Optimizing Green Energy Consumption of Fog Computing Architectures

Adrien Gougeon, Benjamin Camus, Anne-Cécile Orgerie

Optimizing Green Energy Consumption of Fog Computing Architectures

Adrien Gougeon*, Benjamin Camus* and Anne-Cécile Orgerie*
*Univ. Rennes, Inria, CNRS, IRISA, ENS Rennes, France
Email: {adrien.gougeon, anne-cecile.orgerie}@irisa.fr

***Abstract*—The Cloud already represents an important part of the global energy consumption, and this consumption keeps increasing. Many solutions have been investigated to increase its energy efficiency and to reduce its environmental impact. However, with the introduction of new requirements, notably in terms of latency, an architecture complementary to the Cloud is emerging: the Fog. The Fog computing paradigm represents a distributed architecture closer to the end-user. Its necessity and feasibility keep being demonstrated in recent works. However, its impact on energy consumption is often neglected and the integration of renewable energy has not been considered yet. The goal of this work is to exhibit an energy-efficient Fog architecture considering the integration of renewable energy. We explore three resource allocation algorithms and three consolidation policies. Our simulation results, based on real traces, show that the intrinsic low computing capability of the nodes in a Fog context makes it harder to exploit renewable energy. In addition, the share of the consumption from the communication network between the computing resources increases in this context, and the communication devices are even harder to power through renewable sources.**

I. Introduction

The growing importance of the Cloud paradigm has led to many propositions to reduce its costs and to improve its profitability. Energy consumption is an important part of the operational cost of Cloud architectures. It is becoming a limiting factor to build more data centers and is also becoming dominant in the Total Cost of Ownership (TOC) [21]. In addition to its energy consumption, the environmental impact of the Cloud, and IT technologies in general, increases dramatically. Between 2013 and 2017 the share of electricity consumed in the world by the IT sector went from 11% to 14% and keeps growing [11]. Regarding the utilization phase of IT, data centers were responsible for 32% of the global consumption of the sector in 2017. The network, which is essential for Cloud computing, was responsible for 34% of this consumption [11]. Cloud providers are pressured by governments to reduce their environmental impact and they also want to provide a better image to the end-user. In order to reduce this impact, research around greener Cloud has been important in the last few years.

Many works have been conducted to reduce the environmental impact of the Cloud, either by reducing its overall consumption [3], [8], or by considering the usage of energy coming from renewable sources [12], [18], [23], [2], [5]. These works present noticeable advancements, according to the current architectures of the Cloud. However, the emerging Fog paradigm is supposed to be deployed in the near future [1]. This new architecture, as an extension of the Cloud, brings new challenges concerning energy consumption and environmental impact. Works have been conducted around the deployment of Fog architectures taking into account its energy consumption [9], [28], [16] and low-latency requirements [1].

The concept of Fog is still recent and there is no consensus on its definition yet. In this work, we consider it as a distributed architecture composed of geographically distributed low-capabilities nodes and a distributed resource management (i.e. no centralized control node). In this model, each node, or Point-of-Presence, receives and processes local or close users' requests without general coordination. This work aims to find solutions to adapt the recent Fog paradigm to the current trend toward reduction of energy consumption and increasing use of green energy (i.e. produced by renewable sources). More and more Cloud providers resort to producing their own renewable energy, like Apple for instance [5]. Yet, the distributed nature of Fog resources makes it harder to exploit locally-produced green energy as more sites are considered. The main challenges tackled in this work are:

- job scheduling in a fully distributed architecture;
- dealing with the intermittent production of energy from renewable local sources (solar panels);
- improving energy consumption of the Fog without impacting severely its benefits in terms of latency.

The results show that exploiting green energy in a Fog context is a tough task. The low capabilities of the nodes limit severely the number of jobs a node can host, leading to issues in properly exploiting the green energy produced locally. Another main result of this work is that the energy used by the communication network between the Fog nodes constitutes the major part of the energy consumption of the whole platform, and should consequently concentrate the attention for reducing the overall Fog impact.

The remainder of the paper is structured as follows: Section II presents related work; Section III details the employed models; Section IV describes our approach; Section V presents the experimental setup and the results; Section VI concludes this work and introduces future work.

II. Related Work

On average, an idle server can consume up to 70% of the power consumed by the server running at the full CPU speed [3]. In addition, many servers inside data centers are

under utilized or even not utilized at all. In 2010, a study on 188 data centers estimated that 10% of the servers inside them are never utilized [21]. Therefore, a simple solution to reduce energy consumption is to switch-off unused machines and gather used resources on the smallest number of machines. This problem, known as consolidation, is widely explored.

Beloglazov and al. [3] present a work to reduce energy consumption with consolidation, while respecting the SLA (Service Level Agreement), crucial for Cloud providers. Their work is based on thresholds policy concerning the CPU usage of a physical machine. On the one hand, a lower threshold defines if VMs on the server should be migrated and the servers are switched-off to save energy. On the other hand, an upper threshold defines if VMs on the server should be migrated to ensure sufficient elasticity for the remaining VMs.

Decentralized Clouds are geographically distributed over a large area to provide better Quality-of-Service. Several data centers, each with their own on-site renewable source, contribute to the same Cloud system. In a context of Cloud providers aiming to improve their consumption of green energy, the decentralized approach offers new possibilities. The intermittent production of green energy on a single site might be counterbalanced by an overproduction in another site. Multi-site Clouds bring new challenges and solutions to optimize their green energy consumption. Tang and al. [23] propose a solution to reduce the brown energy consumption between multiple, widely geographically distributed, data centers. The main idea is to take advantage of a complementary production of green energy over the various data centers.

Several works have introduced some principles to implement the Fog, but few took in consideration its energy consumption. Deng and al. [9] present a vision of the Fog-Cloud interplay, while taking into account the trade-off between power consumption and latency. The latency sensitive requests are processed locally on the Fog devices, whereas the others are dispatched in the Cloud using wide area network (WAN).

The feasibility and interest of the Fog keep being demonstrated by recent works. But, its energy consumption and environmental impact is often neglected. Some research consider the energy-efficiency of Fog architectures, but only consider it as a trade-off between latency and power consumption and do not intend to improve it. In addition, the usage of on-site green energy has not been explored in this context. The goal of our work is to optimize the green energy consumption of distributed Fog architectures while also considering overall energy-efficiency.

III. Fog computing model

A. Fog Architecture

Two main Fog architectures stand out in literature. The first one is a fully connected Fog: it typically refers to a Fog covering a small area, such as a city or just a building or a factory. Such architecture is adapted to applications looking for communication between nodes, such as augmented reality video games [24] or coordinating machines in a factory [20]. The second one is a hierarchical Fog, it allows to cover a larger area, and thus is adapted to applications looking for aggregating data over a wide area, such as video surveillance applications [19] or balance the load between Cloud and Fog, such as low-latency image-recognition applications [10]. Also, green energy production may varies between different localization, thus, the larger area covered by a hierarchical Fog architecture may allows to mitigate the green production.

Here, we consider an ISP-like network where the Fog computing nodes are the Internet boxes of the end-users (also called home gateways). Such an architecture was considered even before the appearance of the Fog paradigm by ISP to deliver low-latency services to their users [26]. This architecture is adapted to applications looking for aggregating data over a wide area, such as video surveillance applications [19] or balance the load between Cloud and Fog, such as low-latency image-recognition applications [10]. Relying on the existing ISP network to build a Fog architecture ease the deployment and maintenance operations.

The main objective of a Fog architecture is to process users requests, named hereafter jobs. Conversely to other Cloud-oriented work presented, we consider here a fully-distributed Fog environment, which means that jobs are not submitted to a central scheduler. Each new job is sent from an IoT or end-user device to the closest Fog node. Jobs are continuously submitted to the platform and treated as soon as they are received. We do not consider a distributed file system such as a network file system. We assume that jobs do not require to share data among them, as it is typically the case for video-stream decoding applications for instance [26].

B. Power models

For simplicity sake, we consider an homogeneous Fog architecture, where all nodes are identical, with the same number of cores and power profile to ease the understanding of scheduling policies' behavior. The power model comes from Kaup and al. [17]. Nodes have a static power consumption based on their idle consumption (P_{idle}), Ethernet consumption ($P_{eth,idle}$) and WLAN consumption ($P_{wlan,idle}$); and a dynamic consumption ($P_{cpu}(u)$) based on the CPU usage (u). In addition, while booting or shutting down, the consumption is modified to reflect the actual device behavior.

$$P = P_{idle} + P_{cpu}(u) + P_{eth,idle} + P_{wlan,idle}$$

A virtual machine (VM) is created to host each job and this VM is deleted after the end of the job. The only parameter for the creation of a VM is the number of required cores. Cores are not shared between VMs (i.e. we do not consider over-commitment here as it would modify the performance and energy consumption model of jobs).

The power model used for the network consumption comes from Guegan and al. [13]. The energy consumption of a link corresponds to the consumption of the its two Ethernet devices' ports connected by the link. It is the sum of the idle power consumption of both ports (P_{idle}^{net}), and its maximum dynamic power consumption is given by the following

equation depending on the maximum packet size (Maximum Transmission Unit), the transmitted data volume (D), the link bandwidth (BW), and an energy cost per processed byte (E_{byte}) and per processed packet (E_{pkt}):

$$E_{max} = \frac{P_{idle}^{net} \times D}{BW} + 2 \times \left(8 \times D \times E_{byte} + \frac{E_{pkt} \times D}{MTU} \right)$$

C. Energy Production

Concerning the power supply, we distinguish two kinds of production: brown energy (from the regular electrical grid) and green energy (from on-site renewable sources). Although nodes are all linked to the electrical grid, some of them can also consume on-site green energy, produced by photovoltaic panels connected to them. The green energy production is variable over time and not known in advance. For simplicity's sake, the excess of produced green energy is not re-injected in the electrical grid. Similarly, to simply examine the integration of renewable energy sources into Fog platforms, we do not consider here energy storage devices. Indeed, their sizing requires specific studies depending on the nodes power consumption and on the photovoltaic panel production, and it would thus add variability to our results [18].

IV. Integration of on-site renewable energy

The Fog paradigm is strongly latency-oriented. A key metric for this kind of architecture is the latency perceived by application users. The goal of this work is to explore the interest of a trade-off between the latency and the optimization of the green energy consumption in a distributed Fog architecture. Our approach consists in exploring several job allocation algorithms and consolidation policies to evaluate whether green energy can be exploited within this context. In particular, we play with the latency constraint to determine if its release can bring significant improvements in terms of brown energy consumption.

As latency is a strong constraint, greedy algorithms are favored to provide fast energy-efficient allocation policies. We first explicit three greedy allocation algorithms used to select a host for a job submitted to a specific node: Lowest Latency, First Green, and Most Green (detailed hereafter). Then, we provide three greedy consolidation policies used to consolidate the Fog workload: Consolidate All, Consolidate Brown, and Consolidate Ratio. The energy efficiency of these algorithms is explored in Section V.

A. Allocation Algorithms

To place jobs, the criteria are the latency between the Fog node and the user, and the green energy produced at the node site. All three algorithms are greedy by latency in their search for a host. The first algorithm only considers latency and is used as baseline. The two others consider green energy in addition to the latency. Note that we consider here VM, but the proposed algorithms would work without any change with containers, since we do not consider VM migration.

1) Lowest Latency: This first algorithm aims to reduce the end-to-end latency of a job, from the initial node to the host of the job. The initial node is the node on which the job was submitted and is by definition the closest node to the end-user. The host is the node on which the job is running. The algorithm performs a search in the hierarchical architecture to find the available node that is the closest to the initial node, and consequently to the end-user. This algorithm is expected to provide the overall best latency of the studied algorithms.

2) First Green: The second algorithm, Algorithm 1, aims to find a host with the lowest latency, but producing locally green power. The idea is similar to the previous algorithm: this algorithm will continue searching until it has found a node that can host the job, but only considering nodes powered by green energy. Yet, returning a node producing green energy is considered as a weak constraint, otherwise it could cause an extremely inefficient allocation. Indeed, if the algorithm finds a potential host producing green power, this node may be very far from the initial node, thus heavily impacting the latency. One of the principal goal of the Fog is to be close to the end-user, often for latency issues. In that sense, we consider that, in order to run properly, some applications require a specific latency threshold. This threshold is given as a parameter of the algorithm and is exceeded only if one cannot find a host in this latency range (whether it produces green energy or not).

The initial node is added to an array of nodes kept sorted by increasing latency as detailed in Algorithm 1. As long as we do not meet a termination condition, we successively remove the first node of the array and add its direct neighbors. The first node explored with enough free cores to host the job is stored as *brown_host*. Termination conditions are:

- there are no more nodes in the array;
- a node with enough free cores and producing green power has been found;
- the maximum latency has been exceeded and we have found a node powered with brown energy.

This algorithm explores the compromise between latency and green energy consumption. The submitted jobs are expected to be hosted further from their initial submission node, but with less brown energy consumption in comparison with the first algorithm.

3) Most Green: This third algorithm is similar to the second one (Algorithm 1). The only difference is that it searches for the node producing the most green power in its latency range, and does not stop at the first node producing green energy.

B. Consolidation Policies

The consolidation process implies to switch-off machines in order to save the static energy consumption of the nodes. Switching-off a machine implies a period during which it is unavailable; and the machine must be switched-on to host jobs again, making it unavailable again for a specific duration. To measure the impact of consolidation, we use the number of nodes that are placed on a booting host, i.e, a host which is awaken by the placement of the job on it, or a host already in the booting process. The unavailability of nodes can reduce

Algorithm 1: First Green

```
1  Inputs: initial_host, job_cores, max_latency ;
2  Outputs: suitable host for the job ;
3  potential_hosts + = initial_host ;
4  brown_host = NULL ;
5  host = potential_hosts[0] ;
6  while host ≠ NULL && (host→latency ≤
     max_latency || brown_host == NULL) do
7      if host→spare_cores ≥ job_cores then
8          if host→green_power > 0 then
9              return host
10         end
11         else
12             if brown_host = NULL then
13                 brown_host = host ;
14             end
15         end
16     end
17     potential_hosts.pop() ;
18     potential_hosts + = host→neighbors ;
19     host = potential_hosts[0] ;
20     potential_hosts.sort() ;
21 end
22 return brown_host
```

the responsiveness of the architecture. Some jobs might have to wait for a machine to boot up until they can run on it. However, once a job is allocated, the boot duration is not an issue anymore: it does not impact the latency perceived by the user anymore. In this sense, we introduce the notions of final latency: the time from initial node to the host after the allocation and eventual booting duration, only taking into account the communication time, i.e., latency due to the distance between users and Fog nodes.

Jobs with consequent duration and many interactions with the end-user should optimize final latency, for example an application hosting a video game. Conversely, applications submitting many jobs with short duration should tend to reduce the number of jobs placed on booting hosts. In this work, we consider jobs looking to optimize the final latency as they usually last longer and consequently can better make advantage of renewable energy sources. The consolidation routine intervenes whenever a job ends on a node, and can only shutdown a node if there is no job placed on it. Three greedy consolidation policies are explored in this work.

1) Consolidate All: The first consolidation policy is rather simple: whenever a job ends on a host, if it has no other job placed on it, then it is switched-off. This consolidation policy is expected to significantly reduce the energy consumption, but will surely increase the initial latency.

2) Consolidate Brown: The second consolidation policy consists in consolidating only nodes which do not produce green energy. As this energy has a negligible cost, it could be interesting to keep the nodes with green energy turned-on, and thus reducing the impact on the responsiveness of the architecture. However, a node previously shut down because it did not produce green energy at this moment will not be switched on because it produces green energy again. It will be switch on when a job is allocated on it. This policy is particularly suitable for the allocation algorithms First Green and Most Green. These algorithms always search for nodes producing green power to place jobs, and with this consolidation policy, these nodes will be kept turned-on, thus reducing initial latency.

3) Consolidate Ratio: The third consolidation policy considers the trade-off between initial latency and reducing energy consumption. While previous consolidation policies only consider the state of the current host, this policy considers the state of all nodes with the same feeder node as him, i.e., in the same district, as detailed in Section V-A1. The routine checks the availability of the other nodes to take a decision. A node is considered available if it is switched-on and has a percentage of cores available at least equal to the per-node availability threshold, given as a parameter to the routine. A core is available if it is not used by a VM on the node. The routine classifies each node as available or not available and maintains a target node availability ratio, given as a parameter of the routine. The node availability ratio is the percentage of nodes available among the nodes checked. If shutting down this node maintains the node availability ratio, then it is shutdown. If it cannot be shutdown, the routine determines how many nodes should be switched-on to recover the node availability ratio, and tries to switch on this number of nodes among the checked nodes. The idea behind this node availability ratio consists of reducing the initial latency: available nodes are kept powered on even if they are not used for speeding up the deployment of new incoming jobs.

V. Experimental results

A. Experimental Setup

For the evaluation, we use SimGrid [6], a simulation toolkit designed for parallel and distributed large-scale system analysis. Its models include CPU, network, VMs and energy consumption, and have been theoretically and experimentally proved to be efficient and accurate [14], [25], [15].

1) Fog Architecture: As mentioned in Section III, we evaluate our policies on a hierarchical Fog architecture based on the ISP topology provided in [7]. This architecture comprises five levels of network devices: core, backbone, metro, feeder, and end-user. There is 5 to 10 end-users per feeder. Latency is not included in the original paper [7]. We used real measurements between European cities as a reference for the latency. The used values are presented in Table I. They are randomized following a Gaussian function to reflect the dynamic conditions of ISP networks.

2) Job Traces: To the best of our knowledge, there are currently no available traces for Fog infrastructures. Hence, we chose to use regular Cloud traces for our experiment. Eucalyptus traces are anonymized traces built from the log files of several systems running Eucalyptus private clouds [27].

TABLE I
LATENCIES USED FOR THE LINKS (FROM *https://wondernetwork.com/*).

Network distance (km)	**Distance between cities (km)**	**Source city**	**Destination city**	**Latency (ms)**	**Standard deviation**
0.1-0.5	14	Rotterdam	Alblasserdam	0.721	0.05
5-15	14	Rotterdam	Alblasserdam	0.721	0.05
1-50	35	The Hague	Alblasserdam	1.478	0.161
50-500	300	Munich	Frankfurt	8.122	1.552
550-600	610	Paris	Marseille	18.607	0.07

The traces have been reworked to fit our architecture: jobs have been scaled for each job not to ask for more cores than the number of cores of one node, i.e., four cores (a unique job is not distributed, it can only take place on one host at a time). After being scaled, jobs are duplicated in the traces, so at the busiest moment of the simulation 73% of the available Fog cores are used. This allows to have the architecture neither underutilized nor overutilized.

3) Green Power Traces: The traces used to represent the green energy production on our nodes are from real traces of photovoltaic panels as part of the Photovolta project [22], carried out at the University of Nantes in France. The actual electricity production of the panels is logged every five minutes. We use production traces of 35 sessions recorded at various dates to represent the heterogeneity of the production at different locations. These production traces are scaled to match the energy consumption of the Fog nodes. Each scaled production trace covers, on average, half of the maximal consumption of a node. As nodes of the same district are geographically close, we employ the same trace for each node producing green energy in the same district (i.e. attached to the same feeder in the ISP network). We distribute the 35 scaled traces over the 260 districts considering that close districts also have the same solar irradiance.

4) Nodes Power Model: We simulate our Fog architecture considering each node as a Raspberry Pi(RPI) 3 B (cf. *https://www.raspberrypi.org/products/raspberry-pi-3-model-b/*). This kind of single-board computer meets our vision of the Fog, being easily deployable and replaceable with a low consumption profile, but still significant computational capacity [17]. Values used for the power model are summarized in Table II. Note that $P_{eth,idle}$ is negative to account for the periodic CPU activity even when Ethernet network is not used; this activity stops when the network is active.

TABLE II
POWER MODEL OF THE RASPBERRY PI 3 B, IN WATT [17].

Function	Value
P_{idle}	1.488
$P_{eth,idle}$	-0.1176
$P_{wlan,idle}$	0.899
$P_{cpu(u)}$	$0.6191u$ (with $u \in [0,1]$ the CPU utilization)

This power values come from Kaup and al. [17]. The job traces used in our simulation do not refer to data exchange between nodes and we suppose that data exchange time is far lower than data processing time for the considered Fog applications. It corresponds to highly distributed applications where data is produced and consumed locally. During shutting down or booting, a node consumes as much as if all its cores were fully loaded. Each node is considered to take 150 seconds to boot and 10 seconds to shut down [5].

5) Network Power Model: As explained in Section III-B, the network power model is from a paper by Guegan and al. [13]. Values used are described in Table III.

TABLE III
NETWORK POWER MODEL VALUES [13].

Parameter	Description	Value
P_{idle}^{net}	idle power	1.12 W
E_{byte}	energy per byte	3.4 nJ
E_{pkt}	energy per packet	197.2 nJ
MTU	maximum packet size	1500 bytes
BW	bandwidth	10 to 100 GBps

B. Results Analysis

We run simulations to explore the interest of distributed allocation algorithms and consolidation policies in order to integrate renewable energy sources to power Fog infrastructures. The parameters varying through simulations are summarized in Table IV. Latency range do not apply for the algorithm *Lowest Latency* and the parameters *Availability ratio* and *Availability threshold* only applies when the consolidation policy is set to *Consolidate Ratio*. Each simulation runs for a simulated duration of three days and with a configuration file fixing the variable parameters for this specific simulation. Each configuration is simulated 10 times, the results provided hereafter exhibit the average and standard deviation values of these 10 runs for each configuration.

TABLE IV
PARAMETERS EXPLORED IN THE SIMULATIONS.

Variables	**Values Explored**
Allocation Algorithm	Lowest Latency, First Green, Most Green
Consolidation Policy	None, Consolidate All, Consolidate Brown, Consolidate Ratio
Number of green producers per district	1, 3, 5
Latency Range (ms)	0, 5, 10, 20
Node availability ratio	0.1, 0.2, 0.3, 0.4
Per-node availability threshold	0.25, 0.5, 0.75

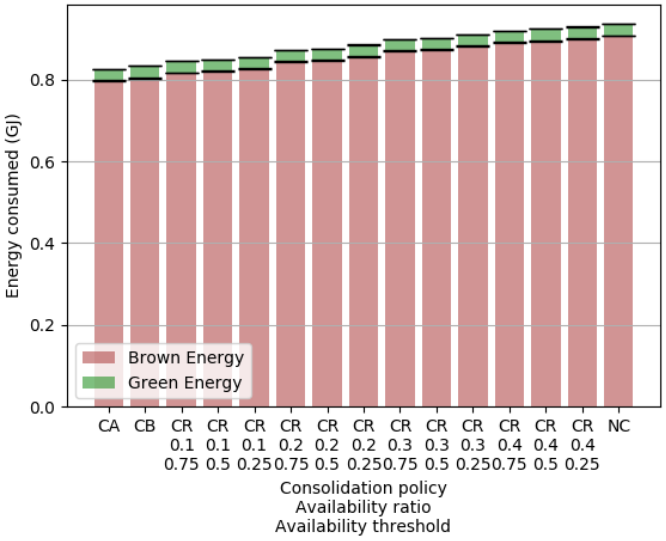

Fig. 1. Energy consumption of nodes depending on the consolidation policy. The allocation algorithm *First Green* is used. The number of green producers per district is fixed at 1 and the latency range is fixed at 5ms.

1) Energy Consumption of Nodes: The energy consumption of nodes for the allocation algorithm *First Green* is shown in Figure 1. It presents the energy consumption of the infrastructure for the nodes, depending on the consolidation policy. The policies are those detailed in Section IV: NC : No Consolidation, CA : Consolidate All, CB : Consolidate Brown, CR : Consolidation Ratio. The behavior of the consolidation policy *Consolidation Ratio* depends on two parameters, as explained in Section IV: node availability ratio and per-node availability threshold.

The other allocation algorithms follow the same trend and are omitted for saving space. For each allocation algorithm, the energy consumed by the nodes is similar. Without consolidation the infrastructure consumes significantly more energy. Among the consolidation policies, *Consolidate Ratio* is the one consuming the most energy since it consolidates less nodes by keeping some available nodes. The overall green energy consumption is small compared to the brown energy consumption, but it is coherent considering that only one user node per district produces green energy and they can produce only during daytime.

The allocation algorithms *First Green* and *Most Green* have an additional parameter compared to *Lowest Latency*: the latency range. Concerning more particularly the allocation algorithm *First Green*, the results of the simulations exploring the impact of the latency range and the number of green producers per district is shown in Figure 2. For these experiments, we do not apply any consolidation policy in order to isolate the latency effect.

The number of green energy producers has an obvious impact on green energy consumption as it significantly improve its green energy consumption. Increasing the latency range, allowing to search farther for a node producing green power, does not significantly improves the consumption of

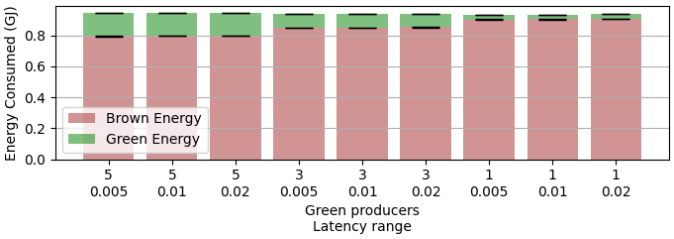

Fig. 2. Energy consumption of nodes depending on the number of green producers and latency range. The allocation algorithm *First Green* is used and no consolidation policy is applied.

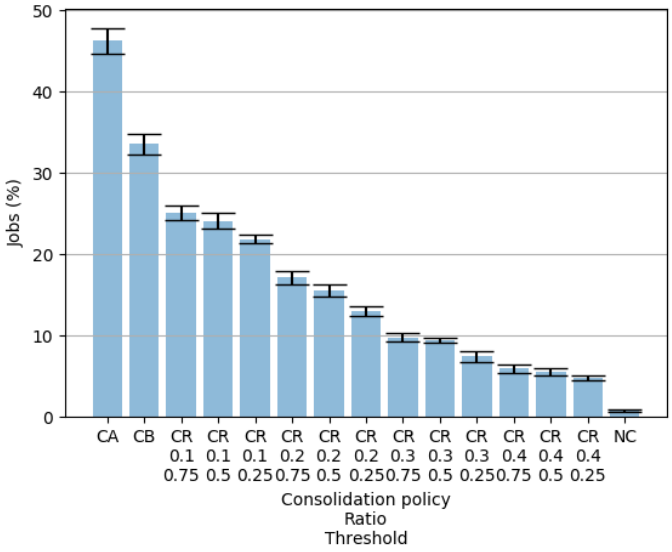

Fig. 3. Percentage of jobs placed on booting host depending on consolidation policy and with allocation algorithm First Green.

green energy, as shown in Figure 2.

2) Energy Consumption of the Network: The network energy consumption account for an important part of the platform total energy consumption. As the major part of the consumption is due to its idle consumption, network energy consumption is nearly identical in every simulation and represents, on average, 53.6% of the total platform energy consumption.

C. Impact of Consolidation on Latency

Latency is a key feature in the Fog. We have seen in Section V-B1 that consolidation policies reduce significantly the energy consumption of the nodes in a Fog environment. However, consolidation implies to switch-off nodes, and then switch them back on when needed. Figure 3 presents the percentage of jobs placed on a booting node, or on a node being woken up to host this job, depending of the consolidation policy applied. We observe that the percentage of jobs placed on a booting host decreases quickly as the consolidation policy becomes less aggressive, and thus less efficient to save energy.

D. Final Latency

Figures 4, 5 and 6 present the impact of allocation algorithms on final latency, i.e., how far the jobs are placed

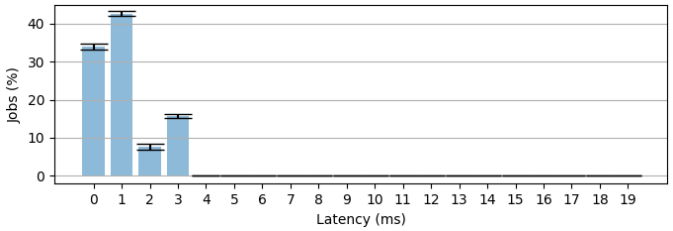

Fig. 4. Final latency with the allocation algorithm *Lowest Latency* and one node producing green energy per district.

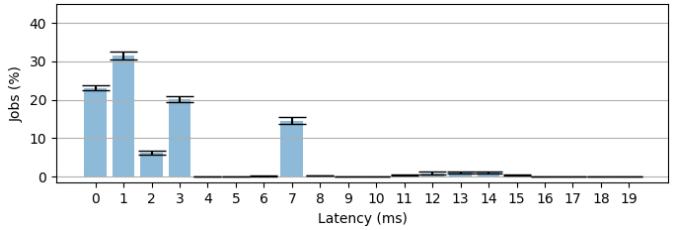

Fig. 5. Final latency with the allocation algorithm *First Green*, one node producing green energy per district and a latency range of 20 ms.

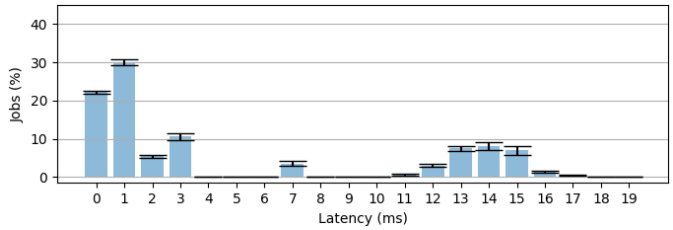

Fig. 6. Final latency with the allocation algorithm *Most Green*, one node producing green energy per district and a latency range of 20 ms.

from the initial node in terms of latency. The number of green producers per district is fixed to one and there is no consolidation policy applied. For the allocation algorithms *First Green* and *Most Green* (Figures 5 and 6), the latency range is fixed to 20 ms.

As expected, allocation algorithms follow their constraints. *Lowest latency* provides the best results as it only searches for the closest available node. *First Green* goes slightly farther as it looks for a node producing green energy. Finally, *Most Green* provides the worse results: as its main constraints is to find the node producing the most green energy in its latency range, it will often allocate jobs far from the node on which they were submitted. The spike at 7 ms and the Gaussian-like distribution around 14 ms are due to the architecture, e.g., there is no node distant by 5 ms but many are distant by 7 ms in the architecture.

E. Discussion

To summarize, consolidation is an important feature to reduce the energy consumption of Fog computing. But, it requires to deal with applications that are not heavily impacted by being launched on a booting node, increasing considerably its initialization. That is to say, consolidation is primarily oriented for applications running on long periods of time, or being highly predictable. Yet, using small nodes, as Raspberry Pi, which are often considered in literature as a key target for Fog nodes, does not allow to take advantage of efficient allocation policies. A greedy first fit allocation performs almost as good as more advanced greedy algorithms.

In this work, jobs are allocated at submission time and stay on the same node for all their duration. The benefits of migrating job between nodes to exploit more green power or to improve the consolidation could be investigated [5]. Also, here we consider a fixed Fog architecture that is connected through wired networks. Future Fog architectures could include mobile Fog nodes relying on wireless communications. As our proposed algorithms are fully distributed, they would be suitable for such a case.

The Fog nodes considered in this work are homogeneous, which may not reflect reality. Relaxing this assumption of homogeneous nodes requires to consider greedy algorithms with optimization metrics other than the availability of green energy – metrics including a trade-off between the energy-efficiency of the nodes and local green availability. However, heterogeneity could be interesting to explore from an energy-efficient perspective. Indeed, the interest of producing green energy on a given location could vary with the computing capability and consequently, the energy efficiency of the nodes located nearby.

In this work, we consider on-site renewable energy production with sources owned by the Fog operator, as it is the case for large-scale Cloud providers [5]. Yet, as we have shown through simulation, exploiting local renewable energy sources in a low-power Fog architecture is complex, even when the geographic distribution allows for variable green production, and even with simple models (e.g. homogeneous nodes) that limits the hidden and cascade effects. While green energy produced but not consumed is considered as wasted here, other work explore alternatives like reselling it to the grid or sharing it through a virtual pool [4].

VI. Conclusion

On average, consolidation policy *Consolidate All* allows to reduce the energy consumption of the nodes by 12.4% compared to simulations without consolidation. But, due to the nature of consolidation, the responsiveness of the infrastructure is considerably reduced with, on average, 46.4% of the jobs placed on a booting node, and consequently delayed. This may seem to be an important issue considering that major goals of the Fog consist in being more responsive than the Cloud and reducing the end-to-end latency. However, the consolidation only impacts the allocation of new jobs, which will have to wait for the node to boot up and/or the VM to start. This behavior may be inadequate for jobs with a small lifetime, as the boot duration will be more impacting. But, for longer jobs, the problem can be considered differently. The initialization of a job will be longer, in average, to start on a node, but the real application latency will remain low during the job execution, and it will consume green energy.

We tried to allocate jobs on specific nodes to benefit at most of the green energy produced locally. But, even if the green production is important, a Fog node in itself does not consume much energy, even when fully loaded. In addition, in our Fog context without batteries, the nodes have low processing capabilities, reducing the number of jobs that can be hosted on a single node. We also found that, in our context, network in between the Fog nodes accounts for 53.6% of the total energy consumption, on average, although we do not rely heavily on the network as we do not use job migration for instance.

Distributed Fog architectures must be considered in the future. Many applications are envisioned to rely on their capabilities. The energy consumption of such architectures has to be considered to reduce their running cost and environmental impact. The optimization of green energy has not been conclusive in our performance-oriented context with small Fog nodes and without batteries. However, consolidation may be considered to reduce its consumption. It consequently decreases the energy consumption, but may not be adapted to every kind of application. Also, the consumption of the network between the Fog nodes should be taken into account as it represents the main energy consumption share.

As future work, we would like to explore the use of local energy storage devices in order to evaluate at which cost it would be feasible to have a Fog architecture completely autonomous in terms of electricity (as it is the case, at a different power scale, in wireless sensor networks with energy harvesting capabilities for instance). This may require the use of energy storage devices and prediction methods for green production.

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] Arif Ahmed et al. Fog computing applications: Taxonomy and requirements. *CoRR*, abs/1907.11621, 2019.

[2] Sherif Akoush, Ripduman Sohan, Andrew Rice, and Andy Hopper. Evaluating the viability of remote renewable energy in datacentre computing. Technical report, University of Cambridge, 2016.

[3] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing. *Future Gener. Comput. Syst.*, 28(5):755–768, May 2012.

[4] Benjamin Camus, Anne Blavette, Fanny Dufossé, and Anne-Cécile Orgerie. Self-Consumption Optimization of Renewable Energy Production in Distributed Clouds. In *IEEE Int. Conf. on Cluster Computing*, pages 1–11, 2018.

[5] Benjamin Camus, Fanny Dufossé, Anne Blavette, Martin Quinson, and Anne-Cécile Orgerie. Network-aware energy-efficient virtual machine management in distributed Cloud infrastructures with on-site photovoltaic production. In *Int. Symp. on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 1–8, 2018.

[6] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter. Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917, June 2014.

[7] L. Chiaraviglio, M. Mellia, and F. Neri. Energy-aware backbone networks: A case study. In *International Conference on Communications Workshops*, pages 1–5, 2009.

[8] Ismael Cuadrado-Cordero, Anne-Cécile Orgerie, and Christine Morin. GRaNADA: A Network-Aware and Energy-Efficient PaaS Cloud Architecture. In *IEEE International Conference on Green Computing and Communications (GreenCom)*, 2015.

[9] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang. Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption. *IEEE Internet of Things Journal*, 3(6):1171–1181, 2016.

[10] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan. Cachier: Edge-Caching for Recognition Applications. In *International Conference on Distributed Computing Systems (ICDCS)*, pages 276–286, 2017.

[11] Hughes Ferreboeuf and al. Lean ICT, pour une sobriété numérique. *Rapport intermédiaire du groupe de travail – The Shift Project*, 2018.

[12] Íñigo Goiri, Kien Le, Md. E. Haque, Ryan Beauchea, Thu D. Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. GreenSlot: Scheduling Energy Consumption in Green Datacenters. In *Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, 2011.

[13] Loic Guegan, Betsegaw Lemma Amersho, Anne-Cécile Orgerie, and Martin Quinson. A Large-Scale Wired Network Energy Model for Flow-Level Simulations. In *Int. Conf. on Advanced Information Networking and Applications (AINA)*, volume 926, pages 1047–1058, March 2019.

[14] F. C. Heinrich, T. Cornebize, A. Degomme, A. Legrand, A. Carpen-Amarie, S. Hunold, A.-C. Orgerie, and M. Quinson. Predicting the Energy-Consumption of MPI Applications at Scale Using Only a Single Node. In *IEEE Int. Conf. on Cluster Computing*, pages 92–102, 2017.

[15] T. Hirofuchi, A. Lebre, and L. Pouilloux. Simgrid vm: Virtual machine support for a simulation framework of distributed systems. *IEEE Transactions on Cloud Computing*, 6(1):221–234, 2018.

[16] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker. Fog Computing May Help to Save Energy in Cloud Computing. *IEEE Journal on Selected Areas in Communications*, 34(5):1728–1739, 2016.

[17] F. Kaup, S. Hacker, E. Mentzendorff, C. Meurisch, and D. Hausheer. Energy models for NFV and service provisioning on fog nodes. In *IEEE/IFIP Network Operations and Management Symp. (NOMS)*, 2018.

[18] Yunbo Li, Anne-Cécile Orgerie, and Jean-Marc Menaud. Balancing the use of batteries and opportunistic scheduling policies for maximizing renewable energy consumption in a Cloud data center. In *Euromicro Int. Conf. on Paral., Distr., and Network-Based Processing (PDP)*, 2017.

[19] Openfog consortium. out of the fog: Use case scenarios (visual security surveillance). https://www.openfogconsortium.org/wp-content/uploads/Video-Surveillance-Use-Case.pdf, 2018.

[20] Openfog consortium. process manufacturing in beverage industry. https://www.openfogconsortium.org/wp-content/uploads/Beverage-Industry-Short.pdf, 2018.

[21] Anne-Cecile Orgerie, Marcos Dias de Assuncao, and Laurent Lefevre. A Survey on Techniques for Improving the Energy Efficiency of Large-scale Distributed Systems. *ACM Comp. Surveys*, 46(4):47:1–47:31, 2014.

[22] Photovolta project. http://photovolta2.univ-nantes.fr/, accessed April 2020.

[23] Xueyan Tang, Changbing Chen, and Bingsheng He. Green-aware Workload Scheduling in Geographically Distributed Data Centers. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 82–89, 2012.

[24] B. Thomas, B. Close, J. Donoghue, J. Squires, P. De Bondi, M. Morris, and W. Piekarski. ARQuake: an outdoor/indoor augmented reality first person application. In *International Symposium on Wearable Computers*, pages 139–146, 2000.

[25] Pedro Velho, Lucas Mello Schnorr, Henri Casanova, and Arnaud Legrand. On the validity of flow-level tcp network models for grid and cloud simulations. *ACM Trans. Model. Comput. Simul.*, 23(4):1–26, 2013.

[26] Jon Whiteaker, Fabian Schneider, Renata Teixeira, Christophe Diot, Augustin Soule, Fabio Picconi, and Martin May. Expanding Home Services with Advanced Gateways. *SIGCOMM Comput. Commun. Rev.*, 42(5):37–43, 2012.

[27] R. Wolski and J. Brevik. Using parametric models to represent private cloud workloads. *IEEE Transactions on Services Computing*, 7(4):714–725, 2014.

[28] Y. Xiao and M. Krunz. Distributed Optimization for Energy-Efficient Fog Computing in the Tactile Internet. *IEEE Journal on Selected Areas in Communications*, 36(11):2390–2400, 2018.