Towards Digital Sustainability: Involving Cloud Users as Key Players

Anas Mokhtari, Baptiste Jonglez, Thomas Ledoux

Towards Digital Sustainability: Involving Cloud Users as Key Players

Anas Mokhtari
IMT Atlantique, INRIA, LS2N, UMR CNRS 6004, F-44307
Nantes, France
anas.mokhtari@imt-atlantique.fr

Baptiste Jonglez
IMT Atlantique, INRIA, LS2N, UMR CNRS 6004, F-44307
Nantes, France
baptiste.jonglez@inria.fr

Thomas Ledoux
IMT Atlantique, INRIA, LS2N, UMR CNRS 6004, F-44307
Nantes, France
thomas.ledoux@imt-atlantique.fr

***Abstract*—Due to the rapid growth of Cloud services, data centers have become major energy consumers, resulting in significant CO_2 emissions. Several infrastructure-focused strategies, such as resource consolidation, have been used to reduce the carbon footprint of Cloud infrastructure. However, end-users are often left out of the picture. Since they are the primary target of Cloud applications, it would be beneficial to actively involve them in reducing the carbon footprint of Cloud applications.**

In this paper, we offer end-users a way to influence the carbon footprint of Cloud applications they use. To this end, we ask end-users to select a high-level mode to control the carbon footprint of a Cloud application. We then design a dynamic adaptation algorithm that determines an appropriate configuration for the application for each request, based on the end-user mode and on the carbon intensity of the infrastructure energy sources.

We implement and evaluate our system on a simple image-resizing application. We run experiments on SeDuCe, a Cloud infrastructure testbed partially powered by solar panels. Our results show that we save energy consumption by up to 84% when all end-users agree to degrade the quality of the application's output, and we provide a good quality-energy trade-off when end-users make heterogeneous choices. In addition, we are able to improve quality by leveraging the available green energy budget.

***Index Terms*—cloud computing, human-centered computing, digital sustainability, carbon footprint, green energy**

I. Introduction

Digital services and infrastructure account for a significant share of global greenhouse gas emissions. This carbon footprint is set to increase further as the use of new technologies is growing while efficiency improvements are expected to slow down [1]. Several research studies aim to reduce the carbon footprint of the IT sector by optimizing Cloud infrastructure: reducing their energy consumption [2], using renewable energy sources [3], or consolidating workloads on fewer physical servers [4]. However, most of these studies consider end-user behavior as a fixed input parameter, *i.e.,* they assume that users generate a workload that must then be managed by the infrastructure. Similarly, Cloud (SaaS) applications running on the infrastructure are often considered *black-box* systems that simply consume resources such as CPU, memory or disk.

This research was supported by the French project OTPaaS.

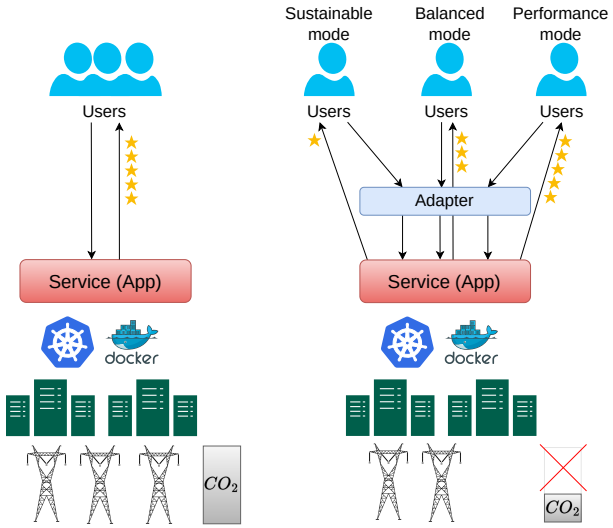
We believe that actively involving end-users is beneficial and can help to reduce the overall carbon footprint of digital services. By "end-user", we refer to the final user accessing a given application running on a Cloud infrastructure. Our approach is based on the realization that end-users actually have flexible needs: they may have different requirements regarding the level of quality offered by the hosted application, such as the number of results of a search query, the quality of an image, the response time to a query, etc. In addition, we are witnessing a growing awareness of digital sustainability: end-users may be ready to accept a further degradation of the quality if they are informed of a difficult situation, such as when the infrastructure is overloaded or when renewable energy is not available.

In this work, we go one step further: we enable end-users to play an active role in the energy consumption of their use of the software. We allow end-users to choose a high-level carbon footprint-related "mode of operation" for the SaaS application they are using. This mode is then used to determine an appropriate quality-energy trade-off for the application, given the current state of the infrastructure. This relies on a level of flexibility offered by Cloud applications: we ask developers to add parameters to the application, so that our system can dynamically configure the application to prioritize either high quality or low energy consumption.

The structure of this paper is as follows: Section II outlines our approach, while Section III illustrates the approach with a concrete use-case. Section IV introduces our adaptation algorithm. Experimental validation of our approach is detailed in Section V, and Section VI highlights the current scope of the work and associated challenges. Section VII reviews related work, and Section VIII concludes with our findings and future work.

II. Overview

In this section, we present our approach for reducing the application's carbon footprint through user involvement. Many SaaS applications follow the same pattern: a service – such as an e-commerce website or a map service – is generally



(a) Usual operation.

(b) Carbon footprint-aware operation through user involvement.

Fig. 1: Involving or not Cloud users.

accessed via a web browser. The browser sends user requests to the application, which is hosted on a server in a datacenter. Traditionally, all requests are processed in the same way for any user, regardless of the energy context of the underlying infrastructure (see Figure 1a). As a result, all users benefit from the same quality of service.

The idea of our approach is to enable each individual user to control their carbon footprint when using a digital service. We then want the service to adapt each response according to each user's choice, for instance by returning more or fewer results or by adjusting the output quality of the response. In practice, the choice of the user is expressed in a web browser extension that offers several *carbon footprint-related modes*. The selected mode is then transmitted to the Cloud application through an HTTP header.

We currently offer users three different modes (Figure 1b):

- **Performance mode** is chosen by a user when they want the best quality, regardless of the resulting carbon footprint. This mode corresponds to the operation of standard SaaS applications;
- **Sustainable mode** is chosen by a user when they wish to use the service with minimum carbon impact. In this case, the quality of the responses may be degraded, but in a controlled manner;
- **Balanced mode** is a balance between the other two modes: the user wants the best quality with the smallest possible carbon footprint. In this case, we have to find a trade-off between the quality and the carbon footprint in the current energy context. The "energy context" represents the current carbon intensity of each energy source used to power the datacenter infrastructure.

To make our approach applicable to a large class of Cloud applications, we introduce an *adaptation layer* that intercepts each request, looks at the HTTP header indicating the mode selected by the user, looks at the current energy context, and uses an *adaptation algorithm* to dynamically determine which application-specific parameters should be added to the request. Finally, the adaptation layer forwards the modified request to the application itself, and then forwards the response from the application back to the user (see Figure 1b).

To enable our adaptation layer to select good energy-quality trade-offs, the Cloud application must be *flexible* enough. We ask application developers to introduce *per-request parameters* in their application: the value of the parameters should influence the energy consumption for processing the request, as well as the quality of the response. We call each possible combination of parameters a *configuration*. We give an example of such a flexible application in the next section.

III. Example Application

We consider an image-resizing service that creates low-resolution images based on a larger original image (thumbnails or previews). Typically, a *resampling filter* is applied to the output image to improve its visual quality. This kind of service is CPU-intensive: the larger the output image size, the more processing is required; in addition, each resampling filter algorithm has a different processing cost. We select five values for the resolution parameter (144, 360, 480, 720 and 1080 pixels) and four techniques for the filter parameter (*Nearest*, *Bilinear*, *Bicubic* and *Lanczos*) [5], which gives 20 different configurations.

Each configuration requires a different amount of energy to resize an image. The configuration $(360, Nearest)$, which resizes an image to 360 pixels using the *Nearest* filter, consumes 75 mJ for each image. However, the configuration $(1080, Lanczos)$ requires almost ten times as much energy with 741 mJ. We describe the measurement methodology in Section V-A and the full results in Section V-B.

To assess the quality of each configuration, we rely on the visual perception of users to evaluate the output image quality. We define five levels of output quality: *Excellent* (⋆⋆⋆⋆⋆), *Very good* (⋆⋆⋆⋆), *Good* (⋆⋆⋆), *Medium* (⋆⋆) and *Poor* (⋆). For the evaluation in Section V, we classified our 20 configurations by visually inspecting example output images: the resulting quality levels are shown in Figure 4. We discuss this methodology in more details in Section VI.

The Python snippets in Listing 1 and 2 are example implementations of this image-resizing application in Python, using the *Flask* web framework. The first version of this application, in Listing 1, is not flexible: it simply reduces the input image to a fixed size.

To make it more flexible, we define two configuration parameters: the output image height (`height`) and the resampling filter (`r_filter`) to be applied to the resized image. The modified code is shown in Listing 2. The developer has updated the resize function to add the two new arguments. Most importantly, the developer has specified the possible values for each parameter and exposed them to the adapter using the *Decorator pattern* [6]. The decorator function can

extract the parameters as well as the acceptable values, and feed them as input to our adapter.

Any other software engineering method can be used to parameterize the application. The sole constraint is that our adapter must be configured with a list of parameters as well as a set of acceptable values for each parameter.

Listing 1 Code before modification

```
app = Flask(__name__)
@app.route('/resize/')
def resize_img(image):
    height = 480
    width = 480
    size = (width, height)
    image.thumbnail(size)
    ...
    return image
```

Listing 2 Code after modification

```
app = Flask(__name__)
@param('height', [144, 360, 480, 720,
 ↪  1080])
@param('r_filter', [0, 1, 2, 3])
@app.route('/resize/<int:height>/
 ↪  <int:r_filter>')
def resize_img(image, height, r_filter):
    ratio = (height / image.height)
    width = int(image.width * ratio)
    size = (width, height)
    image.thumbnail(size, r_filter)
    ...
    return image
```

IV. Adaptation Algorithm

In this section, we describe our adaptation algorithm. We consider the case of an infrastructure powered by two different types of electrical energy source:

- Power grid: considered to be an "unlimited" source of electrical power, but with a potentially significant carbon footprint;
- Solar panels: their maximum power is limited and variable. However, they provide low-carbon electricity.

This infrastructure is powered solely by solar panels as long as they can meet the energy needs. If solar energy is insufficient, the power supply system switches to hybrid mode, adding the grid as a complementary energy source.

For Performance mode users, reducing the carbon footprint is not a priority: their workload will be processed using power from the grid (i.e., brown energy). It means that we can devote all the available green power to processing the workload of the other two modes (Sustainable and Balanced). Within this green power budget, we would like to maximize the quality of responses for sustainable and balanced users.

To do so, we have to find the configuration of parameters to assign to each user mode according to three different inputs: the workload of user requests, its distribution according to the modes chosen by these users, and the green power available. These inputs are variable over time, and we therefore need a dynamic adaptation algorithm.

At each time step, solar panels deliver a power of P_s watts for the SaaS service. Our algorithm (See Algorithm 1) consists of four main steps, shown in Figure 2:

1) For sustainable mode workload, find the parameter configuration that minimizes the output quality (Quality of Experience, *QoE*). It will consume P_{sust} watts (Lines 1 and 2);
2) For balanced mode workload, find the parameter configuration that maximizes quality using only the remaining $(P_s - P_{sust})$ power. If this remaining power is insufficient for any configuration, take the one that minimizes quality. It will consume P_{bal} watts (Lines 3-6);
3) Return to sustainable mode to search for a new parameter configuration that maximizes the quality using only the remaining $(P_s - P_{bal})$ power. If no other configuration satisfies the power condition, keep the configuration from Step 1 (Line 7);
4) For performance mode workload, select the parameter configuration that maximizes quality, regardless of the remaining power $(P_s - P_{sust} - P_{bal})$ (Line 9).

Algorithm 1 The adaptation algorithm

Inputs: solar power P_s (watts) ; workloads L_{sust}, L_{bal} and L_{perf} (req/s); set of configurations {conf} ; quality function qoe : conf $\rightarrow$ int
Outputs: triplet of configurations (conf_{sust}, conf_{bal}, conf_{perf}), one for each user mode
Algorithm:

1: $\text{conf}_{min} \leftarrow \arg\min_{\text{conf}}(qoe(\text{conf}))$
2: $\text{conf}_{sust} \leftarrow \text{conf}_{min}$
3: **if** $P(\text{conf}_{sust}, L_{sust}) + P(\text{conf}_{min}, L_{bal}) \geq P_s$ **then**
4: $\quad \text{conf}_{bal} \leftarrow \text{conf}_{min}$
5: **else**
6: $\quad \text{conf}_{bal} \leftarrow \arg\max_{\text{conf}}(qoe(\text{conf}) \mid (P(\text{conf}_{sust}, L_{sust}) + P(\text{conf}, L_{bal})) \leq P_s)$
7: $\quad \text{conf}_{sust} \leftarrow \arg\max_{\text{conf}}(qoe(\text{conf}) \mid (P(\text{conf}, L_{sust}) + P(\text{conf}_{bal}, L_{bal})) \leq P_s)$
8: **end if**
9: $\text{conf}_{perf} \leftarrow \arg\max_{\text{conf}}(qoe(\text{conf}))$

We note that our algorithm works in all cases of green power availability: if we have no green power (*e.g.,* at night), it will simply select the minimal quality for both sustainable and balanced users.

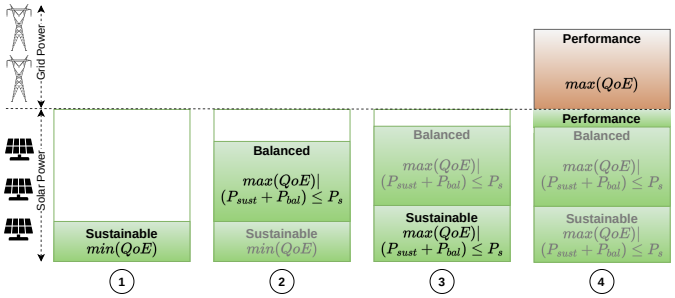


Fig. 2: Our adaptation algorithm steps.

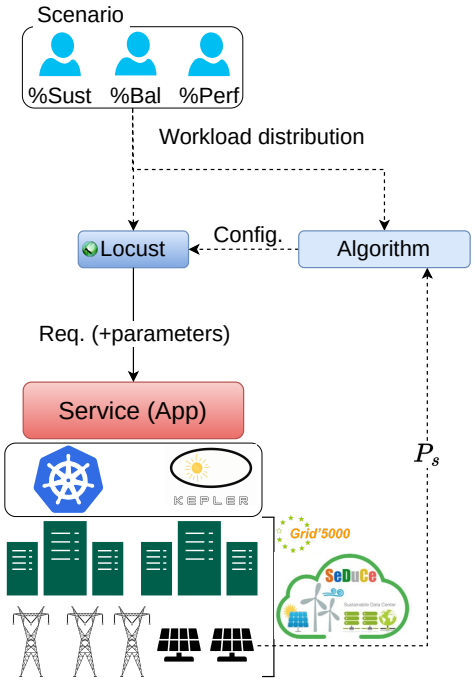


Fig. 3: Experimental setup.

V. Evaluation

A. Experimental Setup

To validate our approach, we deployed a prototype on the physical servers of the *ecotype* cluster (Figure 3). The special feature of this cluster, which is part of the *Grid'5000* experimental testbed [7], is that it is powered by solar panels and the electricity grid, thanks to the *Seduce* project [8]. The characteristics of these servers are detailed on the official *Grid'5000* website [9].

To ensure reproducibility of our experiments, we used the *EnOSlib* library [10] that provides a modern toolkit for automatic deployment and configuration systems. At the platform level, we used *Kubernetes* [11] to easily deploy and manage the application services to be evaluated. In addition, we deployed the *Kepler* software probe [12] to measure the application's energy consumption. This will help us evaluate our approach. We developed the image resizing application [13], described on Section III, using *Python* and *Flask*.

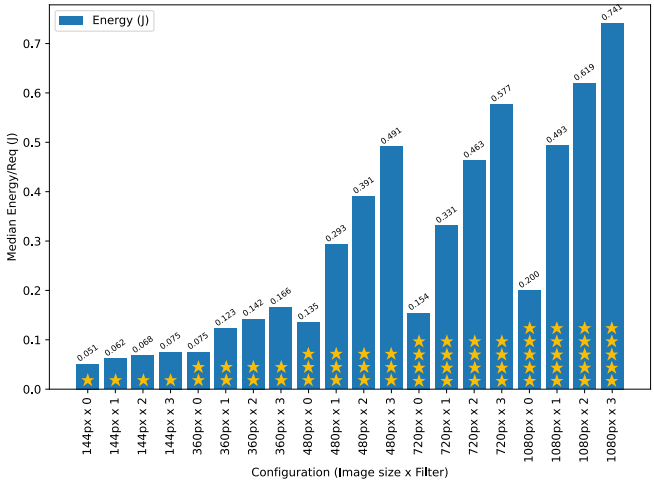


Fig. 4: Application calibration results. The four resampling filters (*nearest*, *bilinear*, *bicubic* and *Lanczos*) are denoted from 0 to 3, respectively, for ease of reading.

The *Locust* load generator [14] plays two roles in our case: the first is to generate user requests based on a specific load distribution according to the three usage modes (Performance, Balanced and Sustainable). The second is to apply the configuration parameters recommended by our adaptation algorithm to these requests.

B. Calibration

Our adaptation algorithm needs to know the energy consumption and the quality of each configuration of the SaaS service. We measure the energy consumption by benchmarking the application on our experimental setup, in a preliminary *calibration* phase where we iterate on all configurations.

Figure 4 shows the calibration results for the twenty parameter configurations (x-axis) of our image resizing application. Each bar reflects the energy consumption (in Joules) needed to process a single user request using a given parameter configuration. This energy consumption is measured with the *Kepler* software probe. In addition, Figure 4 also shows the level of output quality determined in Section III for each parameter configuration.

C. Adaptation scenarios and results

For our experiment, we assume that our application has 13 W of available green power, with a workload of 120 Req/s for the entire 60-second experiment. The green power and workload values are chosen to provide sufficient energy for sustainable and balanced user modes, which is the most interesting case. Other cases are not described in this paper due to limited space. We consider nine *scenarios* with various distributions of user-selected modes.

The results of our adaptation algorithm are shown in Figure 5. On the x-axis, we show the different scenarios, expressed as the relative percentage of each user mode in the workload. On the y-axis, each bar represents the average power

(in Watts) consumed by the application while it processes all user requests. This power is again measured with the *Kepler* software probe. The amount of available green power is shown as a horizontal dotted line. Finally, we also use stars on the figure to represent the quality level obtained by each user mode in the configuration. The first bar represents a "*Baseline*" with no adaptation algorithm. We consider it as the 0^{th} scenario in the rest of the paper. We note that it is equivalent to the scenario with 100% of *Performance*-mode users: this is because our algorithm produces the maximum output quality in this case, just like the baseline.

In Figure 5, we see that for *Performance* mode, *Excellent* quality is guaranteed in all scenarios, and that *Balanced* mode users always obtain a higher quality than *Sustainable* mode users. When no user selects the *Performance* mode (1^{st}, 5^{th} and 7^{th} scenarios), the energy consumed is predominantly green: the configurations selected by the adaptation algorithm have succeeded in achieving the goal. In the case of a single mode (1^{st} and 5^{th} scenarios), the adapter is able to ensure *Medium* quality for all users, thanks to the available green energy budget. On the other hand, in the case of *Sustainable* and *Balanced* modes (7^{th} scenario), users of the first mode are set to *Poor* quality, so that users of the second mode benefit from *Very good* quality. This corresponds exactly to our definitions of these modes (see Section II).

The impact of involving users is clearly visible on energy consumption in this experiment. With the exception of the 100% performance scenario, we are always able to reduce the application's energy consumption thanks to our adaptation algorithm. In the best scenarios (1^{st} and 5^{th}), we manage to reduce energy consumption by up to 84.7% compared to the baseline, and we emit almost no CO_2 by using only green energy. This comes at the price of degraded quality. Even more interesting: in the 3^{rd}, 6^{th} and last scenarios, energy consumption is reduced by up to 32.7% without any visible degradation in quality, even for *Balanced* and *Sustainable* users.

To conclude, the results of our experiment show that the adapter successfully strikes a balance between quality and carbon footprint, depending on the user's priorities and the overall green power budget.

VI. Current scope and challenges

Quality evaluation. Unlike Quality of Service (QoS) and measurable metrics such as response time and service availability, Quality of user Experience (QoE) is influenced not only by technical factors, but also by social and psychological ones. This is why we felt it was more important to use an illustration of the QoE (image perception) to involve the end-user, as their psychology is very different from one individual to another. However, evaluating QoE this way is challenging, since it is very application-dependent and may require costly campaigns with human subjects. The next step to improve quality evaluation is to use test protocols developed by ergonomists or human science specialists to help us better assess user satisfaction and thus better qualify QoE.

Online calibration. Our current proposal requires preliminary calibration of all possible application configurations. We realize that this may be hard to scale, because it is both time-consuming and energy-consuming. We could instead rely on online learning during actual application usage: the consumption of each configuration would be measured with real requests, eventually stabilizing to an energy consumption model equivalent to our calibration. The main challenge is to attribute the observed energy consumption back to each configuration: at each time step, we can only measure a single value of energy consumption, while different users may be using different configurations during this time step.

Exploit other forms of application flexibility. Request parameterization, like in our image resizing application, is one way of adding flexibility to the application. In the context of a microservices architecture, an alternative would be to dynamically reconfigure the structure of the architecture. This would alter the flow of internal calls, allowing for instance to eliminate the most energy-hungry microservices when required.

Extension to public Clouds. Our approach requires measuring the energy consumption of the SaaS application with different configurations to be able to perform the calibration step. Measuring the energy consumption is usually not possible on a public Cloud infrastructure, because the virtualization layer prevents access to low-level CPU hardware counters. As such, our calibration step cannot be performed on a public Cloud infrastructure. However, our adaptation algorithm can be used on a public Cloud infrastructure, since we only need an estimation of the energy consumption (given by the result of the calibration).

VII. Related Work

Several research work have investigated the SLO guarantee, energy-efficiency and their trade-off on Cloud applications.

Brownout. A pathfinder approach to the adaptive management of resources and applications in Cloud computing is "brownout" by Klein et al. [15]. The authors proposed that Cloud applications should be divided into two parts: mandatory and optional. Mandatory parts must be permanently active; optional parts, on the other hand, need not be permanently active and can be temporarily deactivated to avoid over-provisioning of resources. The authors of [16] identified the challenges and propose directions for future brownout research. Notably, leveraging microservice architectures and renewable energy, Xu et al. [17] offered a self-adaptive approach to resource management for interactive and batch workloads. These contributions essentially propose a dynamic trade-off between resource consumption and QoS. Although we consider energy consumption instead of computing resource consumption, our approach achieves a similar goal with different means.

Trade-off quality – performance. In [18], the authors studied in-depth analysis of energy consumption and performance trade-offs, enabling intelligent use of green energy for interactive Cloud applications. Larsson et al. [19] introduced

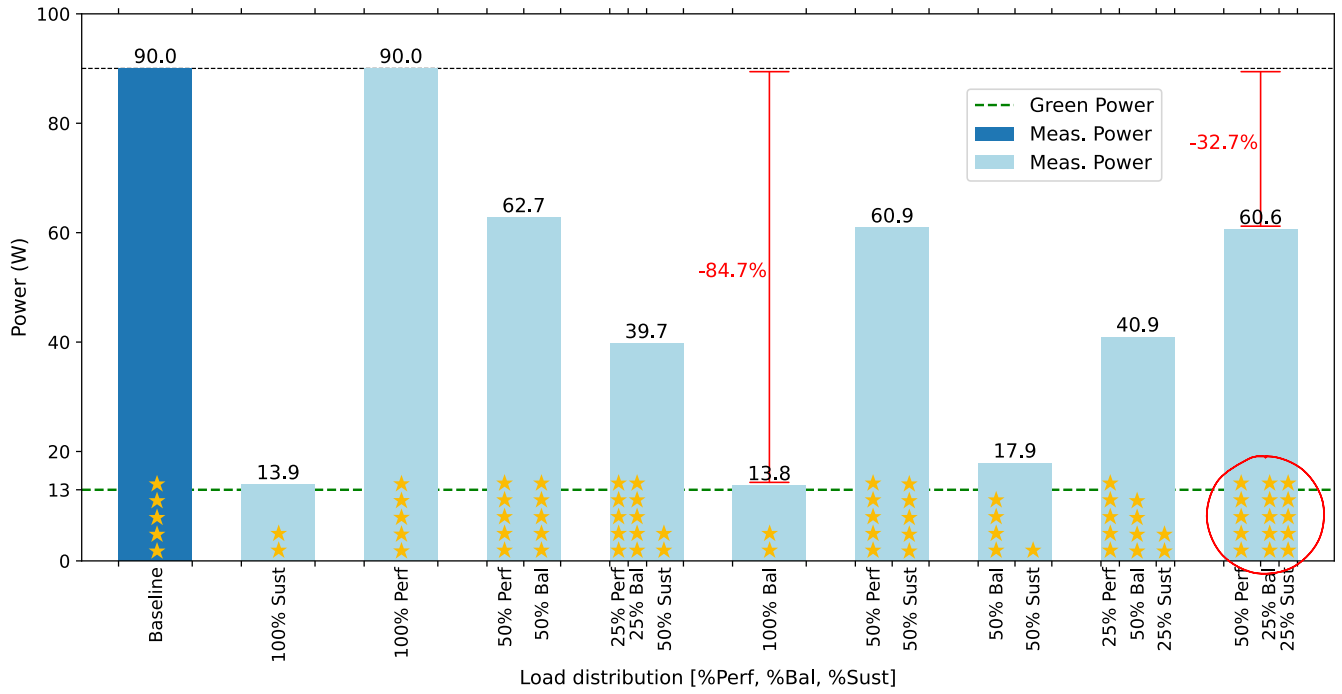


Fig. 5: Evaluation of the application's power consumption and output quality after adaptation. Each column is a different user mode distribution scenario (numbered from 0 to 9, Baseline is the 0^{th} one).

the notion of Quality-Elasticity to deal with poor resource utilization. The concept of Quality-Elasticity goes beyond the Brownout application: instead of a simple binary choice to reduce the quality or not, it introduces degrees of quality.

Users involvement. The authors of [20] gave IaaS/PaaS users a choice of three execution modes (large, medium, small) for their jobs. Smaller execution modes require fewer resources, but take longer to complete. They achieve gains through spatial consolidation with a bin-packing algorithm. Madon et al. [21] focused on "direct" data-center users, i.e. *IaaS* and *PaaS* users such as DevOps developers, to study four types of job submit behavior: delaying, output degrading, reconfiguring, and renouncing. They study the impact of these behaviors on several different parameters, including energy consumption and performance.

Conclusion. To the best of our knowledge, there is no related work involving end-users in the energy consumption of SaaS applications. End-users choose a carbon footprint-related mode for their use of the Cloud application, being masters of their own trade-offs between quality and energy footprint.

VIII. Conclusion

In this paper, we proposed an adaptation approach that reduces the carbon footprint of Cloud applications by involving end-users. It achieves this thanks to a controllable quality-energy trade-off. We have evaluated our proposal with a simple application on an experimental testbed, and found that our algorithm can reduce energy consumption while finding good trade-offs depending on end-user preferences.

Future work will focus on the challenges mentioned above, such as quality evaluation protocols and online calibration. We will also investigate a more realistic applications, for which we expect the quality-energy trade-off to be more challenging to design.

Finally, we plan to extend the adaptation algorithm to work with more diverse energy sources, for instance by taking into account the carbon intensity of the energy sources instead of relying on a simple distinction between "clean" green energy and "dirty" brown energy.

Acknowledgment

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).

References

[1] C. Freitag, M. Berners-Lee, K. Widdicks, B. Knowles, G. S. Blair, and A. Friday, "The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations," *Patterns*, vol. 2, no. 9, 2021.

[2] S. Long, Y. Li, J. Huang, Z. Li, and Y. Li, "A review of energy efficiency evaluation technologies in cloud data centers," *Energy and Buildings*, vol. 260, p. 111848, 2022.

[3] W. Deng, F. Liu, H. Jin, B. Li, and D. Li, "Harnessing renewable energy in cloud datacenters: opportunities and challenges," *iEEE Network*, vol. 28, no. 1, pp. 48–55, 2014.

[4] P. Jacquet, T. Ledoux, and R. Rouvoy, "SweetspotVM: Oversubscribing CPU without Sacrificing VM Performance," in *CCGrid'24 - 24th IEEE/ACM international Symposium on Cluster, Cloud and Internet Computing*. Philadelphia, United States: IEEE, May 2024, pp. 1–10. [Online]. Available: https://hal.science/hal-04454043

[5] "Resampling filters," https://pillow.readthedocs.io/en/latest/handbook/concepts.html#filters, 2024, [Online; accessed 01-02-2024].

[6] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed. Addison-Wesley Professional, 1994.

[7] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jégou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab *et al.*, "Grid'5000: A large scale and highly reconfigurable grid experimental testbed," in *The 6th IEEE/ACM International Workshop on Grid Computing, 2005.* IEEE, 2005, pp. 8–pp.

[8] J. Pastor and J. M. Menaud, "SeDuCe: Toward a testbed for research on thermal and power management in datacenters," in *Proceedings of the Ninth International Conference on Future Energy Systems*, 2018, pp. 513–518.

[9] Ecotype, https://www.grid5000.fr/w/Nantes:Hardware#ecotype, 2024, [Online; accessed 01-02-2024].

[10] R.-A. Cherrueau, M. Delavergne, A. Van Kempen, A. Lebre, D. Pertin, J. R. Balderrama, A. Simonet, and M. Simonin, "Enoslib: A library for experiment-driven research in distributed computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1464–1477, 2021.

[11] Kubernetes, https://kubernetes.io, 2024, [Online; accessed 01-02-2024].

[12] Kepler, https://sustainable-computing.io, 2024, [Online; accessed 01-02-2024].

[13] "Image resizing application," https://gitlab.imt-atlantique.fr/fil-a3-frontback-2023-energy, 2024, [Online; accessed 01-02-2024].

[14] Locust, https://locust.io, 2024, [Online; accessed 01-02-2024].

[15] C. Klein, M. Maggio, K.-E. Årzén, and F. Hernández-Rodriguez, "Brownout: Building more robust cloud applications," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 700–711.

[16] M. Xu and R. Buyya, "Brownout approach for adaptive management of resources and applications in cloud computing systems: A taxonomy and future directions," *ACM Comput. Surv.*, vol. 52, no. 1, jan 2019. [Online]. Available: https://doi.org/10.1145/3234151

[17] M. Xu, A. N. Toosi, and R. Buyya, "A self-adaptive approach for managing applications and harnessing renewable energy for sustainable cloud computing," *IEEE Transactions on Sustainable Computing*, vol. 6, no. 4, pp. 544–558, 2020.

[18] M. S. Hasan, F. Alvares, T. Ledoux, and J.-L. Pazat, "Investigating energy consumption and performance trade-off for interactive cloud application," *IEEE Transactions on Sustainable computing*, vol. 2, no. 2, pp. 113–126, 2017.

[19] L. Larsson, W. Tärneberg, C. Klein, and E. Elmroth, "Quality-elasticity: Improved resource utilization, throughput, and response times via adjusting output quality to current operating conditions," in *2019 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE, 2019, pp. 52–62.

[20] D. Guyon, A.-C. Orgerie, C. Morin, and D. Agarwal, "Involving users in energy conservation: a case study in scientific clouds," *International Journal of Grid and Utility Computing*, vol. 10, no. 3, pp. 272–282, 2019.

[21] M. Madon, G. Da Costa, and J.-M. Pierson, "Characterization of different user behaviors for demand response in data centers," in *European Conference on Parallel Processing*. Springer, 2022, pp. 53–68.