schedulers (FCFS, easy-backfilling...) would have been of interest. Second, we use only one input trace (MetaCentrum)

which comes from a research infrastructure and not a production cloud, and we perform a selection from it (see 4.2) that might make us miss the big picture. Finally, our study includes all the limitations related to the use of a simulation, especially when dealing with human behaviors which are unpredictable.

7 Conclusion and future works

In this paper, we study four different ways for a user of a data center to curtail her load for a certain period of time by changing submission behavior. These behaviors are delaying, degrading, reconfiguring or renouncing the jobs during the time period. We show experimentally through simulation on real world data that these behaviors have a certain latency for decreasing the load on the infrastructure. Indeed, they cannot decrease the load due to jobs that are already running on the infrastructure. Therefore, we define two quantities, the *fluid* and *residual* mass, and discuss the experimental results according to the ratio of these two quantities. We also discuss the pros and the cons of each behavior in the light of their energy saving potential, impact on scheduling and acceptability to the user. We hope that this work will pave the way for studies involving the user more intensely.

Future work will focus on (i) improving the data center model to deal with the model simplifications listed in Subsection 6.4, (ii) proposing schedulers capable of leveraging the efforts made by the user (e.g., through "green SLA"), (iii) elaborating on the user model to more realistically account for submission patterns and response to feedback from the infrastructure (as proposed by Feitelson [5]) and (iv) going beyond the limited scope of demand response to reason on the sustainability of the infrastructure as a whole.

Acknowledgements and Data Availability Statement. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr). The scripts and instructions necessary to reproduce and analyze our result are available in a Figshare repository [15].

This work was partly supported by the French Research Agency under the project Energumen (ANR-18-CE25-0008) and DataZero2 (ANR-19-CE25-0016).

References

1. Amokrane, A., Langar, R., Zhani, M.F., Boutaba, R., Pujolle, G.: Greenslater: On Satisfying Green SLAs in Distributed Clouds. IEEE Transactions on Network and Service Management **12**(3), 363–376 (Sep 2015). https://doi.org/10.1109/TNSM.2015.2440423
2. Basmadjian, R., Botero, J.F., Giuliani, G., Hesselbach, X., Klingert, S., De Meer, H.: Making Data Centers Fit for Demand Response: Introducing GreenSDA and GreenSLA Contracts. IEEE Transactions on Smart Grid **9**(4), 3453–3464 (Jul 2018). https://doi.org/10.1109/TSG.2016.2632526

3. Dupont, B., Mejri, N., Da Costa, G.: Energy-aware scheduling of malleable HPC applications using a Particle Swarm optimised greedy algorithm. Sustainable Computing: Informatics and Systems **28**, 100447 (Dec 2020). https://doi.org/10.1016/j.suscom.2020.100447
4. Dutot, P.F., Mercier, M., Poquet, M., Richard, O.: Batsim: A Realistic Language-Independent Resources and Jobs Management Systems Simulator. In: 20th Workshop on Job Scheduling Strategies for Parallel Processing. Chicago, United States (May 2016). https://doi.org/10.1007/978-3-319-61756-5_10
5. Feitelson, D.G.: Resampling with Feedback — A New Paradigm of Using Workload Data for Performance Evaluation. In: Dutot, P.F., Trystram, D. (eds.) Euro-Par 2016: Parallel Processing. pp. 3–21. Lecture Notes in Computer Science, Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-43659-3_1
6. Freitag, C., Berners-Lee, M., Widdicks, K., Knowles, B., Blair, G.S., Friday, A.: The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations. Patterns **2**(9) (Sep 2021). https://doi.org/10.1016/j.patter.2021.100340
7. Garg, S.K., Yeo, C.S., Buyya, R.: Green Cloud Framework for Improving Carbon Efficiency of Clouds. In: Jeannot, E., Namyst, R., Roman, J. (eds.) Euro-Par 2011 Parallel Processing. pp. 491–502. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23400-2_45
8. Guyon, D., Orgerie, A.C., Morin, C.: Energy - Efficient IaaS-PaaS Co-Design for Flexible Cloud Deployment of Scientific Applications. In: 2018 30th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD). pp. 69–76 (Sep 2018). https://doi.org/10.1109/CAHPC.2018.8645888
9. Guyon, D., Orgerie, A.C., Morin, C., Agarwal, D.: Involving users in energy conservation: A case study in scientific clouds. International Journal of Grid and Utility Computing **10**(3), 272–282 (Jan 2019). https://doi.org/10.1504/IJGUC.2019.099667
10. Haque, M.E., Le, K., Goiri, Í., Bianchini, R., Nguyen, T.D.: Providing green SLAs in High Performance Computing clouds. In: 2013 International Green Computing Conference Proceedings. pp. 1–11 (Jun 2013). https://doi.org/10.1109/IGCC.2013.6604503
11. Hilty, L.: Computing Efficiency, Sufficiency, and Self-sufficiency: A Model for Sustainability? In: LIMITS 2015, First Workshop on Computing within Limits. s.n., Irvine, CA, USA (Jun 2015). https://doi.org/10.5167/uzh-110766
12. Klusáček, D., Tóth, Š., Podolníková, G.: Real-Life Experience with Major Reconfiguration of Job Scheduling System. In: Desai, N., Cirne, W. (eds.) Job Scheduling Strategies for Parallel Processing. pp. 83–101. Lecture Notes in Computer Science, Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-61756-5_5
13. Lannelongue, L., Grealey, J., Inouye, M.: Green Algorithms: Quantifying the Carbon Footprint of Computation. Advanced Science **8**(12), 2100707 (2021). https://doi.org/10.1002/advs.202100707
14. Liu, Z., Wierman, A., Chen, Y., Razon, B., Chen, N.: Data center demand response: Avoiding the coincident peak via workload shifting and local generation. Performance Evaluation **70**(10), 770–791 (Oct 2013). https://doi.org/10.1016/j.peva.2013.08.014
15. Madon, M., Da Costa, G., Pierson, J.M.: Artifact and instructions to generate experimental results for Euro-Par'2022 paper: Characterization of different user

behaviors for demand response in data centers (Jun 2022). https://doi.org/10.6084/m9.figshare.19948352
16. Orgerie, A., Lefèvre, L., Gelas, J.: Save Watts in Your Grid: Green Strategies for Energy-Aware Framework in Large Scale Distributed Systems. In: 2008 14th IEEE International Conference on Parallel and Distributed Systems. pp. 171–178 (Dec 2008). https://doi.org/10.1109/ICPADS.2008.97
17. Wierman, A., Liu, Z., Liu, I., Mohsenian-Rad, H.: Opportunities and challenges for data center demand response. In: International Green Computing Conference. pp. 1–10 (Nov 2014). https://doi.org/10.1109/IGCC.2014.7039172
18. Zarnikau, J., Thal, D.: The response of large industrial energy consumers to four coincident peak (4CP) transmission charges in the Texas (ERCOT) market. Utilities Policy **26**, 1–6 (Sep 2013). https://doi.org/10.1016/j.jup.2013.04.004