
Master research Internship

Internship report

# Optimizing Green Energy Consumption of Distributed Cloud Architectures

**Domain: Distributed Computing - Networking and Internet Architecture**

*Author:*
Adrien Gougeon

*Supervisor:*
Anne-Cécile Orgerie
Benjamin Camus
IRISA - Myriads

**Abstract:** Cloud computing is now an essential component of the IT world. With the emergence of more and more smart and connected objects, the utilization of Cloud resources will continue to grow. 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 green energy has not been considered. The goal of this work is to exhibit an energy-efficient Fog architecture considering the integration of green energy.

# Contents

# 1 Introduction

Before the 2000s, companies which wanted to offer online services, had to host several servers in order to offer their services. However, this model has several issues and the biggest one is the intermittent usage of the services provided by the companies. To handle this intermittent workload, companies had to possess enough on-site servers to respond to the higher possible workload or they could face an issue in providing services to their end-users. On the other hand, when the workload is low, many of the servers required to handle a high workload are underutilized or even not utilized at all. Worse, when a company activity grows it has to extend its park of servers to respond to the new needs of its end-users. Another problem with having its own on-site servers is to hire people for their maintenance. The principle of Cloud computing emerged to tackle these issues.

The key idea with the Cloud is to rent resources centralized in a Cloud data center. An important feature of the Cloud computing is the elasticity of the provided resources. A Cloud user may increase or decrease its demand for resources dynamically, greatly reducing the impact of intermittent workload. However, the centralization of resources, i.e., machines, in a Cloud data center is not enough to improve the overall efficiency of the system. The Cloud computing paradigm has been possible and developed by using Virtual Machines (VM).

A VM emulates a physical system and many isolated VMs can be placed on the same physical machine. Resources given to a VM, i.e., mostly CPU and RAM, can be dynamically modified, thus reducing the need of over-provisioning. In addition, VMs can be dynamically migrated from a physical machine to another. The features offered by VMs greatly contributed to the fast development of Cloud architectures. The centralization of many servers and the utilization of VMs allow a simplified and far more efficient usage of the physical machines.

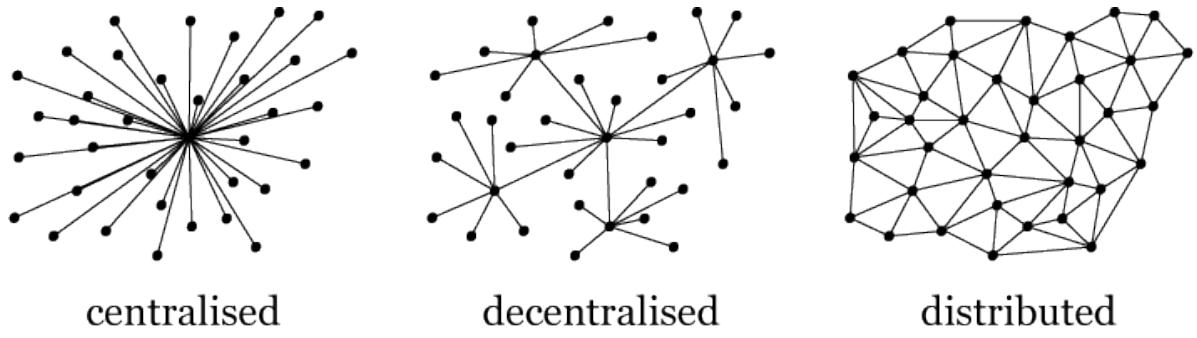


Figure 1: Classical network architectures. The Cloud is either centralized or decentralized while the Fog aims to be distributed[1].

Since the creation of the concept of Cloud computing, its underlying architecture has always been evolving. In the first stages of the Cloud, the most common design for a Cloud architecture was the centralization, as shown in Figure 1. This mono-site management is quite simple, every request is sent to the Cloud data center, processed locally, and sent back to the end-user. However, while such an architecture has simple management, it presents some issues:

- Single-Point-Of-Failure: in a mono-site system, if the site fall, everything is lost;

[1]From https://techcrunch.com

- bandwidth utilization: the network usage may encounter congestion if many services provided on-site have a high bandwidth utilization;
- latency: end-users may be far from the data center, causing unacceptable latency in the service provided.

Current architectures of Clouds tend to be decentralized, as depicted in Figure 1. In this architecture, there are multiple, smaller, data centers answering end-users requests. Data centers communicate between them to share the workload. The overall management of a decentralized architecture might be centralized or decentralized. A centralized management means that a data center is the leader of the others. This solution tackles latency and bandwidth issues of centralized architecture, but still has a Single-Point-of-Failure with the leader. The management can be decentralized too, meaning the absence of a main coordinator. Such management is more resilient but is more challenging to operate. Sharing information across multiple sites is more expensive in terms of time and energy, in consequence a local optimum is preferred over a global optimum for resource allocation.

While decentralized architectures present good advantages over centralized ones, some emerging applications require extremely low latency. The Fog aims to respond to this new kind of services. The concept of Fog is still recent and there is no consensus on its definition. We consider it as a distributed architecture, as presented in Figure 1. In this model each node, or Point-of-Presence, receives and processes local or close data without global coordination. The Fog does not intend to replace the Cloud but to complete it. The Cloud is useful for long term storage and aggregation of data, and the Fog intends to extend the services offered by the Cloud closer to the end-user.

The usefulness of the Cloud paradigm has been proven along the years. More and more services rely on the Cloud and Cloud providers continue their expansion and deployment of more data centers. This growth is foreseen to continue with the emerging Internet-of-Things and Fog paradigm.

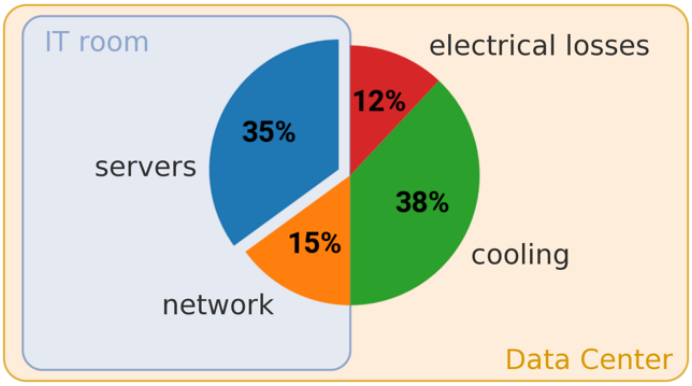


Figure 2: Energy consumption of a data center[2] [21].

This growing importance of the Cloud paradigm has been the source of many research works to reduce its costs and improve its profitability. Energy consumption has always been an important part of the running cost. It is becoming a limiting factor to build more data centers and is also becoming dominant in the Total Cost of Ownership (TOC) [22]. Figure 2 shows a distribution

[2]Courtesy of David Guyon

of the consumption inside a typical data center [19]. 12% of the consumption is due to electrical losses caused by the transport of the energy and its conversion. An important part of the energy is consumed by the cooling system. Indeed, cooling is essential in a data center to improve the lifetime of the components. Finally, the major part of the energy is consumed inside the IT room, i.e., to run machines and network devices. Figure 3 presents a general idea of the energy hungry components in a typical server machine. The CPU usage is clearly the major consumer [19].

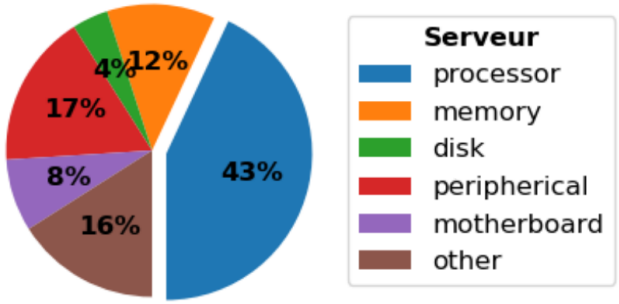


Figure 3: Energy consumption of a typical server machine[2] [19].

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 [13]. Regarding the utilization phase of IT, data centers were responsible for 32% of the global consumption of the sector in 2017. In addition, the network, which is crucial with the notion of Cloud computing, was responsible for 34% of this consumption [13]. 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, or by considering the usage of green energy. These works present noticeable advancements, according to the current architectures of the Cloud. However, the emerging Fog paradigm, still conceptual at the moment, is supposed to be deployed in the near future. 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 in account its energy consumption.

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. The main tackled challenges are:

- to schedule jobs in a fully distributed architecture;
- to deal with the intermittent production of energy from renewable sources;
- to improve the energy consumption of the Fog without impacting severely its benefits in terms of latency.

The remainder of the paper is structured as follows: Section 2 presents related work; Section 3 details the employed models; Section 4 describes our approach; Section 5 presents the experimental setup and the results; Section 2.3.3 concludes this work and introduces future work.

# 2 Related work

Energy consumption and environmental impact of Cloud architectures is an increasingly important problematic. In this part we present previous work conducted to tackle those issues. Firstly, we consider research work about reducing energy consumption in Cloud architectures. Secondly, we present research work aiming to improve the exploitation of green energy in Cloud architectures. Finally, we introduce research work considering energy consumption in Fog architectures.

## 2.1 Reducing Overall Consumption in Cloud Architectures

Cloud computing has become necessary to support and develop the new face of the IT world. End-users ask for more and more availability, low latency, and variety of services, causing the underlying architectures of the Cloud, network and data centers, to expand quickly. This growth induces an inevitable increase of energy consumption of these infrastructures, a raise becoming problematic in terms of cost for Cloud providers and for the environment [13, 22].

This section presents research work conducted to reduce the energy consumption in Cloud computing.

### 2.1.1 Energy Models of Cloud resources

The energy consumption of a physical machine has two parts: a first part is static and a second part is dynamic. The static part do not depend on the load of the machine, as opposed to the dynamic part which depends on the load. On average, an idle server can consume up to 70% of the power consumed by the server running at the full CPU speed [5].

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 [22]. Therefore, a simple solution to reduce energy consumption is to switch-off unused machines. In addition, VMs on under-utilized machines can be migrated to switch-off under-utilized machines, reducing again the static cost of machines. However, migration increases energy consumption as both source and destination machines are running while migration occurs. This scheduling problem of VMs on servers is known as consolidation, and is detailed in Section 2.1.3.

Network devices relaying information in the network are subject to the same principle, as depicted in Figure 4. Worse, the static part, in the case of network equipment, can reach more than 90% of its maximal consumption [17]. Cuadrado-Cordero and al.[10] explore a solution to reduce the consumption in the network, presented in Section 2.1.4.

### 2.1.2 Hardware Improvement

Cloud data centers host a great number of machines to fulfill their duty. As presented in introduction, the consumption of these machines represent an important part of the overall energy cost in a Cloud data center. A possible optimization to reduce the environmental impact of data centers is to improve the various hardware components of the machines to make them more energy-efficient. The main impact of this optimization happens when acquiring new equipment. This work does not intend to treat this subject. Instead, it focuses on the management of Cloud data centers to reduce their energy consumption.

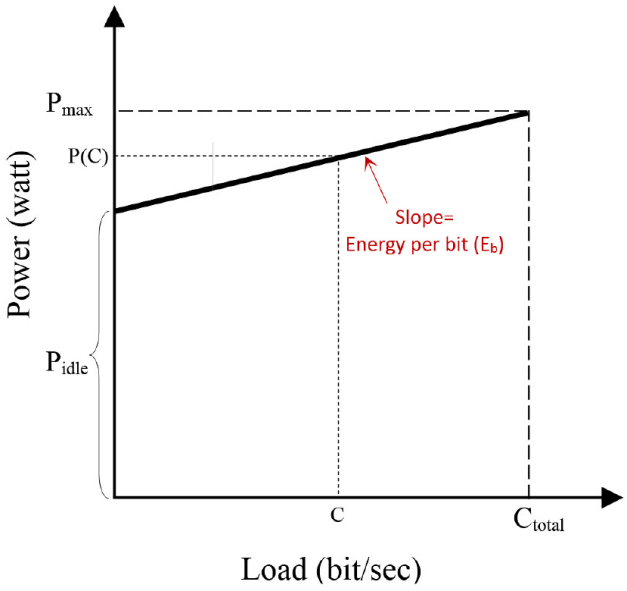


Figure 4: Power consumption trend versus load [17].

### 2.1.3 Consolidation

Consolidation consists in migrating VMs placed on under-utilized servers to others servers, in order to keep a minimum number of servers powered-on. However, Cloud architectures owe their fast development to their scalability and elasticity. Consolidating servers reduces energy consumption, but may lead to an increase in response time, as physical machines do not switch-on instantly. Cloud providers must respect Service-Level Agreements (SLA) and ensure a minimal response time and overall Quality of Service (QoS). A rupture of SLA leads to heavy financial compensations and should be avoided. Cloud data centers can keep servers not fully utilized or some empty servers switched-on to respect QoS constraints.

Beloglazov and al. [5] presented a work to reduce energy consumption with consolidation, while respecting of SLA, 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. This elasticity allows VMs to grow, i.e., ask for more CPU power, without having to migrate to another server with sufficient remaining CPU power.

Firstly, they consider the initial placement of new VMs. The placement of VMs can be seen as a bin packing problem. Best Fit Decreasing is a well-known greedy algorithm providing a solution with $11/9 * O + 1$ bins, where $O$ is the optimal number of bins. In their work they present a Modified Best Fit Decreasing (MBFD) algorithm, as detailed in Algorithm 1. The MBFD algorithm sorts VMs by decreasing utilization and then finds the server (host) on which each VM would consume the less energy.

Secondly, they detail the dynamic placement of VMs to ensure the thresholds policy. If the lower threshold is reached on a physical machine, then all VMs present go through the MBFD algorithm while the original host is switched-off. Conversely, if the upper threshold is reached, only

**Algorithm 1:** Modified Best Fit Decreasing (MBFD) [5]

```
1  Inputs: hostList, vmList
2  Outputs: allocations of VMs
3  vmList.sortDecreasingUtilization()
4  foreach vm in vmList do
5  |  minPower ← MAX
6  |  allocatedHost ← NULL
7  |  foreach host in hostList do
8  |  |  if host has enough resource for vm then
9  |  |  |  power ← estimatePower(host, vm)
10 |  |  |  if power < minPower then
11 |  |  |  |  allocatedHost ← host
12 |  |  |  |  minPower ← power
13 |  |  |  end
14 |  |  end
15 |  end
16 |  if allocatedHost ≠ NULL then
17 |  |  allocate vm to allocatedHost
18 |  end
19 end
20 return allocation
```

a part of the present VMs must be migrated. Their VMs migration policy aims to minimize the number of migrations. To this end, they tend to migrate VMs with the highest CPU utilization.

As often, the solution is a question of compromise. Higher thresholds reduce energy consumption but induce more SLA violations. In comparison to Non Power Aware (NPA) and Dynamic Voltage and Frequency Scaling (DVFS) policies, their solution reduce the consumption by 84% and 66% respectively. However, this reduction in consumption induce 1.1% of SLA violations.

Consolidation reduces significantly the consumption in Cloud environment and is often considered in addition to other contributions. It exist many various consolidation policies and algorithms but the principle of switching-off underutilized machines is always the same.

### 2.1.4 Optimizing Network consumption

Cloud architectures are oriented around big Cloud data centers, often far from the end-user. Information is sent from everywhere to a specific data center, then processed, and finally sent back. The network is already a considerable energy consumer, more than data centers themselves. Considering the utilization phase, it was responsible for around a third of the global consumption of the IT sector in 2017 [13]. The Cloud paradigm causes an increasing use of the network and might cause an excessive waste of energy when the source and the destination are close. For example, when two coworkers are working on the same online shared file.

In that sense, a semi-decentralized Platform-as-a-Service (PaaS) architecture, GRaNADA [10], has been proposed to reduce energy consumption in the network. The main objective of GRaNADA is to close the gap between the host of the information and the end-user. This modification in

the network reduces the number of hops, decreases latency, and reduces the number of network interfaces needed, e.g., routers and switches. Those devices can be switched-off to save energy. In fact, it offers a solution similar to a Fog architecture by bringing services closer to the end-user.

GRaNADA distributes the computation between the users of the Cloud and the network devices close to them, if available. In this approach, some end-users are providers, hosting information, while others are normal end-users. Within the same group (named microcloud) only one end-user is the host and deploys a light virtual machine. End-users communicate between them instead of communicating with a Cloud data center. This policy reduce the need for network equipment, and, as only one end-user is a host, network flooding is also reduced. In addition, the authors developed a routing protocol selecting the greener path between end-users, reducing again network consumption.

Experimentation with real traces shows that GRaNADA saves up to 75% of the network consumption compared to a centralized approach. It is also more energy-efficient compared to another popular semi-decentralized approach with nano data centers [27]. A limitation of this approach is its reduced efficiency with a large geographical distribution of end-users working on the same information. In such configuration, the problem is not the distance between end-users and the Cloud, but the distance between end-users themselves.

### 2.2 Exploiting Green Energy in Clouds

The classical energy delivered to infrastructures is known as brown energy. This energy is produced by non-renewable sources, such as nuclear power plants and coal centrals. It is opposed to green energy, produced by renewable sources, such as solar panels and wind turbines. Brown energy has a cost on its utilization, depending on the amount of energy consumed and the pricing of the provider. On the other hand, the green energy produced on-site can be freely used, or even sold through the electrical grid operator. However, on-site green energy production has an important initial cost, due to the installation of the collectors, i.e., solar panels or wind turbines.

Green energy represents a more cost-efficient solution in the long term and its use reduces greatly the environmental impact of the infrastructures. In addition, the current global environmental issue and the increasing energy consumption of the Cloud are strong incentives for data center operators to rely on green energy. Major Cloud providers such as Amazon, Apple and Microsoft have already started to build solar panels facilities [6].

Green energy represents a great alternative to reduce energy consumption from the regular grid and environmental impact for Cloud providers. However, its utilization faces a major issue: green energy production is intermittent. Green energy production falls dramatically if coming from wind turbine when there is not enough (or too much) wind, or during night when coming from solar panels. Batteries can be used to deal with the intermittent production and to store energy for later usage. This solution allows to reduce the waste of green energy, when it cannot be consumed at production time. Yet, batteries present some disadvantages:

- their purchase and maintenance costs are important;
- due to internal resistance and self-discharge batteries have important energy losses;
- chemicals used to produce batteries are harmful for the environment.

Considering aforementioned problems we do not investigate battery-based solutions.

Due to the intermittent production, it is clear that green energy cannot be used alone and is complementary to brown energy. We will present research conducted to optimize the usage of green energy over brown energy in data centers. Firstly, we present solutions to optimize green energy consumption in centralized Cloud architectures, i.e., mono-site Clouds. Secondly, we discuss solutions in decentralized architectures, i.e., multi-site Clouds.

### 2.2.1 Mono-site Clouds

The green energy production is intermittent. If we consider that the production is sufficient when available, then a simple solution is to delay jobs until green energy is available. However, an essential notion to keep in mind is the deadlines of the jobs. In classical Cloud data centers, most submitted jobs are executed as soon as possible. But, if jobs can cope with a delay, a specific scheduling policy can optimize the green energy consumption.

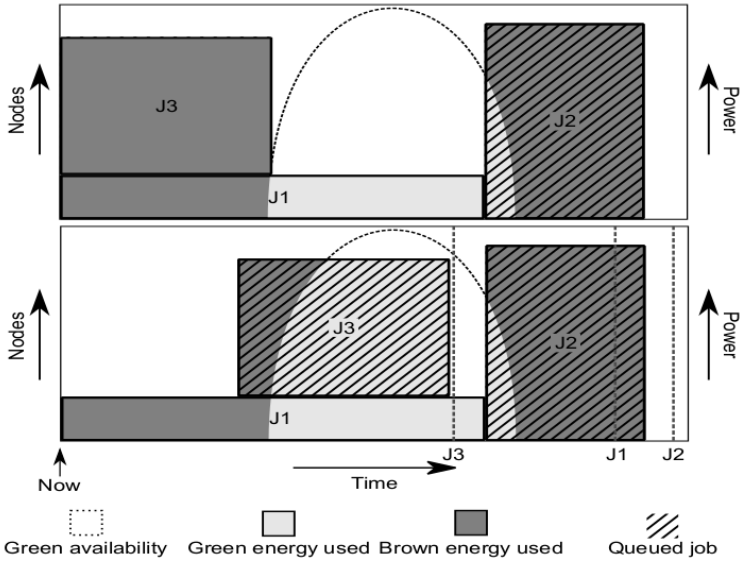


Figure 5: Scheduling 3 jobs (J1-J3) with classical backfilling (top) and Green-Slot (bottom). The deadlines of the jobs are the vertical lines [14].

GreenSlot [14] presents a solution optimizing green energy consumption in a data center while respecting deadlines of submitted jobs. Figure 5 presents the principle of GreenSlot. With a classical backfilling algorithm (top), jobs are scheduled as soon as possible, causing a waste of green energy and an overuse of brown energy. Conversely, GreenSlot (bottom) delays jobs to reduce the waste of green energy, while respecting deadlines. The model is focused on reducing the cost of energy and green energy is assumed to have a zero cost (on-site own production). In contrast, brown energy has a variable cost, and when deadlines cannot be met using only green energy then consumes brown energy when it is the cheapest. Jobs are greedily scheduled according to their slack, i.e., remaining time before deadline is reached. To validate their model they used three real scientific computing workloads. Each workload implements a different pipelined approach to the sequencing and mining of the genome of a bakers yeast.

Results show a significant increase in the green energy consumption by up to 117% while reducing the overall energy cost by up to 39%, compared to a standard back-filling scheduler, as of

2011.

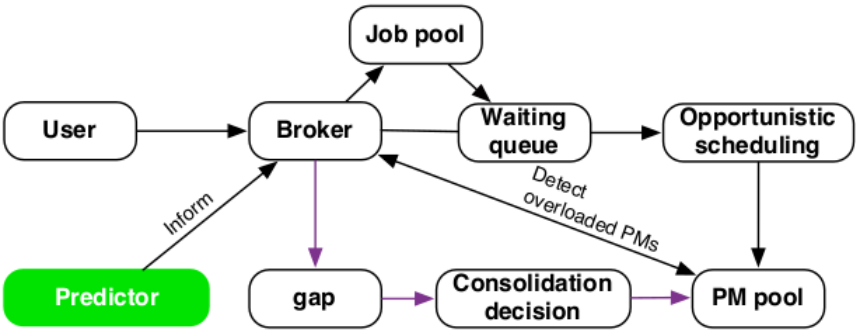


Figure 6: PIKA overview [20].

The scheduling policy of GreenSlot is static, in consequences, if green energy is running out, e.g., due to thick clouds or rain when relying on solar panels, jobs will continue using brown energy. PIKA [20] extends the delaying idea of GreenSlot by adjusting dynamically the workload to the available green energy. The principle of PIKA is depicted in Figure 6. Users submit jobs to the broker. Jobs are pooled and put in a waiting queue to be scheduled later. The broker, according to the available energy in the next time slot given by the predictor, decides whether or not to call the gap function. The gap function determines consolidation decisions. If green energy is not sufficient some physical machines are switched-off and jobs are pushed to the job pool through the broker. If green energy is plentiful, some physical machines are switched-on. The opportunistic scheduler deploys jobs from the waiting queue when physical machines are available.

Compared to a typical energy-efficient (but not renewable-aware) management of the data center and using real traces, PIKA [20] increases the green energy consumption by 110% and reduces the brown energy consumption by 44.6%. However, PIKA increases the overall energy consumption by 31%. This is due to its aggressive politic on consolidation, inducing a significant number of VM migrations compared to a classical energy-efficient management where VM are migrated only when physical machines are overloaded. Nevertheless, the extra energy consumed is green.

Green solutions in mono-site Cloud often rely on delaying the jobs to meet the production of green energy. This kind of solution is great for batch jobs with slack, but is opposed to the fundamental rules of response time and elasticity of the Cloud. However, PIKA considers uninterruptible jobs along with batch jobs in its model and still present significant brown energy savings.

### 2.2.2 Multi-site Clouds

Decentralized Clouds are geographically distributed over a large area to provide better QoS. 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. [25] 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. Each job submitted to the

scheduler is added to a job pool. If the slack time of a job in the pool reaches its minimum then, the algorithm pushes immediately the tasks of the job to the data center where it will consume the less brown energy. At every time step (15 min), a prediction of the available green energy at each data center is updated. According to this prediction, jobs are dispatched over data centers. The algorithm chooses data centers on which jobs will consume the less brown energy considering their available green energy and cooling constraints. To reduce overall consumption, consolidation is applied when a task ends. Experimental results show a reduction up to 40% of brown energy consumption compared to a Round Robin algorithm. However, this work only considers jobs with slack, i.e., scientific workload and high performance computing tasks in particular. In addition, the placement of the tasks is static.

Another work [4] proposes a dynamic scheduling of the running tasks. The goal of this work is to rely at 100% on green energy. Tasks are scheduled on data centers with enough green energy to compute them. But, if the green energy production falls under a specific threshold, tasks are either moved to a data center with sufficient green energy production or halted. Results show that data centers would need many more machines to host tasks from data centers where the production of green energy is insufficient (50% more in their example). Otherwise, the overhead on the delay of tasks becomes problematic. In addition, this work targets jobs with consequent slack, as there is no deadline considered. It also induces a great number of VM migrations, as jobs are migrated every time the green production is not sufficient.

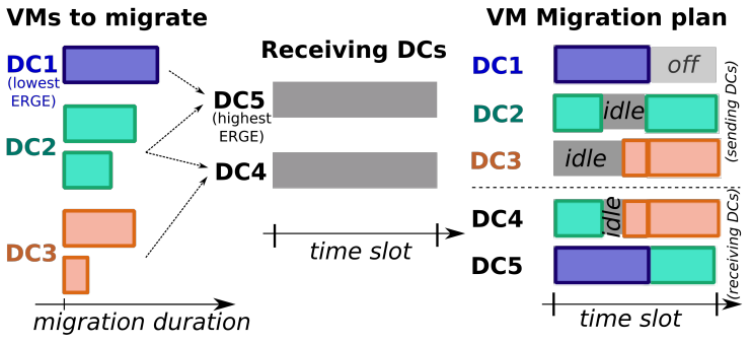


Figure 7: Example of the migration algorithm of NEMESIS with 5 data centers (DC) [7].

The great number of VM migrations might create network congestion, in addition to the increase in energy consumption. NEMESIS [7] presents a network-aware solution to optimize green energy consumption between multi-site Clouds, and to reduce overall energy consumption. Figure 7 depicts the migration algorithm of NEMESIS. Data centers with low expected remaining green energy (ERGE) send workloads to data centers with high ERGE, in order to optimize the consumption of green energy. The VM migration plan is built to avoid congestion at the sending and receiving data centers. In addition, this work considers consolidation optimization and on/off switching cost for the servers. NEMESIS is compared to others VM placement and migration algorithms on real traces. It is notably compared to the MBFD algorithm presented in Section 2.1.3. Simulation results show that, in addition to increasing the ratio of green energy over brown energy, NEMESIS reduces the overall energy consumption. This improvement is mainly because NEMESIS takes into account the energy cost to migrate VMs and their remaining computation time to avoid wasteful migrations.

In addition to network congestion, VMs migration induces an overhead on network consumption.

The authors of SCORPIUS [6] propose an alternative to optimize green energy consumption over multi-site Clouds other than VM migration. Yet VMs migration is still considered, the objective of SCORPIUS is to share green energy between the data centers with the help of Smart Grids. Smart Grids could allow to share energy between multiple sites across a country. Data centers producing green energy, but without the possibility to use it, can inject it into the grid, creating a virtual pool of green energy.

The model of SCORPIUS considers the cheapest solution between migrating VMs to another data center or taping the virtual pool for green energy. Green energy is considered as shared when at the same moment a data center is injecting green energy and another one is taping energy from the pool. SCORPIUS is efficient to optimize the green energy consumption, but current Smart Grids are still limited in size and their management at larger scale is still an open issue. Another limitation of SCORPIUS is that it does not consider inter-grid exchanges. It might be an issue in countries with multiple energy providers and thus, multiple Smart Grids like in the U.S.

## 2.3 Energy Consumption in the Fog

Current architectures constituting the Cloud have proven to be efficient. The properties of scalability and elasticity have made it the major infrastructure to process information. Yet, the emerging Internet-of-Things brings new challenges to reduce the latency and growing bandwidth utilization. New real-time applications require to process considerable quantities of data with an extremely low latency, a latency unreachable considering the current architectures. For example, latency is important in applications such as remote medical treatment, collision avoidance for autonomous vehicles or virtual and augmented reality.

The Fog aims to respond to this demand by bringing the Cloud closer to the end-users. Its goal is to complete the Cloud. However, the Fog computing paradigm is still in its early stages and its definition is not clear among the community. For example the devices which would support the Fog (the point-of-presence) differ among the papers debating on this subject. Nevertheless, the Cloud already faces important energy issues and the Fog will surely bring new challenges. The network architecture supporting the Fog will be less efficient and reliable than in a data center, hardware devices will be more heterogeneous and the distributed architecture implies new management policies.

This section introduces some visions of the Fog, how it would extend the Cloud, and how it would impact the energy consumption. Section 2.3.1 presents approaches considering the energy consumption and efficiency of the Fog. Section 2.3.2 describes how it could improve the energy consumption of the current Cloud architecture.

### 2.3.1 Energy-aware Fog

Several works have introduced some principles to implement the Fog, but few took in consideration its energy consumption. Deng and al. [11] present a vision of the Fog-Cloud interplay, while taking into account the trade-off between power consumption and latency. Their work is guided by the idea to make the Fog as an intermediate layer between the end-user and the Cloud, extending the Cloud closer to the end-user. Their architecture is depicted in Figure 8. End-users submit requests through a front-end portal, this input is forwarded to the closest Fog device using local area network (LAN). 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 Fog devices are light-weight Cloud servers with large data storage and correct computation capabilities.

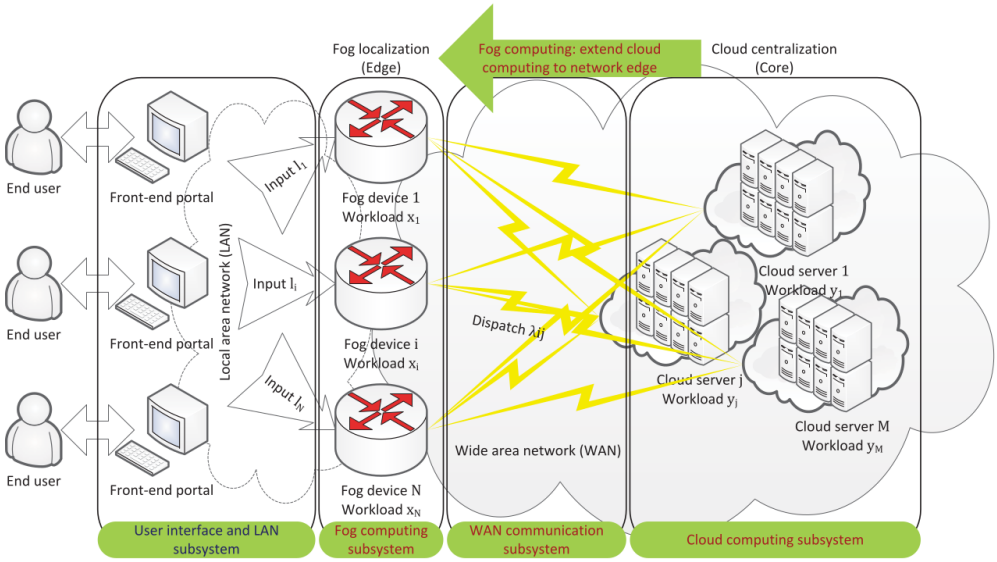


Figure 8: Overall architecture of a Fog-Cloud computing system with four subsystems and their interconnections/interactions [11].

They build a model representing the differences in latency and power consumption in the Fog and in the Cloud. In this model, communication delay is considered negligible in LAN compared to WAN. The processing delay in the Cloud is far better than in the Fog, but does no cover the communication delay.

Solving their model brings the conclusion that keeping workloads on Fog devices reduce latency by diminishing communication delay between users and processing machine. However, due to the inefficiency of Fog devices, it increases overall power consumption. This work introduces the principle of energy-aware Fog, but results are coarse-grained. They state to use simulations with numerical results, but no unit of time nor power are shown.

Xiao and al. [31] introduce a Fog architecture aiming to reduce latency while considering energy-efficiency of the Fog devices and evaluated with concrete simulations. The considered nodes are smaller than in the previous architecture and may be easily overloaded. These nodes, while being smaller than typical Cloud servers, still face the idle power issue presented in Section 2.1. As a result, the power efficiency of a node increases with the workload: as more requests are processed per unit of time, less energy is consumed per request. However, the computation delay grows with the amount of submitted workload. In fact, to optimize the power consumption of a node, the overall delay should be at the closest of the deadline. To optimize the power consumption and delay in the grid of Fog nodes, their idea is to share workload between close nodes. Their model shows that, even if there is a high constraint on power consumption, the delay can be respected, if the workload is shared with others nodes with more delay tolerant workloads. Conscious of the overhead caused by the sharing of information such as available power, and to obtain a scalable architecture, their model is distributed, with decisions based on local information.

To support their contribution, they simulate their architecture on a case study with several

self-driving buses. The simulation is based on videos generated from street view images of real bus routes, in Dublin. To retrieve useful information from these videos, frames need to be processed. They consider a network of Fog nodes spread around the bus routes. Each bus sends its frames to the closest Fog device. Then, they consider three cases: Fog devices cooperate between each other in a 500m radius, or Fog devices cooperate only with the closest other Fog devices, or they do not cooperate at all.

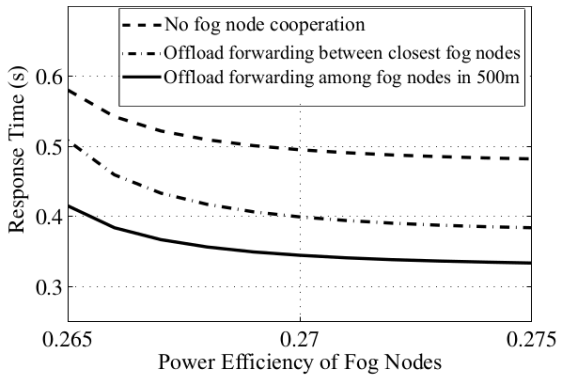


Figure 9: Average response time with different power efficiency [31]

Results of this experiment are presented in Figure 9. In their work, they define power efficiency as the amount of power consumed to process a unit of workload. In consequence, a node with a low power efficiency will consumes less energy to process a unit of workload than a node with a high power efficiency. The cooperation between nodes, i.e., offload forwarding, improves the response time by allowing nodes to treat more frames in average. This is really interesting in terms of scalability and distributed architecture management. Their work shows that the offload forwarding provide important benefits in latency, even when the power efficiency is low. The notion of small Fog devices cooperating without centralized management is an important step forward, and shows an interest in terms of latency. However, the difference in consumption with a classical Cloud infrastructure is not represented.

### 2.3.2 Saving Energy with the Fog

The Cloud has a considerable and growing energy cost. The Fog will rely on much smaller devices and is supposed to be less energy-efficient. However, some research works propose Fog architectures that could have a better impact on the energy consumption than the Cloud already has.

An approach more energy-efficient than the Cloud is explored through the use of Nano Data Centers (NDCs) [17]. As part of a Fog architecture, NDC are small devices (Raspberry PI), at the premises of end-users. In this architecture, the Cloud data centers are replaced by NDCs hosting all the information. NDCs are supposed to be more energy-efficient than data centers for specific usages and applications, not in the general case. The authors rely on a flow-based and a time-based energy consumption model to compare the energy consumption of NDCs and data centers. The flow-based model, built for equipment shared by many users such as routers and switches, is based on the average energy per bit of the traversed equipment. The time-based model, built for devices not shared , such as NDCs and terminals, is based on the ratio of idle time over active time of the

equipment. These models are built to compare the energy consumption of a specific service in the two architectures.

Three kinds of applications are compared:

- applications with static content: applications were data source is close to the end-user and static, such as hosting a website;
- applications with dynamic content: applications were data source is close to the end-user and dynamic, such as video surveillance;
- applications requiring pre-loading: applications were data is not close to the end-user and must be pre-loaded to closer nano server, for example applications doing Video-on-Demand.

In the first case, results do not put an architecture above another. In fact, the quality of comparison depends vastly on the energy-efficiency of the data centers compared with NDCs. In the second case, results show that a dynamic application, with a number of uploads greatly above the number of downloads, consumes much less energy in the NDC architecture. This is due to the cost of upload which is negligible when uploading to a local NDC and as important as a download when uploading to a data center. In these two cases, for the NDC architecture, the downloads are done at 80% by non-local peers, i.e., downloads go through the core network, as in the data center architecture. The last case is not significant as results from NDCs are not compared with results from data centers. The conclusion of this work is that NDCs are more energy efficient than data centers for large amount of data rarely accessed, such as video surveillance.

Research works have shown the Fog to be not only necessary but also feasible. While its description is still blurry, there is a clear need for an architecture closer to the end-user, notably due to latency constraints. Few research works considering its energy consumption have been carried out, but the usage of green energy in a Fog architecture has not been considered to the best of our knowledge.

### 2.3.3 Conclusion on Related Work

The Cloud is a well implanted solution and provides consequent advantages in terms of scalability and elasticity. More and more services rely on it, contributing to its success. This success leads to a growing energy consumption of data centers and network infrastructures. This increasing consumption is becoming a problem to maintain the development of the Cloud. Moreover, it has a considerable environmental impact, as most of the energy consumed comes from non-renewable sources. Many research have been conducted to either reduce the overall consumption of the Cloud or improve its use of green energy over brown energy. Those solutions have contributed to pursue the development of Cloud infrastructures.

Although powerful and efficient, the Cloud has some limitations in terms of latency and network utilization. To tackle these limitations, the principle of Fog architecture emerged, with the idea of bringing the power of the Cloud closer to the data production sites. While the Fog virtualized environment will be similar to what can be observed in the Cloud, it differs on many points. The Fog will supposedly rely on a network architecture less efficient than the Cloud. In addition, its decentralized nature will imply more communication between the nodes, leading to overloads and congestion without proper management. The nodes supporting the Fog may be heterogeneous and with limited capacity, making optimization in this environment more complex than in the Cloud.

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 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, investigating current solutions applied in the Cloud. To the best of our knowledge, such combination has not been investigated yet, and could have beneficial impacts on the sustainability of future Cloud architectures.

# 3 Considered Fog Model

This section details hypotheses and design choices made in this work. We detail the Fog architecture, then the considered green energy production, the computing nodes, and the job submission. Finally, we recall the problematic considered over this model.

## 3.1 Fog Architecture

There are two main Fog architectures that stand out in literature. The first one is a fully connected Fog, as shown in Figure 10. A fully connected Fog typically refer 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 a realty augmented video game [26] or coordinating machines in a factory [3].

The second one is a hierarchical Fog, as shown in Figure 11. A hierarchical Fog allows to cover a larger area. This kind of architecture is adapted to applications looking for aggregating data over a wide area, such as video surveillance applications [2] or balance the load between Cloud and Fog, such as low-latency image-recognition applications [12].

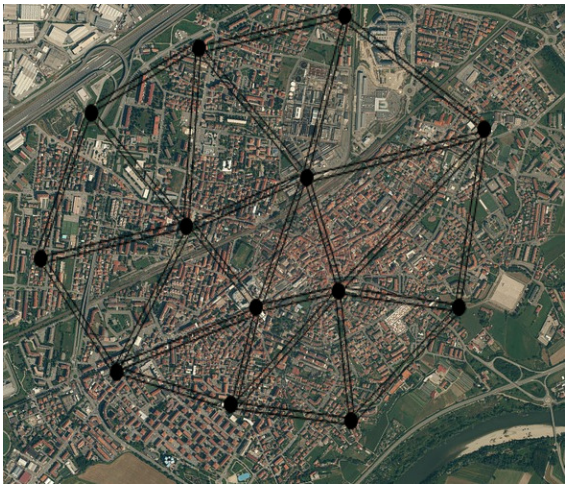

Figure 10: Example of a fully connected Fog over a city.

Both architectures are adapted to different kinds of applications and may be connected to each other. In this work, we focus on a hierarchical fog because the intermittent production of green

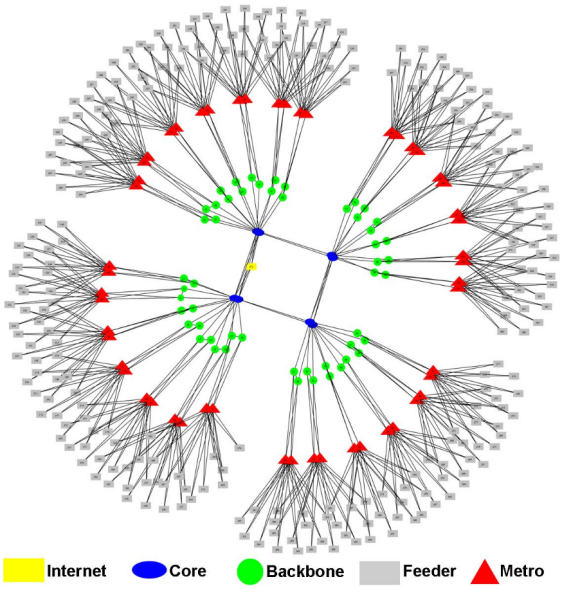


Figure 11: Hierarchical network topology.

energy can be mitigated if we consider a large area. More precisely, 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 [29].

### 3.2 Computing Nodes

As mentioned in Section 3.1, we consider a model of a hierarchical Fog architecture. In this model, Fog nodes are hosted by end-users and connected through the hierarchical ISP network (feeder, metro, backbone and core network layers).

All nodes are considered identical, with the same number of cores and power profile to ease the understanding of scheduling policies' behavior. The power model used is described in Equation 1. This power model is a simplification from a paper by Kaup and al. [18]. In this model, nodes have a static consumption based on their idle consumption, Ethernet consumption and WLAN consumption; and a dynamic consumption based on the CPU usage.

$$P = P_{idle} + P_{cpu}(u) + P_{eth,idle} + P_{wlan,idle} \tag{1}$$

In addition to this power model, nodes can be switched off. Our model takes into account the necessary duration to boot up or shutdown a node. In addition, while booting or shutting down, the consumption is modified to reflect the actual device behavior.

Nodes contain a virtualized environment. In this environment, a VM should be created to host a 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 cores it requires. Cores are not shared between VMs (i.e. we do not consider over-commitment here as it would modify the performance and energy consumption of jobs).

### 3.3 Energy Production

Concerning the power supply, we need to distinguish two kinds of production: brown energy and green energy. Brown energy comes from non-renewable sources, such as nuclear power plants and coal power plants. Conversely, green energy comes from renewable sources, such as wind turbines and photovoltaic panels.

The nodes are the only entities consuming energy in our model. Each node is connected to the electrical grid. The electrical grid is considered to provide only brown energy. Although nodes are all linked to the electrical grid, some of them can also consume on-site green energy. Indeed, some nodes are considered as green energy producers and are connected to photovoltaic panels, allowing them to produce and consume green energy. 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. The nodes can only consume the green energy they produce (no energy exchange among nodes).

### 3.4 Job Submission

The main objective of our Fog architecture is to process jobs. Conversely to other Cloud-oriented work presented, we consider a fully-distributed environment, which means that jobs are not submitted to a scheduler. Each new job is sent from an IoT device or end-user to the closest node of the Fog. Jobs are continuously submitted to the platform and are 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 as it is typically the case for video-stream decoding applications for instance [29].

### 3.5 Our Problematic

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 allocation and consolidation algorithms to optimize the energy consumption, brown or green, of such an architecture. In particular, we play with the latency constraint to determine if its release can bring significant improvements in terms of energy consumption.

## 4 Approach

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.

We first explicit three allocation algorithms used to select a host for a job submitted to a specific node:

- Lowest Latency;
- First Green;
- Most Green.

In a second part, we detail three consolidation policies used to consolidate the Fog workload:

- Consolidate All;
- consolidate Brown;
- Consolidate Ratio.

The energy efficiency of these algorithms is explored in Section 5.

## 4.1 Allocation Algorithms

In this section, we explicit three algorithms used to place jobs submitted in the Fog. To place jobs, the criteria are the latency and the green energy produced by the nodes. All three algorithms are greedy by latency in their search for a host. When a suitable host is found, the amount of cores required by the job are reserved and a VM is created on the host. When the job is finished, the VM is destroyed and the cores are freed. 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.

### 4.1.1 Lowest Latency

The first algorithm, Algorithm 2, only 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 initial node is added to an array of nodes kept sorted by increasing latency. As long as the first node of the array is not able to host the job, we remove the first node of the array and add to the array the direct neighbors of this node. If we found a node that has enough available cores to host the jobs, this host is returned by the algorithm. As the array is kept sorted by latency, the returned node is the best choice in terms of latency. To avoid circular loop in the search, nodes added to the array are marked. Marked nodes cannot be added again in the array, and these nodes are unmarked at the end of the search.

This algorithm is expected to provide the overall best latency of the explored algorithms as it will place jobs at the closest of where they have been submitted.

### 4.1.2 First Green

This second algorithm, Algorithm 3, aims to find a host with the lowest latency, but producing 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 its goal is to find a node producing green energy.

We consider returning a node producing green energy as a weak constraint, otherwise it could cause an extremely inefficient search. For example, if no node produces green energy, then the algorithm would explore the whole architecture for nothing. In addition, even if it founds 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 must have a

**Algorithm 2:** Lowest Latency

```
1  Inputs: initial_host, job_cores
2  Outputs: suitable host for the job
3  potential_hosts + = initial_host
4  host = potential_hosts[0]
5  while host ≠ NULL do
6  |  if host→spare_cores ≥ job_cores then
7  |  |  return host
8  |  end
9  |  else
10 |  |  potential_hosts.pop()
11 |  |  potential_hosts + = host→neighbors
12 |  |  host = potential_hosts[0]
13 |  end
14 |  potential_hosts.sort()
15 end
16 return NULL
```

latency below a specific threshold. This threshold is passed as a parameter of the algorithm and will be exceeded only if we 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 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 is no more nodes in the array: return *brown_host*;
- a node with enough free cores and producing green power has been found: return it;
- the maximum latency has been exceeded and we have found a *brown_host*: return it;
- no node can host the job: return $NULL$.

As in Algorithm 2 nodes added to the array are marked to avoid circular search.

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

#### 4.1.3 Most Green

The third algorithm, Algorithm 4, is similar to the second one (Algorithm 3). The only difference is that it searches for the node producing the most green power in its latency range, and will not stop at the first node producing green energy.

**Algorithm 3:** 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
```

**Algorithm 4:** Most 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  green_host = NULL
6  best_green_power = 0
7  host = potential_hosts[0]
8  while host ≠ NULL && (host→latency ≤ max_latency || brown_host == NULL) do
9      if host→spare_cores ≥ job_cores then
10         if host→green_power > best_green_power then
11             best_green_power = 0
12             green_host = host
13         end
14         else
15             if brown_host = NULL then
16                 brown_host = host
17             end
18         end
19     end
20     potential_hosts.pop()
21     potential_hosts + = host→neighbors
22     host = potential_hosts[0]
23     potential_hosts.sort()
24 end
25 return green_host ≠ NULL ? green_host : brown_host
```

## 4.2 Consolidation Policies

As explained in 2, consolidation have shown great result in reducing energy consumption. We chose to explore consolidation in a distributed Fog environment and determine its usefulness.

The consolidation process implies to switch-off machines in order to save static consumption. Switching-off a machine implies a period during which it is unavailable; and the machine must be switched-on to be able to host jobs again, making it unavailable again for a specific duration. The unavailability of nodes is supposed to reduce the responsiveness of the architecture. Some jobs might have to wait for a machine to boot up until it can run on it. However, once the job is allocated, the boot duration is not an issue anymore: it does not impact the latency perceived by the user any more. In this sense, we introduce the notions of initial latency versus final latency:

- initial latency: time for the first connection from the initial node to the host including booting time, if we chose to place the job on a switched-off node. The allocated job will be able to receive incoming communication only after the initial latency duration.
- final latency: 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 nodes.

The choice between optimizing initial or final latency depends on the kind of submitted job. For example, a job with consequent duration and many interactions with the end-user should optimize final latency, for example an application hosting a video game. Conversely, an application submitting many jobs with short duration should optimize initial latency. In this work, we have chosen to consider jobs looking to optimize the final latency.

The consolidation routine in itself intervenes whenever a job end on a node, and can only shutdown a node if there is no job placed on it.

This section presents three consolidation policies explored in our work.

### 4.2.1 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.

### 4.2.2 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 producing it turned-on and reduce the impact on the responsiveness of the architecture. However, a node previously shut down because it did not produces 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 seems to be promising associated to our 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.

#### 4.2.3 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 5.1.2. The routine will check 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 reserved by a VM on the node.

The routine classifies each node as available or not available and search to maintain a 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 maintain the node availability ratio, then it will be shutdown. If it cannot be shutdown the routine will determine how many nodes should be switched-on to recover the node availability ratio, and will try to switch-on this number of nodes among the nodes checked. 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.

# 5 Validation

This section details the experimental setup used to conduct our simulations and then we analyze the impact of the allocation algorithms, consolidation policies and various parameters in our study.

## 5.1 Experimental Setup

First, we details the experimental setup used for our simulations. Then, we review the simulation platform, the Fog architecture, the traces used for the jobs and for the production of green power. Finally, we detail the power model of the nodes used in our simulations.

#### 5.1.1 SimGrid

To perform our simulations we chose to use the simulation platform SimGrid [8], a simulation toolkit designed for parallel and distributed large-scale system analysis. Its models used for CPUs, network, VMs and energy consumption have been theoretically and experimentally verified in several works [15] [28] [16]. In addition, SimGrid is highly scalable. For example, simulations of Chord [24] with two millions nodes on a single machine last 5.5 hours with SimGrid, while comparable simulators took 10 hours to simulate 300,000 nodes [23].

#### 5.1.2 Fog Architecture

As mentioned in Section 3, we chose to validate our exploration on a hierarchical Fog architecture. We selected a realistic topology similar to an actual national ISP [9]. This topology is represented in Figure 10. In this architecture there is five levels of nodes: core, backbone, metro, feeder, and user nodes. The central nodes are the cores nodes (Figure 12.a) and are located in two cities. There is two group of two close core nodes in each city. The close nodes are distant from 0.1 km to 0.5 km, the groups in the same city are distant from 5 km to 15 km and cores nodes in different cities are distant from 550 km to 650 km. The second level is composed of backbone nodes

| Network Distance (km) | Distance between cities (km) | Source city | Destination city | Latency (ms) | Standard Deviation (ms) |
|---|---|---|---|---|---|
| 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 | 1.161 |
| 50-500 | 300 | Munich | Frankfurt | 8.122 | 1.552 |
| 550-600 | 610 | Paris | Marseille | 18.607 | 0.07 |

Table 1: Latencies used for the links[1].

(Figure 12.b). Backbone nodes are grouped by two, distant from 0.1 km to 0.5 km. Each backbone node is connected to a core node, distant from 50 km to 500 km. The third level is composed of metro nodes (Figure 12.c). Each metro node is connected to two backbone nodes, distant from 0.1 km to 0.5 km. The fourth level is composed of feeder nodes (Figure 12.c). Each feeder node is connected to two metro nodes, distant from 1 km to 50 km. Finally, the last level is composed of the user nodes. Each user node is connected to a feeder node, distant from 0.1 km to 0.5 km. There is 5 to 10 user node per feeder node. The nodes connected to the same feeder constitute a district. To summarize, the architecture is composed of 8 cores nodes, 52 backbone nodes, 52 metro nodes, 260 feeder nodes and 1974 user nodes, for a total of 2346 nodes and 2692 links.

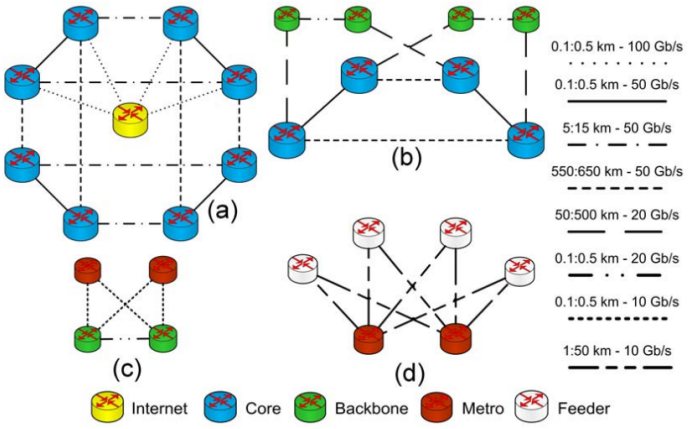


Figure 12: Link description [9]

The source of this topology do not mention latency between the nodes. We used real latency measured between cities as a reference for our latencies [1]. Latencies used are presented in Table 1. Those latencies are randomized following a Gaussian function. We used the same latency for 0.1-0.5 km and 5-15 km because using network equipment has an intrinsic time cost.

### 5.1.3 Traces

This subsection describes the traces used for the simulations. First, we detail traces used for the jobs, and then we detail traces used for the production of green energy.

[1]Latencies are taken from https://wondernetwork.com/.

| Function | Value |
|---|---|
| $P_{idle}$ | 1.488 |
| $P_{eth,idle}$ | -0.1176 |
| $P_{wlan,idle}$ | 0.899 |
| $P_{cpu(u)}$ | $0.6191u$ |

Table 2: Power model of the Raspberry Pi 3 B, in Watt [18]. $u$ is the CPU utilization in the range 0 to 1. The $P_{eth,idle}$ is negative because there is periodic CPU activity while no Ethernet is connected, but this activity stops when the network is active.

#### Job Traces

To our best knowledge, there is no existing Fog traces at the time this article is written. Hence, we chose to use Eucalyptus traces for our experiment. Eucalyptus traces are anonymized traces built from the log files of several systems running Eucalyptus private IaaS clouds. They where first published as part of a work by Wolski and Brevik [30].

The traces have been reworked to fit our architecture: jobs have been scaled so each job cannot 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 takes place on one host at a time). After being scaled, jobs have been duplicated in the traces so at the busiest moment of the simulation 73% of the available cores are used. This was done to have the architecture neither underutilized nor overutilized.

#### 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 [1], carried out at the University of Nantes. The production of the panels is logged every five minutes. We used production traces of 35 sessions recorded at various dates to represent the heterogeneity of the production of green energy depending on the location of the panels.

The traces have been scaled to correspond to the energy consumption of our nodes. Each scaled production trace cover, in average, half of the consumption of a node. As nodes of the same district are geographically close we chose to have the same trace for each node producing green energy in the same district. We distributed the 35 scaled traces over the 260 districts considering that close districts also have the same solar irradiance: districts 0 to 7 use trace 1, districts 8 to 17 use trace 2, and so on.

### 5.1.4 Power Model

We chose to simulate our Fog architecture considering each node as a Raspberry Pi(RPI) 3 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 decent computational capacity.

The power model used for the RPIs is described in Equation 1, and the values used are summarized in Table 2.

This power model is a simplification from a paper by Kaup and al. [18]. The job traces used in our simulation do not refer to data exchange between nodes and we supposed that data exchange time is far lower than data processing time. Thus, we do not adjust the consumption according to

| 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, 2 , 3 |
| Latency Range (ms) | 0, 5, 10, 20 |
| Node availability ratio | 0.4, 0.6, 0.8 |
| Per-node availability threshold | 0.5, 0.75 |

Table 3: Parameters explored in our simulations. Latency range do not apply for the "Latency First" algorithm and the parameters "Availability ratio" and "Availability threshold" only applies when the consolidation policy is set to "Consolidate Ratio".

the data flow from/to each RPI. This assumption depends on the kind of application deployed in the Fog.

A node consumes an amount of energy depending on its state and on the number of active and running cores. This energy consumed can be brown if the node do not produce green power, but it can be partly or fully green when the node produces green power.

We also considered the change in consumption while shutting down or booting. During those periods 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.

## 5.2 Results Analysis

We ran simulations to explore the interest of our distributed allocation algorithms and consolidation policies. Each simulation runs with fixed parameters and variable parameters. The fixed parameters are common to all simulations, those are the architecture in itself (nodes, links, latencies, etc) and the traces. The variable parameters are explored through simulations and are summarized in Table 3. Each simulation runs with a configuration file fixing the variable parameters for this specific simulation. Each configuration is simulated 10 times with a simulated time of three days and the results are averaged.

The results of the exploration of those parameters through simulation are detailed in this section.

### 5.2.1 Total Energy Consumption

The total energy consumption for each allocation algorithm is shown Figures 13, 14 and 15. Each plot present the total energy consumption of the infrastructure, depending on the consolidation policy. The policies are those detailed in the Section 4:

- NC : No Consolidation;
- CA : Consolidate All;
- CB : Consolidate Brown;
- CRi : Consolidation Ratio.

| ID | Node availability ratio | Per-node availability threshold |
|---|---|---|
| CR1 | 0.4 | 0.5 |
| CR2 | 0.4 | 0.75 |
| CR3 | 0.6 | 0.5 |
| CR4 | 0.6 | 0.75 |
| CR5 | 0.8 | 0.5 |
| CR6 | 0.8 | 0.75 |

Table 4: Parameters used in simulations with the consolidation policy "Consolidation Ratio". The ID column identify each combination.

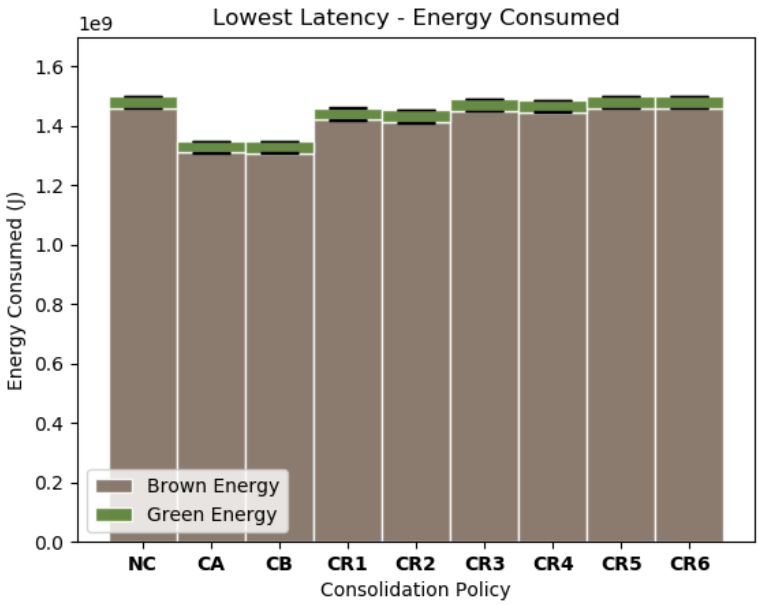


Figure 13: Total energy consumption with the allocation algorithm Lowest Latency. The number of green producers per district is fixed at 1.

The comportment of the consolidation policy "Consolidation Ratio" depends on two parameters, as explained in Section 4: node availability ratio and per-node availability threshold. The values used for those parameters are presented in Table 4. The ID column identify each combination.

The energy consumed is separated in two: brown energy and green energy. The number of green producers here is fixed at one and the latency range for the second and third allocation algorithm is fixed at 5 ms.

For each allocation algorithm the total energy consumption is similar. Without consolidation the infrastructure consumes significantly more energy. Among the consolidation policy, CR is the one consuming the most energy because it consolidates less nodes by keeping a specific ratio of nodes available.

The 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

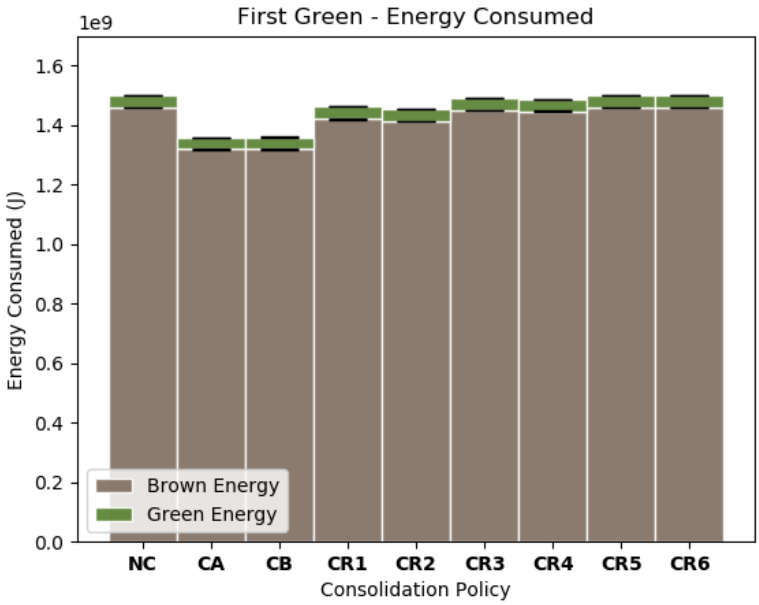


Figure 14: Total energy consumption with the allocation algorithm First Green. The number of green producers per district is fixed at 1 and the latency range is fixed at 5ms.

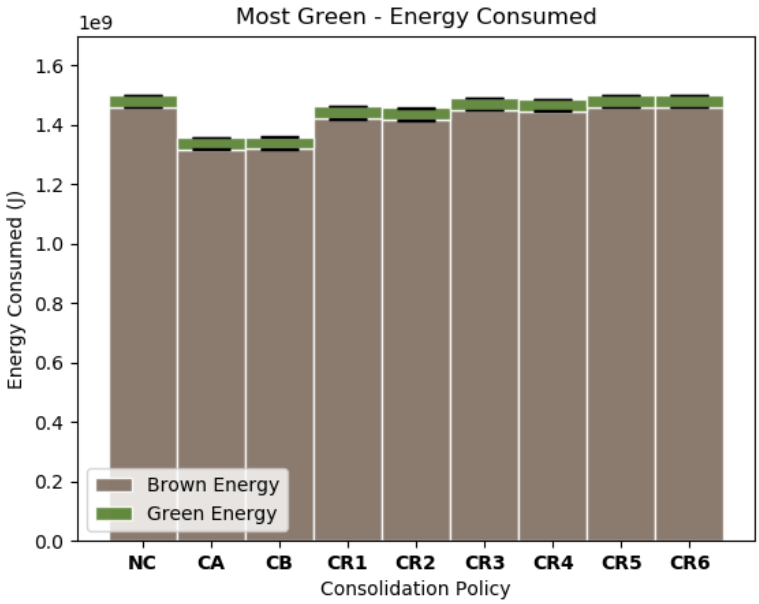


Figure 15: Total energy consumption with the allocation algorithm Most Green. The number of green producers per district is fixed at 1 and the latency range is fixed at 5ms.

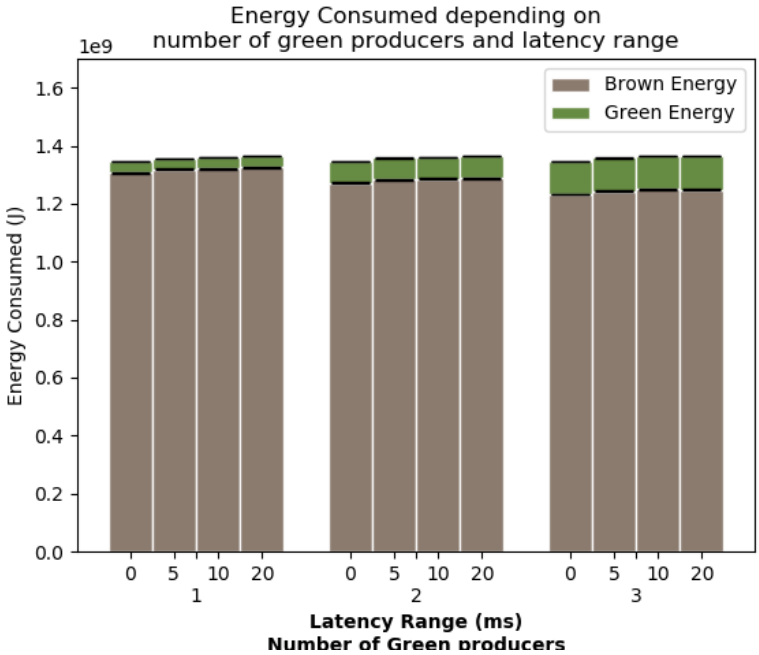


Figure 16: Total energy consumption with the allocation algorithm First Green and the consolidation policy CA.

produce only at 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 policy First Green, the result of the simulations exploring the impact of the latency range and the number of green producers per district is shown in Figure 16. For those experiments the consolidation policy was CA.

The number of green energy producers has an obvious impact on green energy consumption and it significantly improve its consumption. The latency range, allowing to search farther for a node producing green power, increases the total energy consumed without significantly improving the consumption of green energy.

### 5.2.2 Number of messages exchanged

The number of messages exchanged provides an idea of the cost for the network induced by the data exchange between the nodes. More precisely, the number of messages exchanged is the number of communication the nodes do when receiving a job and asking other nodes to find the best allocation, it does not concern messages exchanged while the job is running. Figure 17 presents the impact of the latency range on the number of messages exchanged, considering the allocation policy First Green. The results are worse with the algorithm Most Green, as it searches through every node in its latency range.

The number of messages exchanged increases exponentially with the latency range. This is coherent as the number of potential host grows exponentially with the latency range. However, the number of messages exchanged decreases as the number of green producers increase. This is due

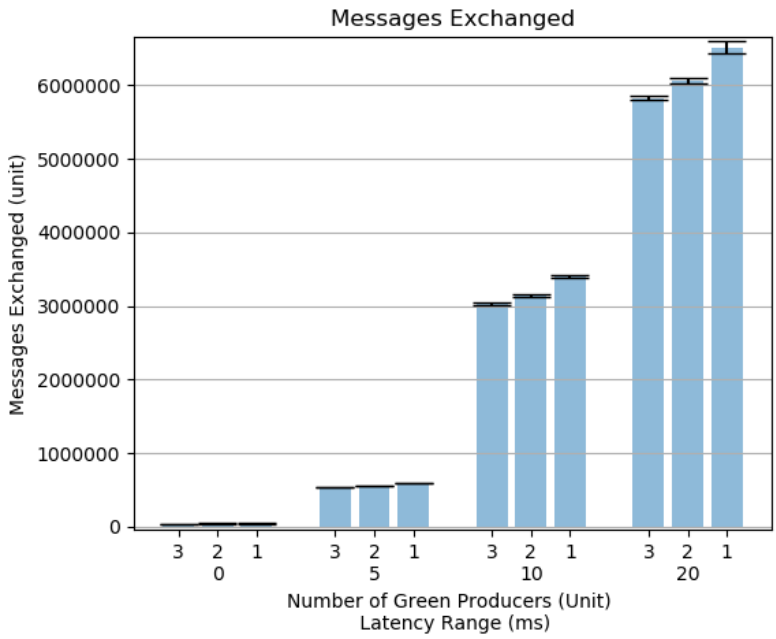


Figure 17: Number of messages exchanged to place jobs, depending on the number of green producers per district and latency range. The allocation policy is First Green and there is no consolidation.

to the allocation algorithm having an increased probability to find a node producing green energy.

### 5.3 Initial Latency

Latency is a key feature in the Fog. We have seen in Section 5.2.1 that consolidation policies reduce significantly the total energy consumption in Fog environment. However, consolidation implies to switch-off nodes, and then switch them back on, if needed. The initial latency considers the impact of switching-on a node when placing a job. Figure 18 presents initial latencies without consolidation with the allocation First Green. The trend is similar to the one with the messages exchanged: the initial latency grows exponentially with the latency range and decreases with the number of node producing green energy per district. However, even with the highest latency range of 20 ms, the average initial latency stays fairly low, below 2.4 ms. With the consolidation policies CA and CB the initial latency grows dramatically and reaches, in average, 70.7 seconds. Figure 19 presents initial latencies with the consolidation policy CR, the allocation algorithm First Green, with the number of green producers fixed at 1 and the latency range fixed at 20 ms. With this consolidation policy we observe the trade-off between latency and power consumption: the more machines are kept turned-on, the lower is the initial latency. However, the impact on consumption is small in comparison to the impact on latency.

### 5.4 Final Latency

Figure 20 presents results concerning the impact of consolidation on final latency. The allocation algorithm is set to First Green, the number of green producers per district is fixed to one and the latency range is fixed to 20 ms. This is the worst combination of parameters to reduce initial latency,

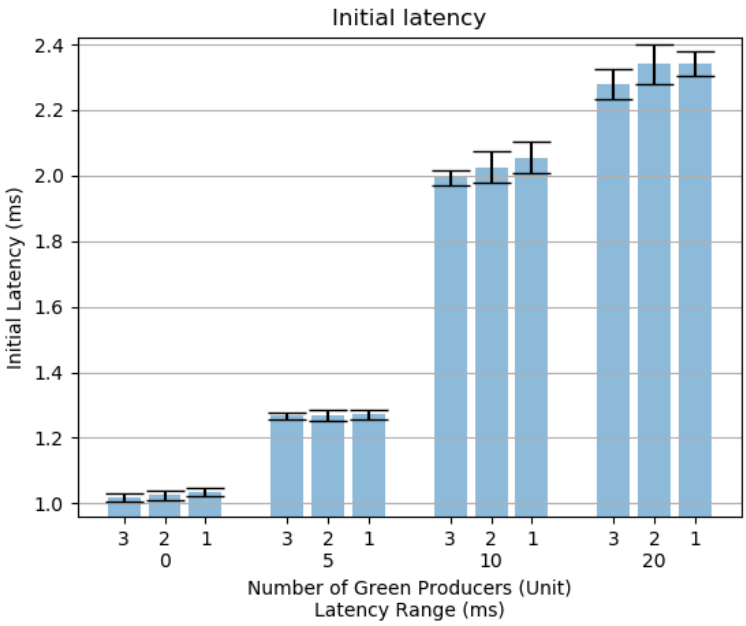


Figure 18: Initial latency without consolidation and allocation algorithm First Green.

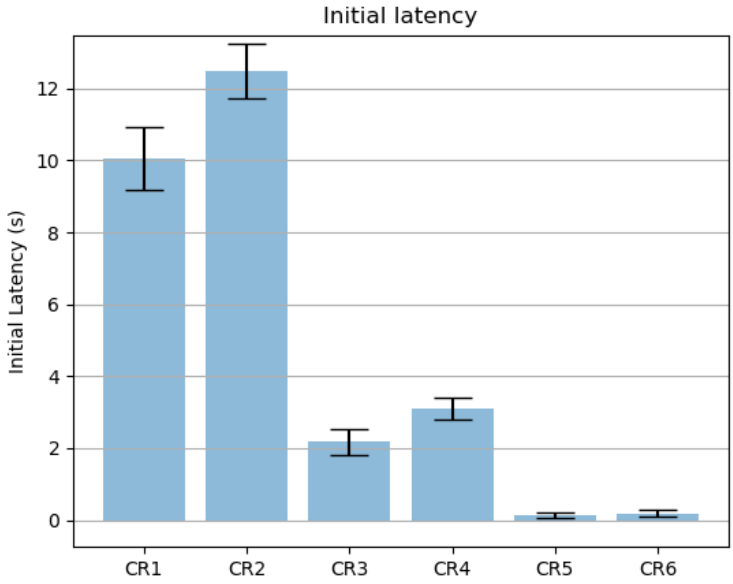


Figure 19: Initial latency with the consolidation policy Consolidation Ratio, allocation algorithm First Green, one node producing green energy per district and a latency range of 20 ms.

however, we observe that with this configuration the final latency is similar to the initial latency without consolidation and the same other parameters. Consequently, the studied algorithms do

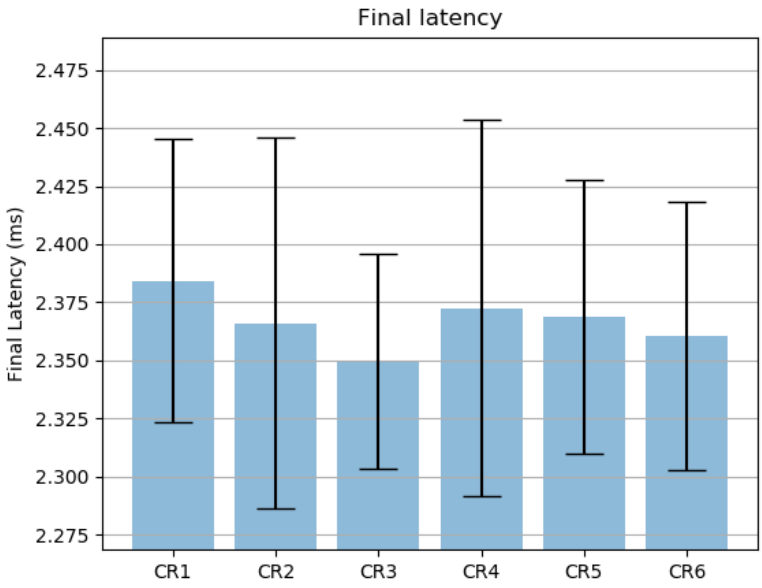


Figure 20: Final latency with the consolidation policy Consolidation Ratio, allocation algorithm First Green, one node producing green energy per district and a latency range of 20 ms.

not have a significant impact on final latency and can be employed indifferently for applications not sensitive to initial latency.

In conclusion, consolidation is crucial to reduce the energy consumption of distributed Fog infrastructures. But, it requires to deal with applications that are not impacted by the initial latency, that is to say, applications runing on long periods of time. The number of green producers can significantly reduce the number of exchanged messages to find the adequate node to allocate a job. 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.

# 6 Limits

This section tries to clarify some points around the possible limits of this work:

- ISP-like network: we chose to rely on an ISP-like network to model our Fog. This architecture allows to justify the variety of green power production as is it positioned over a wide area. However, the allocation algorithms and consolidation policies (except Consolidation Ratio) are fully convertible with other fully distributed architecture.
- Only CPU is considered: we do not consider storage because it is not currently envisioned to be an important bottleneck of the Fog. It is an architecture oriented around processing and aggregating data while the long term storage would be mostly done in the Cloud.
- Mobility: 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. Yet, the inclusion of mobile nodes would change the latency constraints and could lead to use mobility prediction policies or VM migration to stay within the required latency bounds.

- While the migration of VMs has not been investigated here, it is feasible to exploit this feature in this kind of Fog environment as dynamic migration of VMs from a node to another is implemented in the SimGrid toolkit. But, for container-based applications, it is not yet possible in practice. So, not considering this feature, our work can be applied to both virtualization technologies: virtual machines and containers.

- Only photovoltaic green energy: other sources of green energy, such as wind turbines, may not be suitable with our allocation algorithms. Solar power is intermittent but relatively stable, meaning that we can allocate a job on a node producing green power and assume that this node will continue to produce some green power. Conversely, wind power is extremely intermittent and adapting workload according to current wind power could be less effective. However, forcing allocation on nodes that may produce wind power, i.e. connected to wind turbine, may be useful to rely on this energy when it is available. Allocation policies dealing with such extremely intermittent power sources should employ robust prediction algorithms.

# 7 Conclusion

The Cloud is already a powerful architecture answering several needs of the current IT context, such as on-demand resource availability. However, the future of IT and envisioned new applications may be limited by the current architecture of the Cloud. The Fog positions itself as a necessary extension to the Cloud to fill in these limitations. The Fog should notably help reducing network congestion by keeping data locally and processing them closer to the source and destination. In addition, the main added value of the Fog is the reduced latency due to this closer processing, greatly reducing communication time between the source, processing architecture, and destination of data or service.

In the early states of the Cloud, its power consumption was neglected, notably due to the high demand for its services and fairly low cost of energy. Nevertheless, power consumption quickly became at the heart of the design of Cloud architectures, as it represents now the main part of its running cost and the demand for greener infrastructures is globally growing. Many work have been presented to reduce its energy consumption, or improving its usage of green energy. The Fog is envisioned to be largely deployed over the world in many cities, and while its energy consumption has been evoked in several studies, the usage of green energy has not been considered to the best of our knowledge.

In this work, we tried to model a coherent and fully distributed Fog architecture, and to consider the usage of green energy in this context. In addition, we explored the interest of using various consolidation policies in this context.

The usage of consolidation in this context presents interesting results. Independently of the allocation algorithms, the proposed consolidation policies allow to greatly reduce the total energy consumption of the Fog nodes. In the best case, consolidation policy "Consolidate All" allows to reduce the total energy consumption by up to 10.2%. But, due to the nature of consolidation, the

responsiveness of the infrastructure is considerably reduced. In the aforementioned case, consolidation increases initial latency from 0.0010 seconds to 71.6 seconds. This may seem to be an important issue considering that one of the primary goal of the Fog is to be more responsive than the Cloud and reduce the end-to-end latency. However, the consolidation only impact 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 job will have to wait more, in average, to starts on a node, but the real latency will be almost the same as in a distributed Fog without consolidation. In the aforementioned case, the final average latency when running the job is 0.0010 seconds, with or without consolidation.

On the other hand, the proposed allocation algorithms aiming to optimize the green energy consumption have not been conclusive as they do not significantly modify the total energy consumption of the infrastructure. In fact, the experimental setup may not be adequate to take advantage of the proposed allocation algorithms. The green energy is produced by specific nodes equipped with photovoltaic panels, and can only be consumed by themselves. We tried to allocate jobs on these specific nodes in order to benefit at most of this energy. But, even if the green energy production is important, a node in itself do not consume much energy, even when fully loaded. In addition, the nodes in our Fog context are too small, with low processing capabilities, reducing the number of jobs that can be hosted on a single node. Additionally, searching to place jobs on node producing green energy increase the distance between source and destination, and increase end-to-end latency. To fully explore the performances of those allocation algorithms we need to consider more green energy producers and nodes with greater processing capacity, which may be further away from the currently envisioned Fog hardware implementations.

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 this context. However, consolidation may be considered to reduce its consumption. It provides consequent reduction in energy consumption but may not be adapted to every kind of application.

# 8 Future Work

Directions for future work include the following points:

- Semi-distributed Fog: feeder nodes are linked to all nodes of a district, in fact they define a district. It could be interesting to use them more actively to store information about the nodes in their district, notably which ones are potential green energy producers. This model could reduce significantly the number of exchanged messages.

- Job migration: currently, jobs are allocated on host at submission time and will pursue their processing on the same node. The benefits of migrating job between nodes to exploit more green power or to improve the consolidation could be investigated [7].

- Dealing with unused green energy: green energy produced but not consumed is currently considered wasted. Other work considers the benefits of reselling it to the grid or sharing it through a virtual pool [6].

- Various trace for jobs and PV production: running simulations with more various job and photovoltaic power production would provide more insight of cases where this kind of management could be useful.
- Heterogeneous nodes: all nodes in our Fog are currently identical, which may not reflect reality and heterogeneity could be interesting to explore from an energy-efficient perspective. In addition, this point could be particularly useful to explore the interest of producing green energy depending of the size and processing capacity of the node producing this energy, which seems to be currently a limitation to exploit green energy in our model.
- Network consumption: considering the energy cost of the wired network could provide more insight about the overall consumption of the Fog and about the impact of management messages.
- Prevision of photovoltaic production: several studies use prediction to determine how and when to allocate jobs in order to better profit from the green energy production [14] [20] [7]. These predictions could be applied in a Fog context to optimize the consumption of green energy.

## Acknowledgment

I would like to express my gratitude to every person in the MYRIADS team for their welcome and their assistance all along this internship.

## References

[1] Photovolta project. `http://photovolta2.univ-nantes.fr/`, 2011.

[2] 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.

[3] OpenFog Consortium. Process Manufacturing in Beverage Industry. `https://www.openfogconsortium.org/wp-content/uploads/Beverage-Industry-Short.pdf`, 2018.

[4] Sherif Akoush, Ripduman Sohan, Andrew Rice, and Andy Hopper. Evaluating the viability of remote renewable energy in datacentre computing. *Technical reports published by the University of Cambridge*, March 2016.

[5] 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.

[6] Benjamin Camus, Anne Blavette, Fanny Dufossé, and Anne-Cécile Orgerie. Self-Consumption Optimization of Renewable Energy Production in Distributed Clouds. In *IEEE International Conference on Cluster Computing*, pages 1–11, Belfast, United Kingdom, September 2018.

[7] 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 *International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, pages 1–8, Lyon, France, September 2018.

[8] 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.

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

[10] 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)*, Sydney, Australia, December 2015.

[11] 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, Dec 2016.

[12] U. Drolia, K. Guo, J. Tan, R. Gandhi, and P. Narasimhan. Cachier: Edge-caching for recognition applications. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 276–286, June 2017.

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

[14] 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 *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 20:1–20:11, New York, NY, USA, 2011. ACM.

[15] F. C. Heinrich, T. Cornebize, A. Degomme, A. Legrand, A. Carpen-Amarie, S. Hunold, A. Orgerie, and M. Quinson. Predicting the energy-consumption of mpi applications at scale using only a single node. In *2017 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 92–102, Sep. 2017.

[16] 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, Jan 2018.

[17] 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, May 2016.

[18] Fabian Kaup, Stefan Hacker, Eike Mentzendorff, Christian Meurisch, and David Hausheer. The progress of the energy-efficiency of single-board computers. *Technical reports No. NetSys-TR-2018-01*, January 2018.

[19] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan. GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1–5, Dec 2010.

[20] Yunbo Li, Anne-Cécile Orgerie, and Jean-Marc Menaud. Opportunistic Scheduling in Clouds Partially Powered by Green Energy. In *IEEE International Conference on Green Computing and Communications (GreenCom)*, Sydney, Australia, December 2015.

[21] Jiacheng Ni and Xuelian Bai. A review of air conditioning energy performance in data centers. *Renewable and Sustainable Energy Reviews*, 67, 01 2017.

[22] 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 Computing Surveys*, 46(4):47:1–47:31, March 2014.

[23] M. Quinson, C. Rosa, and C. Thiry. Parallel simulation of peer-to-peer systems. In *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 668–675, May 2012.

[24] Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, Frans M Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE Transactions on Networking*, 11, 02 2003.

[25] 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.

[26] 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 *Digest of Papers. Fourth International Symposium on Wearable Computers*, pages 139–146, Oct 2000.

[27] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massouli, Christophe Diot, and Pablo Rodriguez. Greening the Internet with Nano Data Centers. *ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 37–48, 01 2009.

[28] 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):23:1–23:26, December 2013.

[29] 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, September 2012.

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

[31] 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, Nov 2018.