Modeling, evaluating, and orchestrating heterogeneous environmental leverages for large-scale data center management

Vladimir Ostapenco*, Laurent Lefevre*, Anne-Cécile Orgerie†, Benjamin Fichel‡
* *EnsL, UCBL, CNRS, Inria, LIP, University of Lyon*, *Lyon, France* - {laurent.lefevre,vladimir.ostapenco}@ens-lyon.fr
† *Inria, CNRS, IRISA, University of Rennes*, *Rennes, France* - anne-cecile.orgerie@irisa.fr
‡ *OVHcloud*, *France* - benjamin.fichel@ovhcloud.com

***Abstract*—Data centers are very energy-intensive facilities that can generate various environmental impacts. Numerous energy, power, and environmental leverages exist and can help cloud providers and data center managers to reduce some of these impacts. But dealing, with such heterogeneous leverages can be a challenging task that requires some support from a dedicated framework. This article presents a new approach for modeling, evaluating and orchestrating a large set of technological and logistical leverages. Our framework is based on leverages modeling and Gantt chart leverages mapping. First experimental results based on selected scenarios show the pertinence of the proposed approach in terms of management facilities and potential impacts reduction.**

***Index Terms*—Energy efficiency, environmental impacts, data center management**

I. Introduction

The use of information and communication technologies (ICT) has increased considerably. Data centers are at the center of this progress and are very energy-intensive facilities, estimated to represent around 1% of global electricity consumption Masanet et al. (2020). Even if some studies suggest some flat energy consumption of worldwide data centers over the last decade Masanet et al. (2020), some exhaustive review exhibit more contrasted results Mytton and Ashtine (2022). In order to make data centers more energy efficient, a wide variety of approaches have been proposed like Orgerie et al. (2014) and Ayanoglu (2019). These approaches are called energy and power leverages, and their application can help to reduce the energy usage of large-scale infrastructures. These leverages can be very heterogeneous and can involve hardware, software layers or more logistical constraints in a data center. But such energy leverages only cover a part of the consumption and impacts of data centers. Multi-criteria impacts coming from the full life cycle of ICT must be considered beyond energy. These impacts can cover greenhouse gas emissions, pollution, and impact on air, water, and biodiversity. In this work, we refer to environmental leverages as those that can cover both consumption and multi-criteria impacts.

An environmental leverage can be technological or logistical and can impact various components of a data center in different ways. A technological leverage is a leverage that is directly applicable to a component of the data center infrastructure and has an impact on its power/energy efficiency or environmental footprint by modifying its state. A logistical leverage is a leverage that involves logistic (non-technical) aspects of cloud provider management processes.

The high heterogeneity of these leverages makes managing them challenging, particularly in a large-scale environment.

The first challenge consists in modeling and evaluating the impact of heterogeneous leverages. We explore in this work the design of a methodology in order to model them and assess their impacts.

A second challenge lies in combining leverages and determining the impact of a given combination. Leverages can be combined with each other to amplify their impact. However, not all combinations are beneficial, some leverages are not compatible with each other or the use of two combined leverages can reduce their impact. So the impact of the combination of these leverages (side effects, compatibility) must be studied.

Another challenge is the requirement to have a means of managing these leverages in a multi-node environment while respecting the operational constraints, such as the availability of the leverages or the customer constraints.

The final challenge is the ability to assess the impacts of a specific leverage management strategy and to track environmental metrics throughout the management process.

In this work, we propose to enrich the existing management systems, with a novel heterogeneous leverage modeling and a leverage management framework that includes at the same time technological and logistical leverages and aims to improve the overall energy and environmental performance of the entire infrastructure of a cloud service provider. We improve existing modeling approaches by introducing logistical leverages modeling and intelligent action-based leverages that have complex decision logic involving multiple metrics and data center components.

We also propose the concept of an environmental Gantt chart which tracks the entire life cycle of data center infrastructures and permits the positioning of environmental leverages. The Gantt chart can be used for instant infrastructure management, action planning, and even the instrumented replay of the past. It also allows the tracking of metrics to quantify impacts and gains in energy, CO2 and greenhouse gases to further reduce the environmental impact of the data center infrastructure.

Two selected scenarios "Temporary power capping at data center scale" and "Minimizing the global carbon footprint of a data center" are explored and show the benefits of our approach both in terms of flexible management and environmental impacts reduction.

This article is organized as follows. The section Related work covers selected related work about the energy leverages. The section Cloud provider infrastructure, external context, and actors provides a representation of the cloud provider infrastructure in our system, as well as the definition of the external context and actors. The section Heterogeneous environmental leverage introduces the definition of environmental leverages and overviews the main available leverages, while the section Leverage formalization describes leverage formalization. The section Environmental Gantt chart explores the design of the environmental Gantt chart, and the section Leverage management framework overviews the proposed framework. The section Experimental validation covers the experimental validation of our approach with two analyzed scenarios : "Temporary power capping at data center scale" and "Minimizing the global carbon footprint". The section Conclusion and future works concludes this article and presents some future works.

II. Related work

In order to reduce the environmental impact of data center infrastructures, a wide variety of approaches (leverages) exist.

Some of the leverages such as shutdown, DVFS (Dynamic Voltage and Frequency Scaling) and RAPL (Running Average Power Limit) with their impacts are well studied in the literature. Raïs et al. (2018b) and Benoit et al. (2018) evaluate the potential gain of shutdown leverage by taking into account the state transition costs in terms of time and energy. Suleiman et al. (2005) are studying the DVFS (Dynamic Voltage and Frequency Scaling) technique to improve processor energy efficiency for general-purpose microprocessors. Rountree et al. (2012) explore the potential of Intel RAPL technology to replace the DVFS technique. Haidar et al. (2019) investigate the use of RAPL technology to control compute node power consumption, as well as the impact of various levels of RAPL power caps on application performance.

Despite the fact that numerous studies have been conducted to evaluate individual leverages, few studies have considered combining them. Rais et al. (2018) propose a generic framework for their combination through the definition of a table of leverages impacts. Raïs et al. (2018a) propose algorithms that facilitate the reading of the table of leverages and help in extracting knowledge from them. Raïs (2018) in his thesis work delves deeper into combining, benchmarking and extracting knowledge from combined leverages.

Therefore, the majority of works focus on the investigation of a single technological power/energy oriented leverage or their combinations. In this work, we extend the current state of the art by proposing a novel heterogeneous leverage modeling and management approach that combines technological and logistical leverages and that is applicable to large-scale multi-region data centers.

III. Cloud provider infrastructure, external context, and actors

In this section, we highlight how the infrastructure of the cloud provider is represented in our system, define the notion of context external to the cloud provider, and introduce the actors of our system.

A. Logical component

The essential infrastructure of a cloud provider can be represented by logical components. A logical component is a component to which an action affecting the environmental impact can be applied. It could be completely virtual, such as a cloud workload, or it could represent a physical component of the data center, such as a compute node.

In Figure 1, we propose a hierarchical set of logical components that can be represented in our system. This set includes high-level logical components as a cloud provider to which we will apply global goals or constraints, as well as a compute node or workload logical components that can be directly manipulated.

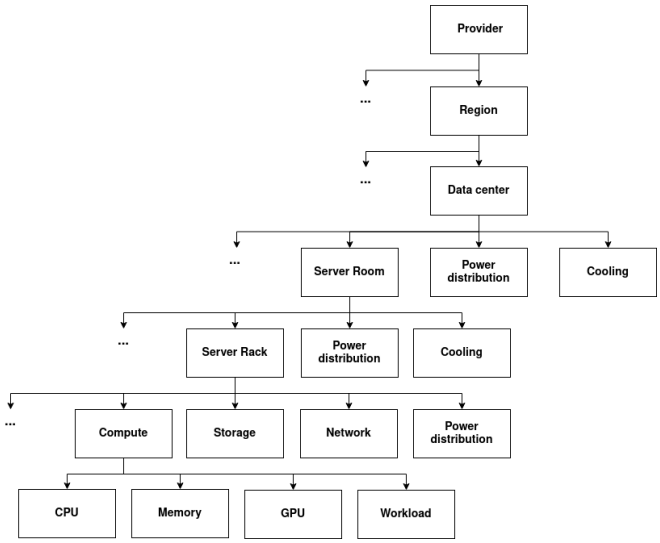

Fig. 1. Logical component hierarchy.

1) Logical component state: The logical component state is the internal state of each logical component (e.g., data center, server, CPU, and memory). The logical component state is the current observable state maintained by the cloud provider and used in the decision process by the leverage management framework. It is retrieved from cloud providers' existing supervision systems, power meters, or even smart power distribution units.

Even though it is entirely dependent on the existing monitoring solution and its characteristics (frequency, accuracy, etc.), our framework keeps a copy of this state. This enables tracking metrics across an entire logical component's life cycle while

remaining independent of cloud providers' metric retention policies.

Logical component state examples:

- Power consumption of an entire data center.
- Power consumption of the particular compute node.
- CPU frequency of the particular compute node.

In Figure 2, we demonstrate an example of a data center and its sub-components' instantaneous power consumption.

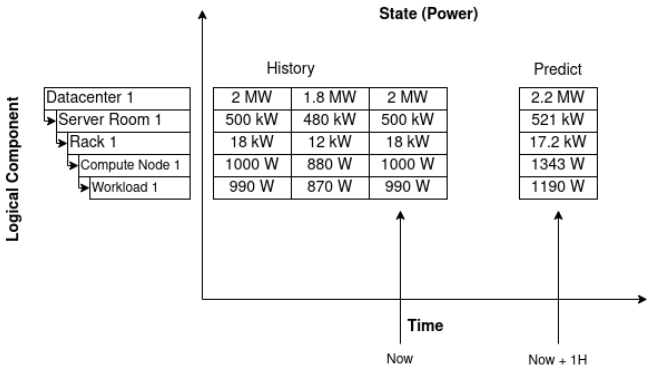

Fig. 2. Power consumption of data center 1 and its sub-components.

Since our framework saves the complete history of the logical component's states, it can be used in the future to create prediction models and forecast state values. This offers a wide range of opportunities for the proactive management of cloud provider infrastructures.

2) Logical component inventory: The logical component inventory is a hierarchical representation of the cloud provider in the form of logical components used by the leverage management framework. The logical component inventory follows the hierarchy of logical components introduced earlier and must be built prior to using the framework. It can be provided by the cloud provider's existing inventory management system.

B. External context

The external context represents the context that is outside of the cloud provider's control but can have an impact on its operation, either directly or indirectly. It cannot be influenced directly by the cloud provider, but it can be useful in our management framework's decision-making process. In this work, we divide the external context into four categories.

The first category is the client's external context, which represents the set of possible client actions that cannot be influenced by the cloud provider but requires him to take some action. A client can initiate the reservation of supplementary resources, their migration, and their release at any moment.

The second category represents the societal external context, such as energy prices, current energy mix, or even the Ecowatt[1] index, which in real-time reflects the load of the electrical system in France.

[1]Ecowatt qualifies in real time the level of electricity consumption in France. (https://www.monecowatt.fr/)

The third category is the external environmental context, such as meteorological conditions, which can affect data center cooling and necessitate a set of actions in order to meet global thermal data center constraints.

The fourth category is IT (Information Technology) external context, which represents any IT event that is not under the cloud provider's control but requires action from it. It could be unexpected urgent software updates or recently discovered security breaches.

The leverage management framework maintains a local copy of the external context to be independent of retention periods and temporary outages of the context providers. The external context can be used to build prediction models and model external context for future action planning.

C. Actors

An actor is an entity that initiates actions on data center infrastructure. In our system, an action can be initiated by the following actors:

- **Leverage management framework** performs a set of actions by placing a set of leverages on the environmental Gantt chart to meet predefined constraints, to reduce impacts or to respond to a request from a cloud provider operator. The actions performed by the leverage management framework are managed by a set of algorithms with varying degrees of complexity. When the framework responds to a request from a cloud provider operator, the algorithm also validates the legitimacy of the operator's request.
- **Cloud provider operator** is a person who manually performs actions on logical components. The cloud provider operator includes a wide range of cloud provider employees, from a technician changing a processor on a compute node to a manager deciding to build a new data center.

IV. Heterogeneous environmental leverage

In this section, we introduce the concept of an environmental leverage and highlight both technological and logistical leverages.

A. Environmental leverage

Raïs et al. (2018a) define leverage as a technique that improves the data center's power or energy efficiency at various levels.

In this work, we extend this definition to an environmental leverage that not only improves power or energy efficiency but also the environmental impact and can be applied in the context of a multi-data center and a multi-region cloud provider. In addition, rather than defining only technological leverages, we introduce logistical leverages associated with cloud provider management processes such as compute node deployment and decommissioning.

B. Technological leverages

A technological leverage is a leverage that is directly applicable to a logical component and has an impact on its power/energy efficiency or environmental footprint by modifying its state. This section provides an inventory of existing technological leverages that we have identified and that are applicable in the context of a cloud data center.

a) Shutdown policies: Since the static part of the energy consumed (consumption even when no workload is executed) accounts for a substantial portion of the total energy consumed by a compute node, shutting down unused nodes results in significant energy savings.

However, shutting down and powering on compute nodes incurs significant time, energy, and power costs. Raïs et al. (2018b), Benoit et al. (2017) evaluate these costs and study the impact of different shutdown leverage states (idle and off) on power usage. Furthermore, they demonstrate that the shutdown leverage affects not only power consumption but also temperature and thus cooling systems. Shutdown policies should therefore be used with great caution.

b) Sleep states: The sleep states is a feature of various data-center components that involves putting resources into sleep mode or turning them on/off based on platform usage. The sleep states is a useful leverage for dealing with idle periods and can provide energy savings by lowering node static consumption. Transitioning between different states of this leverage incurs non-negligible time and energy costs. As a result, this leverage must be used carefully to avoid becoming counterproductive.

c) DVFS (Dynamic Voltage and Frequency Scaling): The DVFS (Dynamic Voltage and Frequency Scaling) technology, which is available on the majority of compute nodes, allows for the modulation of voltage and/or working frequency to improve energy efficiency. This leverage only acts on the dynamic part of power consumption (increase in consumption due to the execution of the workload) and reduces workload execution performance. This lengthens the execution time and may result in missed deadlines. Krzywda et al. (2018) show that the impact of DVFS on power reduction is dependent on IT resource utilization and can be limited. Moreover, the voltage transition requires time and implies energy consumption Kim et al. (2008).

d) RAPL (Running Average Power Limit): RAPL interface was introduced by Intel in 2011 David et al. (2010) in the Intel Sandy Bridge architecture and provides power-limiting capabilities on computing node processor packages and memory by enabling the specification of average power consumption over a time period.

Haidar et al. (2019) show that in some cases, using RAPL power limiting can reduce the overall energy consumption by 30% while maintaining execution performance. Zhang and Hoffman (2015) demonstrate that the RAPL interface is quite accurate and stable for the majority of long-running applications. They also show that RAPL has short state transition times and is quite efficient, with efficiency dependent on workload and power limit. Rountree et al. (2012) investigate the potential of RAPL as a DVFS replacement.

e) Workload migration: Workload migration is a leverage that reduces the use of compute nodes and thus their energy consumption by migrating the workload to another node. This leverage is an excellent candidate for combining with other leverages such as shutdown or sleep states to maximize its impact. The workload migration leverage has significant time and energy costs that must be considered Hamdi and Chainbi (2019).

f) Simultaneous multi-threading (SMT): Simultaneous multi-threading (SMT) is a technique available in modern CPUs to improve overall efficiency through hardware multi-threading at the core level. It allows multiple threads to run concurrently in a single core of a given CPU at the same time, allowing independent threads to keep CPU resources busy at all times. Yu et al. (2017) show that using SMT can be beneficial for some workloads, with a significant increase in energy efficiency, with a 40% decrease in average power consumption. However, because the CPU may spend all of its time context switching between concurrently running threads or processes, this leverage could severely degrade an application's performance and energy efficiency.

g) Scheduling policies: Scheduling policies define how jobs are assigned to compute nodes. The choice of scheduling policy can have a significant impact on the energy efficiency of execution. Scheduling policies can even be used to control power and energy consumption. This leverage has a significant impact on the utilization of data and computing components, and thus on energy consumption. The choice of scheduling policy can be influenced by location-specific metrics such as the carbon intensity of electricity or the energy/power budget.

h) Impact-aware client request load balancing: In today's cloud service architectures, a service can be replicated in multiple geographical locations to serve clients based on various parameters such as client location or current infrastructure load. The idea behind the impact-aware client request load balancing leverage is to route client requests based on metrics like electricity carbon intensity or data center power budget.

i) Compression: Compression leverage aims to reduce the time and energy costs of data transfer by utilizing various compression techniques. It allows you to improve the energy efficiency of a data transfer by using lossy or lossless compression. Rais et al. (2019) demonstrate the utility of the lossless compression process in a variety of contexts and propose a model that determines the relevancy of using the lossless compression leverage for energy efficiency.

j) Application execution environment: An application can run in a variety of environments, including physical machines, virtual machines (VMs), containers, or even serverless platforms. Each execution environment operates differently, affecting the overall application footprint and resulting in varying levels of application execution energy efficiency. The primary goal of application execution environment leverage is to reduce the application footprint through application migration

to a less impacting execution environment. It entails migrating an application from physical machines to virtual machines (VMs), and then from VMs to containers. Virtual machines make better use of physical machine resources. Containers have a much smaller footprint than virtual machines and are more portable. Using this leverage enables the use of more aggressive workload consolidation and workload migration strategies in the future without compromising service quality.

C. Logistical leverages

A logistical leverage is a leverage that involves logistic (non-technical) aspects of cloud provider management processes. This section focuses on logistical leverages applicable in the context of a cloud provider data center that we have identified.

a) Equipment deployment: The equipment deployment leverage involves the installation of a new logical component (a compute node, a server rack, a server room, a data center). This leverage has a non-negligible cost in terms of time and can have a carbon footprint. The duration of the deployment process varies depending on the cloud provider. The carbon impact is implied by the need to take into account the carbon footprint of the deployed equipment's manufacturing and transportation.

b) Equipment decommissioning: The equipment decommissioning leverage entails the decommissioning of a logical component when it reaches the end of its life. Decommissioning involves turning off, disconnecting, and recycling the equipment. This leverage has also a cost in terms of time and a non-negligible carbon footprint. The carbon footprint is caused by the cost of recycling equipment.

1) Component extension or renewal: The component extension or renewal leverage entails adding or renewing a specific internal component of a low-level logical component (compute node, storage node, network node) so that it consumes less energy while performing its function. Furthermore, in some cases, this leverage can extend the lifespan of a logical component.

a) Functional reassignment: The function reassignment leverage involves changing the function of a logical component in order to improve its energy efficiency or reduce the overall environmental impact of the infrastructure. A compute node with a GPU may be more energy efficient when executing certain types of workloads. So, if we discover that executing the workload on another node is more energy efficient, the workload can be migrated to that node.

b) Geographical reassignment: The geographical reassignment leverage involves physically moving infrastructure from one location to another in order to reduce its environmental impact. The decision could be based, for example, on energy supply sources. If the electricity supplied in one region is greener than in another region, we can move equipment and improve the overall environmental impact of both regions. This leverage has significant time and carbon costs, as moving physical infrastructure can take a long time and transporting equipment has a notable carbon footprint.

V. Leverage formalization

Raïs (2018) in his work, formally defines a leverage with the following definition:

Definition 1. *A leverage L is triplet $L = (S, s_c, f_s)$, where $S = \{S_0, S_1 ... S_n\}$ is the set of available valid states of L, s_c is the current state of L and f_s is a function to update the current state to a new state $s_c^{'} \in S$.*

This definition has been proposed for energy and power technological leverages with a finite number of states and a basic state transition function. In this work, we apply and extend this definition to more heterogeneous environmental leverages, including logistical leverages, which impact not only energy or power metrics but also cover other environmental impacts.

The leverage formalization introduced in this section does not cover leverage's impacts and costs, which are discussed later.

The state of a leverage can be changed by applying the state transition function f_s. The state transition function affects the leverage state by performing a set of actions on logical components. Based on the actions performed by the state transition function, we define two types of leverages: atomic action-based leverages and intelligent action-based leverages.

A. Atomic action-based leverage formalization

Atomic action-based leverage has a context-independent and deterministic state transition function that performs a directly applicable action on a single logical component.

For instance, we can consider the RAPL technology as a technological atomic action-based leverage. Assuming that the range of possible power capping configurations ranges from 10 Watts to 120 Watts with a step of 10 Watts and the current power capping configuration is 120 Watts, the RAPL leverage can be defined as follows: $L_{RAPL} = (\{10Watts, 20Watts ... 120Watts\}, 120Watts, f_{RAPL})$. The f_{RAPL} is a state transition function changing RAPL power capping configuration. The f_{RAPL} first verifies the power capping parameter supplied, then executes a command to modify the power capping configuration of the logical component, and finally updates the leverage's current state.

Compute node deployment can be perceived as a logistical atomic action-based leverage. Assuming that the compute node deployment can be initiated, in progress, or finished and the leverage has just been initiated, this leverage can be defined as follows: $L_{COMPUTE\_DEPLOY} = (\{Initiated, InProgress, Finished\}, Initiated, f_{DEPLOY})$. The f_{DEPLOY} is a state transition function that initiates a compute node deployment, changes the deployment leverage state, and tracks the progress of the deployment.

B. Intelligent action-based leverage formalization

Intelligent action-based leverage has a context-dependent state transition function that performs a set of combined actions on multiple logical components. In order to take global actions or meet global constraints, the state transition

function of intelligent action-based leverage can create other technological and logistical leverages.

Power capping at the scale of an entire data center can be formalized as a technological intelligent action-based leverage. Assuming that the range of possible power capping constraints ranges from 0% to 100% with a step of 10% and the current constraint set to 0%, this intelligent leverage can be defined as follows: $L_{POWERCAP} = (\{0\%...100\%\}, 0\%, f_{POWERCAP})$. The $f_{POWERCAP}$ is a state transition function that creates a set of technological atomic action-based power capping leverages on a set of logical components in order to respect the global data center power constraint.

Compute node end-of-life management leverage can be also defined as an intelligent action-based leverage that is at once technological and logistical. Assuming that the end-of-life management can be initiated, in progress or finished, and that the leverage has just been initiated, this leverage can be defined as follows: $L_{EOF} = (\{Initiated, InProgress, Finished\}, Initiated, f_{EOF})$. The f_{EOF} is a state transition function that manages the end-of-life of the compute node by creating both logistical atomic action-based leverages (compute node order, compute node decommissioning) and technological atomic action-based leverages (workload migration and shutdown).

C. Leverage impacts and costs evaluation

A leverage has various impacts on the logical component and can have costs in terms of energy and time. Some leverages may even have a cost in terms of carbon footprint. These impacts and costs must be taken into account by the state transition function of intelligent action-based leverages in order to apply them effectively.

It is therefore necessary to evaluate the impacts and costs of each leverage with all the combinations of available parameters. This evaluation results in a table of leverages that contains for each combination of leverage parameters its impacts on the logical component and its costs.

TABLE I
TABLE OF LEVERAGES OF DVFS LEVERAGE WHILE EXECUTING WORKLOAD 1.

Frequency	Time on	Time off	Power draw
1.2 Ghz	1ms	2ms	17W
1.4 Ghz	1ms	2ms	18W
1.8 Ghz	1ms	2ms	22W
2.0 Ghz	1ms	2ms	24W
2.6 Ghz	1ms	2ms	26W

1) Table of leverages of DVFS leverage: Table I shows an example of the table of leverages of DVFS leverage while executing a specific workload. The table of leverages of DVFS leverage displays the impact of each frequency parameter on compute node power draw and the costs of state transition. This table does not reflect actual values, is workload dependent, and is provided solely for illustrative purposes.

TABLE II
TABLE OF LEVERAGES OF COMPUTE NODE DEPLOYMENT LEVERAGE.

Model	Time	Carbon footprint
Dell PowerEdge R640	3 weeks	1306.37 kgCO_2eq
Dell PowerEdge R630	3 weeks	1285.02 kgCO_2eq
Dell PowerEdge R440	3 weeks	1177.6 kgCO_2eq

2) Table of leverages of compute node deployment leverage: Table II illustrates a table of leverages of compute node deployment leverage. It shows the costs and impacts in terms of time and carbon footprint of deploying various models of compute nodes. The deployment carbon footprint includes the life cycle stages of transportation and manufacturing. The deployment time for any model of compute node is estimated to be 3 weeks.

Rais et al. (2018) propose a formalization of the table of leverages and a methodology for constructing such tables. The proposed methodology relies on metrics and benchmarks to characterize the impact of each leverage and leverage combination on a given compute node. This methodology is validated experimentally for a combination of three distinct leverages while executing CPU-intensive benchmarks.

In this work, we use a similar methodology for constructing tables of technological leverages. For logistical leverages, tables of leverages are built manually or semi-manually, as they comprise data provided by third parties, such as the carbon footprint of a compute node's manufacturing or the duration of compute node deployment process.

The table of leverages containing impacts and costs is a crucial element of the proposed modeling, which has a major impact on the outcome of the state transition function of intelligent action-based leverages.

VI. LEVERAGE IMPLEMENTATION

This section describes how each type of leverage is implemented in the proposed leverage management framework. We begin by presenting the general leverage architecture, followed by highlighting the operation of each type of leverage (based on atomic action and on intelligent action) and providing some examples of leverages implemented in accordance with the proposed architecture. We introduce the concept of leverage catalog at the end of this section.

A. Leverage architecture

A leverage is composed of three parts: parameters, decision logic, and internal data.

1) Parameters part: The parameters part contains all the parameters that a leverage can have on its creation. It can be any type of setting such as the initial state of leverage, the duration of application of the leverage, or the logical component on which it is applied. Each leverage can have its own set of parameters.

2) Decision logic: The decision logic is the state transition function introduced in the leverage formalization section. It contains the algorithms that define the leverage logic. The decision logic can contain any type of algorithm, from a simple short function to a complex AI-based module. The execution of the decision logic results in a set of actions that will be taken by the leverage. It can be a simple action performed directly on a logical component or a sequenced set of other leverages. The decision logic can also result in no action, if, for example, a constraint is already respected.

3) Internal data: The internal data can be thought of as an extended version of the leverage state that additionally stores other data. It contains information relative to the leverage creation and execution. Initially, it only contains the leverage initial state and is filled when the leverage is executed and can be used by any other component to understand the state of the leverage. Internal data usually contains the current leverage state, the leverage execution state, timestamps, actor information, a list of child leverages, and a list of initiated actions.

- **Leverage execution state** describes the current state of the leverage (Started, Applied, Failed).
- **Timestamps part** contains timestamps relative to leverage execution phases (Start time, Apply time, Fail time).
- **Actor information** contains information about the actor who created the leverage.
- **Child leverages list** contains references to the leverages created by the leverage decision logic.
- **Initiated actions list** contains information about the actions initiated by the decision logic.

B. Leverage instance

In order to apply a leverage on a logical component, an actor creates a leverage instance. Therefore, the leverage instance can be defined as a leverage applied to a logical component.

C. Leverage duration

A leverage can be of two types depending on its duration: instant and temporary.

1) Instant leverages: Instant leverages can be thought of as leverages that only change the state of the logical component. Its state transition function performs a series of immediate actions or creates a set of other instant leverages on a set of logical components. One example of this type of leverage is the shutdown leverage, which simply shuts down a specific compute node.

2) Temporary leverages: Temporary leverages are leverages applied for a period of time. The state transition function of temporary leverages creates two sets of actions or leverages: the first set changes the state of the leverage, and the second set returns the leverage to its original state. An example of this leverage is temporary power capping, which ensures that a defined power capping is respected for a predetermined amount of time. In this example, two sets of leverages are created; the first set imposes the power constraint, and the second set loosens it.

D. Atomic action-based leverage implementation

An atomic action-based leverage has a finite set of states and a deterministic and context-independent decision logic that performs a simple action on a single logical component.

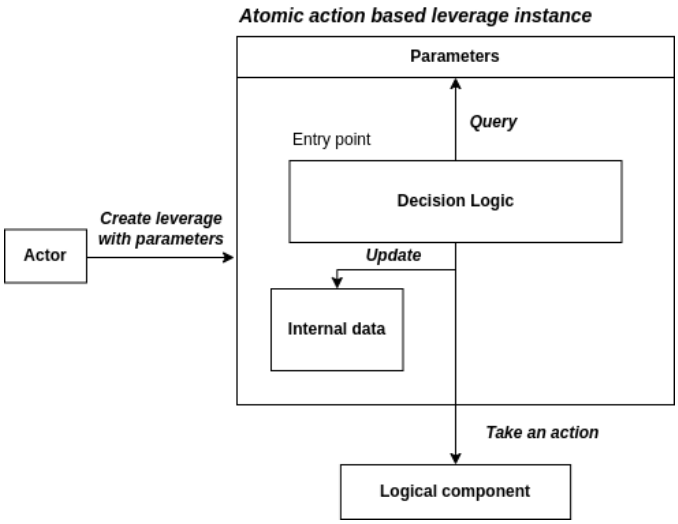

Fig. 3. Operation of an atomic action-based leverage.

1) Atomic action-based leverage operation: In Figure 3, we represent the creation and operation of an atomic action-based leverage. An actor creates a leverage instance with parameters. The decision logic is the entry point and therefore begins to execute. To make a decision, the decision logic queries the leverage parameters and determines whether the leverage can be applied with the parameters provided. Then, the decision logic performs an action on the logical component. During execution, the decision logic updates the internal data of the leverage.

Next, we illustrate some examples of atomic leverages with their modeling according to the proposed architecture.

2) RAPL leverage: Limit the instantaneous power consumption to 40 Watts of processor X of compute node Y.

- **Parameters**: RAPL leverage has two parameters: value of power limit in Watts and the logical component on which the power limit is applied.
- **Decision logic**: Verify if the power limit value can be applied on the specified logical component and apply the power limit.
- **Internal data**: Contains the current leverage state, the leverage execution state information, timestamps and actor information.

3) Workload migration leverage: Migrate the workload X from compute node A to compute node B.

- **Parameters**: Workload migration leverage has three parameters: the source compute node, the destination compute node, and the workload on which the migration is applied.
- **Decision logic**: Verify the parameters provided, and initiate and control the migration process.

- **Internal data**: Contains the current leverage state, the leverage execution state information, timestamps and actor information.

4) Compute node deployment leverage: Deploy a new compute node for the server rack X.

- **Parameters**: Compute node deployment leverage has one parameter: the logical component for which the compute node deployment is requested.
- **Decision logic**: Submit the compute node deployment request following the existing cloud provider's node deployment process.
- **Internal data**: Contains the current leverage state, the leverage execution state information, timestamps, actor information, and initiated action (deployment request information).

E. Intelligent action-based leverage implementation

Intelligent action-based leverage has context and/or state-dependent decision logic that usually creates other technological and logistical leverages on a set of logical components.

1) Intelligent action-based leverage creation: Figure 4 illustrates the creation and operation of an intelligent action-based leverage. An actor first creates a leverage instance with parameters. Then the decision logic begins to execute. In order to make a decision, decision logic queries leverage parameters, logical component states, and the external context. During execution, the decision logic updates the internal data of the leverage. In the example illustrated, the decision logic creates a set of atomic action-based leverage instances with parameters. The decision logic of the created atomic action-based leverage instances queries the leverages parameters and takes precise actions on the logical components.

Next, we illustrate some examples of intelligent action-based leverages with their modeling according to the proposed architecture.

2) Temporary power capping at data center scale: Reduce instantaneous power by 10% on data center X for one hour.

- **Parameters**: Temporary power capping leverage has three parameters: the logical component on which the power capping is applied, the power reduction objective, and the duration of the power capping.
- **Decision logic**: Review inventory and states of logical components to understand current power consumption, decide on which logical components and which atomic action-based power capping leverages can be applied and with what parameters, and create two sets of atomic action-based power capping leverage instances on selected logical components. The first set of atomic action-based leverage instances enforces the power capping and the second set of atomic action-based leverage instances releases the power constraint after 1 hour.
- **Internal data**: Contains the current leverage state, the leverage execution state information, timestamps, actor information, and a list of created atomic action-based power capping leverages.

3) Compute node end-of-life management: Manage the decommissioning of data center X compute nodes that have reached their end of life.

- **Parameters**: Compute node end-of-life management leverage has one parameter: the logical component on which it is applied.
- **Decision logic**: Examine the logical component inventory to detect compute nodes in the data center that have reached their end of life, and create compute node orders following the cloud provider's existing compute order process for each node that has reached its end of life. Once the node is delivered and installed, initiate the compute node decommissioning process.
- **Internal data**: Contains the current leverage state, the leverage execution state information, timestamps, actor information, and a list of created atomic action-based leverages.

F. Leverage catalog

The leverage catalog is a database that contains a catalog of all the leverages available in the system with their characteristics and corresponding tables of leverages.

The leverage catalog is used by the decision logic of intelligent action-based leverages to define the available leverages applicable to a particular logical component and to retrieve their tables of leverages.

VII. Environmental Gantt chart

The environmental Gantt chart can be defined as a timeline of a logical component's entire life cycle. It contains a list of leverage instances and external actions that have been applied to a logical component since the beginning of its life.

The environmental Gantt chart with the internal data of each leverage instance positioned on it can be seen as the global state of the system.

A. Leverage positioning

In order to perform an action on a logical component, an actor positions a leverage instance on the Gantt chart by creating a leverage with the logical component as parameter. A leverage can be positioned for immediate execution (Immediate leverage) or can be planned for future execution (Planned leverage).

B. External actions

The environmental Gantt chart permits the tracking of actions performed on logical components outside of the management system in order to track their entire life cycle without interfering with the current data center management process. External actions are represented on the Gantt chart by specific labels that contain information about the external actions and are placed on logical components. The representation of these actions is required for interoperability purposes.

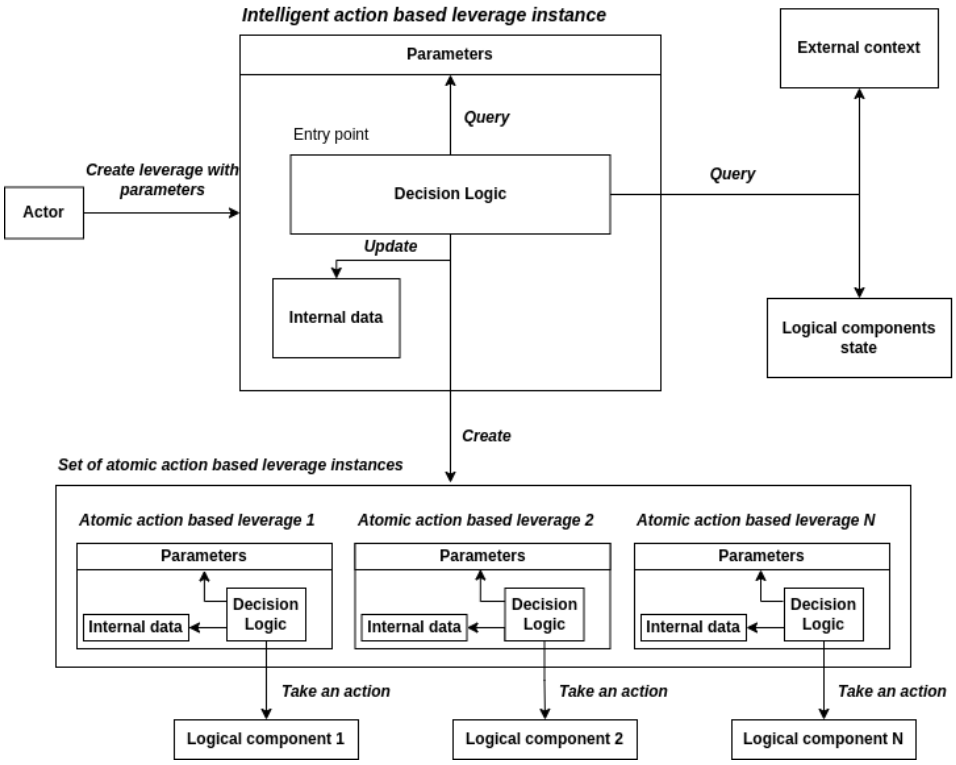

Fig. 4. Creation and operation of an intelligent action-based leverage.

C. Metrics tracking

The environmental Gantt chart can be used to track metrics to quantify the impacts and gains (energy, CO2, and greenhouse gases) of a particular leverage combination placed on it. This feature can be used to reduce the environmental impact of the data center infrastructure.

D. Other usage scenarios

The environmental Gantt chart can also be used for the following purposes:

- **Dashboard** - overview of the current state of the logical component inventory.
- **Life Cycle Assessment** - follow the use and evolution of logical components throughout their life cycle.
- **Replay** - replay a portion of the past in order to improve environmental metrics.
- **Analyze** - monitor and analyze a specific logical component.

E. Example of an environmental Gantt chart

In Figure 5, we illustrate an example of the life cycle of a compute node represented on an environmental Gantt chart.

At the beginning, the data center with the server rooms, racks and compute nodes is created, which is represented by the external action ”Create”. Following that, the data center operator enforced a data center-wide power capping by applying the intelligent action-based ”Power capping” leverage at the scale of the data center. The decision logic of this leverage created an atomic action-based RAPL leverage instance on the CPU of compute node 1. Next, the compute node 1 CPU was replaced, which is referenced on the Gantt chart by the external action ”CPU Replace”.

Finally, after several years of operation, compute node 1 has reached its end of life. The node decommissioning was scheduled by placing the logistical intelligent action-based ”End-of-life management” leverage. This leverage placed a set of atomic action-based leverages on the Gantt chart to manage the compute node decommissioning process. The ”Compute node order” logistical leverage is the first placed for immediate execution. This leverage initiates a compute node order following the data center provider node order process. The ”Migration” technological leverage, which manages the migration of compute node 1 workload, is scheduled after the end of the compute node order process. The decommissioning process is finalized by ”Shutdown” and ”Decommission” leverages which are placed for execution after the end of the workload migration.

VIII. Leverage management framework

The leverage management framework manages the leverages and places them on the environmental Gantt chart. It is the only entity that can directly place the leverages on the Gantt chart. When a cloud provider’s operator wants to apply a leverage to a logical component, he always goes through the leverage management framework.

The leverage management framework can place a leverage or a set of leverages on the Gantt chart for a variety of reasons,

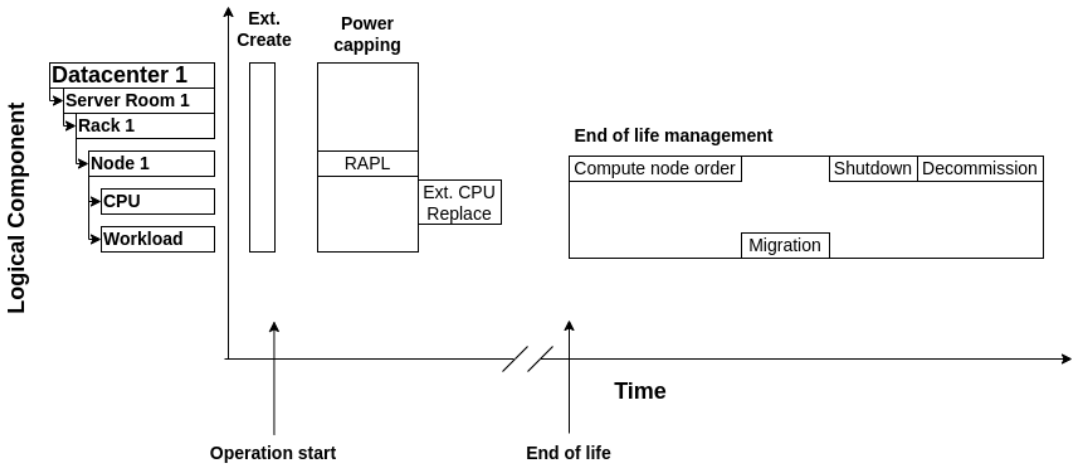

Fig. 5. Gantt chart illustrating the life cycle of compute node 1.

including meeting constraints, responding to a request of a cloud provider operator, and reducing impacts.

Furthermore, the leverage management framework monitors leverage execution to detect and respond if a leverage becomes stall, is unable to complete, is completely broken, or is applied incorrectly.

A. Meeting constraints

One of the main features of the leverage management framework is the creation of leverages in order to meet operational constraints. The operational constraints to be met may be the quality of service, the energy budget, the cooling capacity, the instance migration threshold, or the application availability threshold.

B. Reducing impacts

The leverage management framework can combine and place leverages on the Gantt chart in order to reduce the impacts of logical components. Impacts that can be reduced include energy, carbon, and greenhouse gas emissions.

C. External action management

The leverage management framework is in charge of retrieving external actions and placing them on the environmental Gantt chart.

To accomplish this, the leverage management framework implements an interface with the existing action management platform of a cloud provider, referencing actions performed on logical components. When an external action is completed and added to the action management platform, it is retrieved and placed on the Gantt chart.

IX. Experimental validation

In order to implement and validate our approach, we conducted a set of experiments and implemented proof of concepts of two scenarios. The first scenario concerns the temporary power capping at the scale of a data center by using only technological leverages. The second scenario consists of minimizing the carbon footprint of a cloud provider's infrastructure by improving its management process by combining both technological and logistical leverages.

A. Experimental environment

In this section, we describe the experimental environment that was used to conduct experiments.

1) Infrastructure: All experiments were carried out on the *Gros* cluster of the large-scale test bed for experimental research called Grid'5000 Balouek et al. (2013). This cluster contains 124 nodes that are equipped with high-performance external power meters.

This cluster's nodes have the following specifications:

- System model: Dell PowerEdge R640
- CPU: Intel Xeon Gold 5220 (Cascade Lake-SP, 2.20GHz, 1 CPU/node, 18 cores/CPU)
- Memory: 96 GB

2) Power monitoring: The power consumption of each node in the cluster is individually monitored by a high-precision external power meter OmegaWatt[2]. This power meter has a maximum sampling frequency of 50 Hz and a precision of 0.1 W. During our experiments, we used the power meter with a sampling frequency of 1 Hz, which we consider sufficient for our study.

B. Scenario 1 with technological leverages : Temporary power capping at data center scale

1) Global context: The electricity provisioning for a data center is challenging and is constrained by power bounds. In the current energy context in Europe, where the lack of electricity can cause global power outages, the ability to temporarily limit the instant power consumption at the scale of a data center can be an important requirement.

[2]OmegaWatt is a company producing high-precision external power meters. (http://www.omegawatt.fr/)

The most trivial and intuitive way to limit the instant power consumption of a data center is to use the shutdown leverage on a set of compute nodes. However, the direct use of shutdown leverage not only has an impact on the user by canceling his workload but also has a significant cost in terms of energy and time during switch on and off Raïs et al. (2018b). Furthermore, the massive switch on/off can cause the non-respect of power bound and thermal constraints.

The other way to reduce instant power consumption is to use less aggressive power capping leverages. These leverages allow limiting the power consumption to a certain threshold by slowing down the components of a compute node which could avoid the cancellation of the user's workload. Furthermore, these leverages have lower state transition costs in terms of time and energy Zhang and Hoffman (2015).

These leverages make use of techniques such as DVFS (Dynamic Frequency and Voltage Scaling), RAPL (Running Average Power Limit), or NVIDIA GPU power limiting and can be used to reduce or limit the power consumption of the various components of the compute nodes (CPU, RAM, GPU).

While applying these leverages on each individual node is fairly straightforward, using them at the scale of an entire data center to meet the overall power cap is not a trivial task.

In this section, we propose to use the leverage modeling, the Gantt chart concept, and the leverage management framework discussed earlier to apply power capping at the scale of an entire data center. The validation of the approach was carried out by simulating a small data center composed of five homogeneous compute nodes without any priority and with the same specifications as the nodes of the *Gros* cluster.

2) Leverage modeling: In this section, we highlight how the leverage modeling is used for the temporary power capping scenario. We propose to use two atomic action-based leverages and one intelligent action-based leverage to apply a temporary power capping at the scale of a data center.

a) Atomic action-based power capping leverages:

- **RAPL leverage** allows limiting the power consumption of a CPU of a compute node.
- **Shutdown leverage** allows saving energy by switching off compute nodes.

b) Intelligent action-based power capping leverage: An intelligent action-based power capping leverage allows power capping at the scale of an entire data center through the creation of atomic action-based power capping leverage instances on lower-level logical components, such as compute nodes.

The creation of atomic action-based leverage instances with the needed parameters on the logical components is performed by its decision logic. First, intelligent action-based leverage fetches the data center's actual instantaneous power consumption from the data center's logical component state. If the power constraint is not satisfied, an atomic power capping leverage that has the least impact on a compute node operation is selected from the leverage catalog. In this scenario, the atomic action-based leverage having the least impact is the RAPL leverage.

Then, based on the table of impacts of the atomic action-based leverage and the data center inventory, intelligent action-based leverage is able to determine whether applying RAPL leverage with specific parameters to a group of compute nodes will make the data center meet the power constraint.

In this case, two sets of RAPL leverage instances affecting the less compute node's operation are created and placed on the Gantt chart. The first set is tightening the power capping constraint, and the second is loosening it.

If the power constraints cannot be satisfied using only the RAPL leverage, the decision logic memorizes the most impacting parameters of the previously evaluated leverage and explores the next leverage from the leverage catalog. At this time, no leverage is placed on the Gantt chart. In this scenario, the next explored leverage is the shutdown leverage.

The decision logic explores the table of leverages of the shutdown leverage and decides which nodes can be powered off in addition to applying the RAPL leverages in order to meet the power constraint.

The decision regarding the shutdown leverage is then combined with the RAPL leverage decision for the nodes that are not impacted by the shutdown. This combination of leverage instances with selected parameters is then created and placed on the Gantt chart.

Similar to RAPL leverage, for shutdown leverage, two sets are placed on the Gantt chart to power off previously selected nodes and then power them on to restore the state before the leverage application.

The global outcome of the decision logic execution is a set of RAPL and/or Shutdown leverages placed on the Gantt chart allowing the power constraint to be respected. If the decision logic is unable to identify a leverage configuration that satisfies the power constraint, no leverage instance is created and placed on the Gantt chart.

3) Application and execution of the intelligent action-based leverage: An actor places the previously introduced intelligent action-based leverage on the Gantt chart on the data center logical component with a power constraint and a duration. The power constraint is expressed as a percentage of maximum data center power consumption and represents the power reduction objective.

The intelligent action-based leverage applies the following actions:

- The leverage creates two sets of atomic action-based leverage instances following the decision logic described earlier.
- After the execution of the first set of atomic action-based leverages, a set of compute nodes are slowed down by the RAPL atomic action-based power capping leverage or powered off by the shutdown atomic action-based leverage.
- The set of slowed down or powered-off compute nodes can have any type of configuration, from being empty to containing all data center nodes that have been powered off.

- After the duration of the intelligent action-based leverage, the second set of atomic action-based leverages is executed, the RAPL constraint is released, and all previously powered off nodes are powered on.

In Figure 6, we show an example of a temporary intelligent power capping leverage positioned on the Gantt chart and applied at the data center scale. The intelligent action-based power capping leverage is placed on the "Data Center 1" logical component with the power reduction objective of 10% and a duration of 1 h. The decision logic creates and places two sets of atomic action-based leverage instances on the Gantt chart. The first set contains four RAPL atomic action-based power capping leverage instances with the power parameter of 0.2 kW and one atomic action-based shutdown leverage instance that powers off the compute node 5. The second set is scheduled in 1 hour and contains four RAPL leverage instances relaxing the power capping and a shutdown leverage instance that powers on the compute node 5. In the end, after executing two sets of leverages, the intelligent action-based power capping leverage monitors the data center power state to ensure the power constraint has been lifted.

4) Leverage impact evaluation: In order to unfold the previously described scenario, we first need to evaluate the impact of atomic action-based leverages on the cloud provider's infrastructure and build their tables of leverages. For this scenario, we study the impacts of the RAPL and Shutdown leverages on the nodes of the *Gros* cluster described earlier.

a) RAPL leverage calibration: RAPL leverage is assessed by measuring power consumption with an external power meter while executing the EP (Embarrassingly parallel) kernel of the NAS parallel benchmarks Bailey et al. (1994). EP (Embarrassingly parallel) kernel generates pairs of Gaussian random deviates and makes intensive use of the compute node's CPU. It has almost no inter-processor communication requirements and does not perform any memory operations.

We evaluate and calibrate RAPL leverage with 12 different RAPL power limits ranging from 120 W to 10 W with 10 W decrements. The RAPL power limit is set to the package power domain that includes the entire CPU package, including the DRAM memory controller Khan et al. (2018).

In Table III, we show the table of leverages of RAPL leverage resulting from our assessment. We can see that the RAPL power limitation performs decently in our environment (*Gros* cluster nodes running CPU-bound workload). The power consumption of the entire compute node measured by the external power meter is reduced by almost the same amount as the RAPL power limit. At the 30 W power limit, it reaches a lower bound. There is almost no decrease in the compute node's power consumption after this point. The 125 W power limit value is the RAPL default configuration and is considered as the value when power limiting is not enforced.

However, we must be aware that the table of leverages of RAPL leverage is workload dependent, and its values depend on the application executed when evaluating impacts. Therefore, we must evaluate the leverage impacts and construct a table of leverages on a per-application/workload basis.

TABLE III
TABLE OF LEVERAGES OF RAPL LEVERAGE (MEDIAN ON 10 EXPERIMENTAL MEASUREMENTS).

Power limit	Median compute node power
125 (no limit)	175.89 W
120	169.91 W
110	159.13 W
100	148.57 W
90	137.59 W
80	126.85 W
70	116.10 W
60	105.89 W
50	95.87 W
40	85.31 W
30	76.43 W
20	74.01 W
10	73.72 W

b) Shutdown leverage calibration: Shutdown leverage is calibrated and measured by monitoring power consumption with an external power meter and evaluating the transition time between leverage states. In Table IV, we show the table of leverages of shutdown leverage resulting from this assessment.

E_{OffOn} and T_{OffOn} represent the energy and time required to power on the compute node. E_{OnOff} and T_{OnOff} represent the energy and time required to power it off. P_{idle} and P_{off} represent the median instant power of the compute node in the idle and powered-off states.

This table is used by the decision logic of the intelligent action-based power capping leverage.

In order to simplify this scenario, we do not take into account the state transition costs of shutdown leverage and only use the median instant power of the compute node.

TABLE IV
TABLE OF LEVERAGES OF SHUTDOWN LEVERAGE (MEDIAN ON 10 EXPERIMENTAL MEASUREMENTS).

Parameters	Values
E_{OffOn}(Joules)	20,952.2
T_{OffOn}(Seconds)	192.4
E_{OnOff}(Joules)	52.9
T_{OnOff}(Seconds)	2.6
P_{idle}(Watts)	53.2
P_{off}(Watts)	4.4

5) Results: In this section, we discuss the results obtained by using our implementation of the leverage management framework with the environmental Gantt chart for temporary power capping at the scale of a data center.

In Figure 7, we show the data center's current power state and power objective when running the power capping scenario with various power reduction objectives. The scenario was unrolled with power reduction objectives ranging from 10% to 90%. We can point out that the framework is able to make the data center respect the power constraints for any tested objective value.

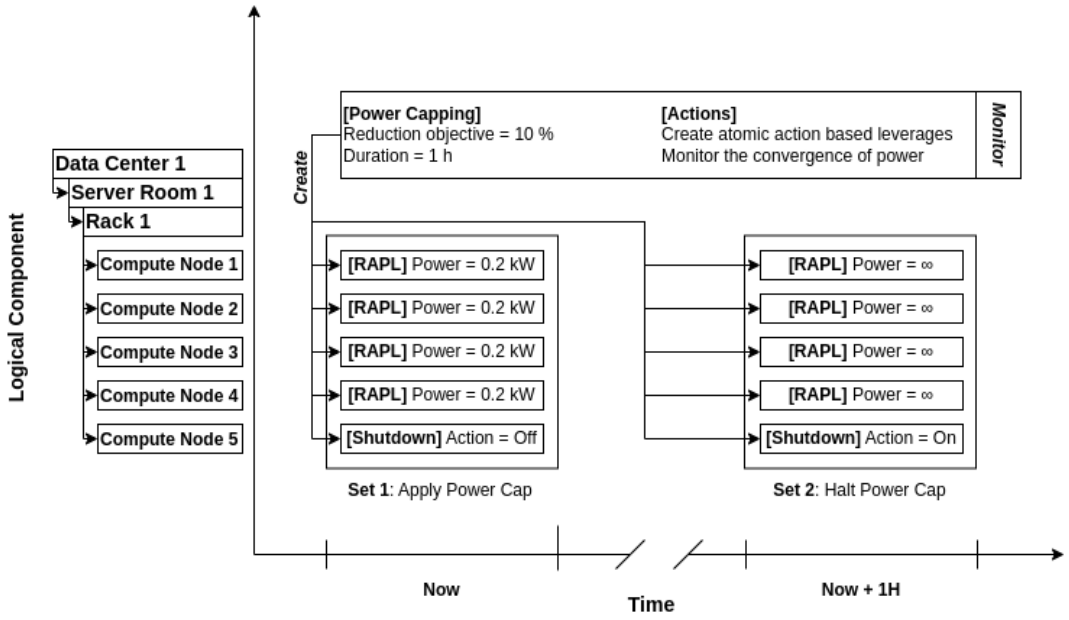

Fig. 6. Example of temporary intelligent action-based power capping leverage positioned on the Gantt chart.

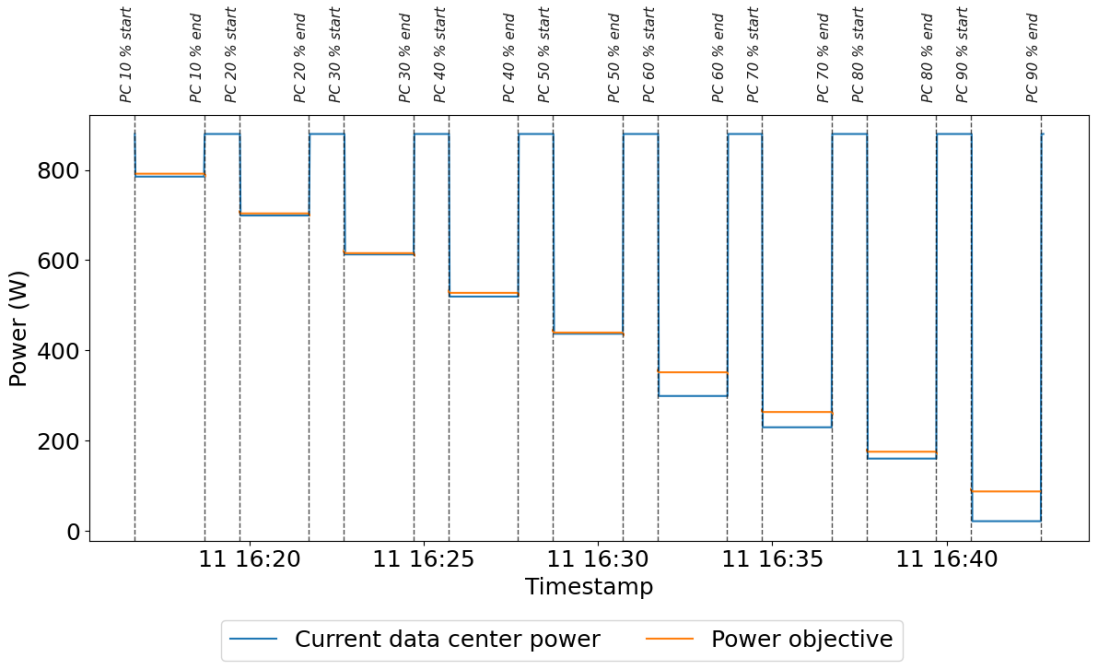

Fig. 7. Data center power state when running the power capping scenario with various power reduction objectives.

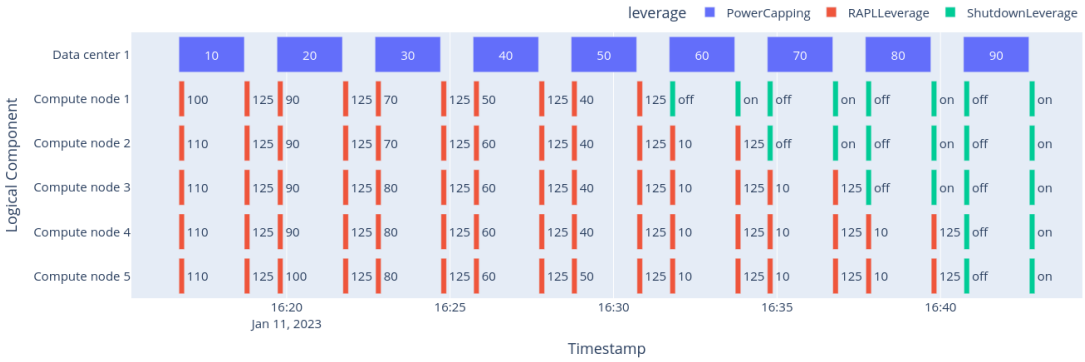

Fig. 8. Power capping scenario execution Gantt chart with intelligent action-based Power Capping leverages, RAPL, and Shutdown atomic action-based leverages.

In Figure 8, the intelligent action-based power capping leverages are positioned at different moments with various power objectives on the "Data center 1" logical component. Each intelligent action-based power capping leverage placed a set of RAPL and Shutdown leverages on the lower-level logical components to meet the power objective of the entire data center.

We can point out that intelligent action-based power capping leverage is able to reduce data center power consumption by 50% by only applying the RAPL leverages on the compute nodes. Starting from the 60% power reduction objective, the intelligent action-based power capping leverage places shutdown leverages that power off the compute nodes to respect the constraint. In order to reduce data center power consumption by 90%, the intelligent action-based power capping leverage places shutdown leverages that power off all compute nodes. After each intelligent action-based power capping leverage execution, the RAPL and Shutdown leverages are placed to return the compute nodes to their initial states.

6) Added value/discussion/limits on explored scenario: The implementation of the first scenario allowed us to demonstrate that the proposed leverage management framework makes it possible to apply a temporary power capping at the scale of an entire data center.

We validated our approach by simulating a small data center of five homogeneous compute nodes taking into account only their consumption. In the real world, data centers have a much higher number of compute nodes and contain other power-consuming components such as networking, cooling, and power distribution systems that must be considered. In addition, a data center contains compute nodes that are often very heterogeneous. In order to use our framework under these conditions, we need to assess each atomic action-based leverage impact for each compute node configuration available in the data center and explore leverages impacting other components of a data center facility.

As the consumption of a compute node with exactly the same specifications within the same cluster can be different and can change over time Diouri et al. (2013), Heinrich et al. (2017), our framework must consider the actual measured power consumption in addition to the tables of leverages in its decision-making process.

C. Scenario 2 with technological and logistical leverages : Minimizing the global carbon footprint

1) Global context: The typical cloud infrastructure is composed of heterogeneous compute resources executing various workloads.

The flexibility of the cloud infrastructure is one of the most important requirements since the customer's workload may be highly variable. This flexibility is accomplished by keeping a pool of idle compute nodes ready to handle client workloads as needed. The compute nodes in this pool are powered on to handle in a timely manner a potential increase in client demand, but by default do not run any client workloads. A client workload can be scheduled on an idle pool compute node, in which case it becomes a production node. The size of the idle pool is typically determined by the cloud provider and must be maintained in order to meet the flexibility constraint.

As the global use of cloud services grows, data center managers must handle this expansion by regularly adding new compute nodes.

Furthermore, each compute node has a lifetime after which it must be decommissioned and replaced by a new one. Because the manufacturing, transporting and decommissioning of compute nodes has a significant carbon impact, it must be calculated and taken into account to understand the overall carbon footprint.

However, while most cloud service providers have established infrastructure growth and idle/production pool management, optimizing idle/production pool energy efficiency and proper node decommissioning is rarely done.

In this context, we propose to use the leverage modeling, the Gantt chart concept, and the leverage management framework to reduce the overall carbon footprint of the cloud provider's infrastructure.

In order to achieve this, we first optimize the production/idle pools by increasing the proportion of energy-efficient compute nodes handling the production workload and powering off the greatest number of idle pool nodes. The idle pool compute

nodes are powered off in a way that ensures there are always enough powered-on nodes to handle client workloads.

Then, we manage the decommissioning of nodes that have reached their end of life and the periodic deployment of new compute nodes in order to keep up with the ever-increasing workloads.

Before applying the previously proposed modeling, we introduce some additional elements that are important in the context of the minimization of the global carbon footprint scenario.

2) Workload: In a cloud environment, a workload mostly represents virtual machines or containers running on compute nodes that can be live-migrated with minimal costs.

3) Infrastructure: The cloud computing infrastructure is typically composed of heterogeneous compute nodes shared between multiple clients. Since compute nodes are heterogeneous, they may have different energy efficiency and lifespan. When a compute node reaches its end of life, it ought to be decommissioned.

At the beginning of this scenario, the cloud provider has 100 heterogeneous compute nodes. Each node has an energy efficiency score assigned based on its age.

Each compute node is part of a compute node pool.

4) Compute node pools: In this scenario, we distinguish two types of compute node pools: production pools, which contain nodes that are executing client workloads, and idle pools, which contain nodes that are not executing any client workloads.

Initially, we consider that the idle pool contains 50 nodes and the production pool contains 50 nodes. We assume that, by default, the cloud provider does not perform any compute node pool optimizations, so all nodes are powered on regardless of whether they are running a workload or not.

Each compute node is ranked in the pool based on its energy efficiency score.

We assume that the idle compute node pool must have at least 10 nodes powered on and ready to accept the client workload to meet the flexibility constraint.

5) Workload growth: Workload growth should be managed by adding new compute nodes as needed. Although workload growth is constant, it is not uniform. To simplify our example, we assume that the cloud provider needs an additional compute node every week.

6) Workload variation: In order to simulate the flexibility of a cloud provider, we add the workload variation in our scenario. The workload variation is determined daily and may require from 1 to 40 additional production nodes. These nodes are always provided by the idle pool. Since the idle pool contains 50 nodes, it can always serve the variation in workload while meeting the 10 powered-on nodes constraint and ordering new nodes for this purpose is never required.

7) Leverage modeling: In this section, we demonstrate how leverage modeling can be used to minimize cloud infrastructure's overall carbon footprint. We propose to use two atomic action-based technological and three atomic action-based logistical leverages coupled with five intelligent action-based leverages in order to reduce the carbon footprint of previously described cloud infrastructure.

a) Atomic action-based leverages:

- Technological
  - **Shutdown leverage** allows saving energy by switching off compute nodes.
  - **Workload migration leverage** migrates a workload from one node to another to free up a compute node.
- Logistical
  - **Compute node order leverage** uses the existing cloud provider's compute node ordering process.
  - **Compute node deployment leverage** uses the existing cloud provider's compute node deployment process.
  - **Compute node decommissioning leverage** uses the existing cloud provider's compute node decommissioning process.

b) Intelligent action-based leverages: The scenario described in this section requires several intelligent action-based leverages, each serving its own purpose and intervening in different parts of the infrastructure management process to minimize the overall carbon footprint of the cloud provider's infrastructure.

c) Pool re-balance leverage: The first intelligent action-based leverage is the leverage that re-balances production and idle pools. The main goal of this leverage is to maximize the number of energy-efficient nodes in the production pool. This leverage is programmed on a daily basis. This intelligent action-based leverage applies the following actions:

- It begins by examining the energy efficiency of the compute nodes in both pools.
- If the least energy-efficient node in the production pool has a lower energy efficiency than any node in the idle pool, its production workload is migrated to the most energy-efficient node in the idle pool.
- Therefore, the production node freed from the workload becomes an idle node and the idle node hosting the workload becomes a production node.
- Workload migration is performed by placing the workload migration atomic action-based leverage on the Gantt chart.
- The process is repeated and migration leverages are placed until the least efficient node in the producing pool outranks the most efficient node in the idle pool.
- At the end of the pool re-balancing process, the production pool contains only the most energy-efficient nodes.

d) Idle pool management leverage: The purpose of this leverage is to reduce the number of idle nodes that are powered on while meeting the 10 powered-on nodes constraint and taking into account the daily variation of the client workload. This leverage is also programmed on a daily basis.

This intelligent action-based leverage applies the following actions:

- The leverage monitors the number of idle nodes and the value of the current day's client workload.

- The leverage powers on the compute nodes if the number of idle powered-on nodes is not sufficient to host the day's workload and meet the constraint of 10 idle nodes.
- The most energy-efficient powered-off nodes are powered on by placing a shutdown atomic action-based leverage with the power-on parameter on the Gantt chart.
- The leverage places powering off shutdown atomic action-based leverages if there are more nodes powered on than are necessary.

e) End-of-life management leverage: The goal of this leverage is to manage the end-of-life of compute nodes. If a node reaches its end of life, it should be replaced with a new more energy-efficient node. Leverage monitors the production and idle pools for nodes that have reached their end of life, and these nodes are marked for decommissioning. The compute node order, deployment, and decommissioning atomic action-based leverages are placed on the Gantt chart. The compute node order leverage is placed for immediate execution. The compute node deployment leverage is scheduled for execution after order leverage and the compute node decommissioning leverage is planned after the deployment leverage. Therefore, the compute node is not fully decommissioned until a new node has been delivered and deployed. This leverage is programmed on an annual basis.

f) Increasing workload management leverage: As introduced earlier, the constantly increasing workload requires an additional compute node every week. In order to manage this, increasing workload management leverages are placed weekly on the Gantt chart by the leverage management framework. The increasing workload management leverage places the compute node order and deployment atomic action-based leverages. The compute node order leverage is placed for immediate execution and the compute node deployment leverage is scheduled for execution after the order leverage.

8) Application of the intelligent action-based leverages: In order to minimize the global carbon footprint of cloud infrastructure management the leverage management framework places previously described leverages on the Gantt chart. When all leverages are executed at least once, the production pool contains only the most energy-efficient nodes, there are enough powered-on nodes for the client workload, the idle pool only has 10 most energy-efficient nodes powered on, and for each node to be decommissioned, a new compute node order, compute node deployment and compute node decommissioning atomic action-based leverage instances have been created.

In Figure 9, we show an example of previously described intelligent action-based leverages positioned on the Gantt chart. This example is given for illustrative purposes in order to better understand the operation of intelligent action-based leverages.

In this example, the intelligent action-based pool re-balance leverage detects that compute nodes 2 and 3 are members of the idle pool and are more power efficient than compute nodes 1 and 4 of the production pool. The workload from compute nodes 1 and 4 is migrated to compute nodes 2 and 3 by positioning the workload migration leverages. As a result, compute nodes 2 and 3 enter the production pool while nodes 1 and 4 leave it. Then, when the idle pool management leverage determines that there are too many nodes in the idle pool, compute node 1 is powered off. These two intelligent action-based leverages are positioned on the Gantt chart by leverage management framework on a daily basis.

The increasing workload management leverage creates compute node order and deployment atomic action-based leverage instances on the server rack 1 logical component. The compute node is ordered and will be deployed in server rack 1. The compute node deployment leverage is executed when the node is delivered at the end of compute node order leverage. The increasing workload management leverage is positioned on the Gantt chart on a weekly basis.

The end-of-life management leverage detects that the compute node 1 reached its end of life and can be decommissioned. The leverage places the compute node order and deployment atomic action-based leverages on the Gantt chart. Following that, the compute node 1 is marked for decommissioning, and compute node decommissioning atomic action-based leverage is placed after order and deployment leverages. When the compute node that is supposed to replace the compute node 1 is delivered and deployed, the compute node 1 is decommissioned. This leverage is positioned by leverage management framework on annual basis.

9) Leverage impact evaluation: For this scenario, we need to study the impacts of all atomic action-based leverages. The impact study of the shutdown leverage was carried out for the previous scenario and is reused for this scenario.

a) Workload migration leverage: Assessing workload migration leverage is a non-trivial task, as it depends on the environment the workload is running in, the virtualization solution used, and the amount of memory and storage used by the workload. If the workload is running inside of a virtual machine, its memory and storage must be copied to the destination compute node. After that, the virtual machine on the source node can be suspended and it can be started on the destination node. This process is called cold virtual machine migration. Therefore, we must take into account the cost of transferring memory and storage through the network as well as the cost of stopping and starting the virtual machine when evaluating the workload migration costs. In this scenario, we assume that workload migration costs are the same as a network file transfer. Thus, in order to assess the workload migration leverage, we evaluate the time and energy costs of transferring files of various sizes between two compute nodes over the network. The experiments were performed on *Gros* cluster nodes and the power consumption is monitored by the external power meters.

In Table V, we show the table of leverages of workload migration leverage resulting from our evaluation. The leverage management framework makes use of this table to calculate the costs associated with workload migration.

b) Compute node order leverage: The compute node order leverage has significant costs in terms of time and

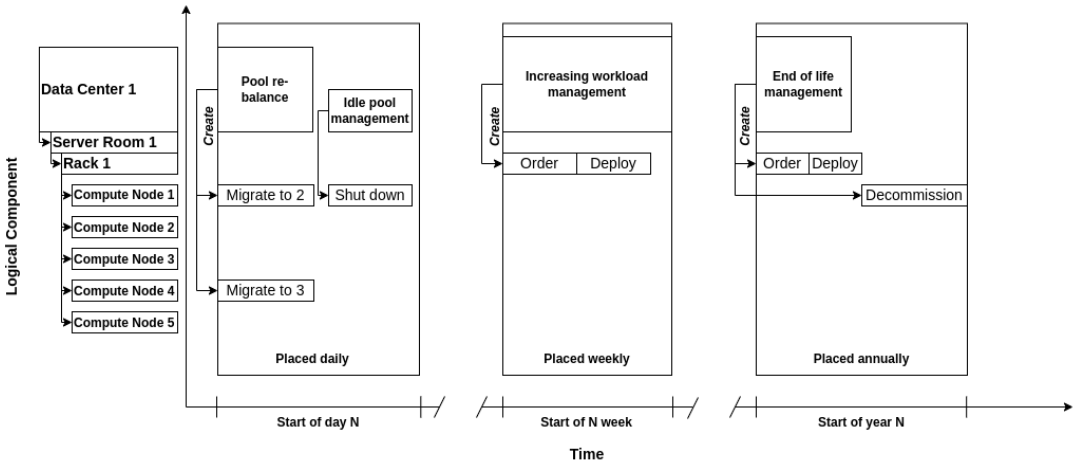

Fig. 9. Example of the positioning of intelligent action-based leverages on the environmental Gantt chart to minimize the carbon footprint of cloud infrastructure management.

TABLE V
TABLE OF LEVERAGES OF WORKLOAD MIGRATION LEVERAGE (MEDIAN ON 10 EXPERIMENTAL MEASUREMENTS).

Workload size	Time	Energy consumed
1 Gigabyte	5.39 Seconds	530.99 Joules
2 Gigabytes	8.51 Seconds	1058.70 Joules
4 Gigabytes	14.70 Seconds	1887.15 Joules
8 Gigabytes	27.02 Seconds	3615.05 Joules
16 Gigabytes	46.19 Seconds	6385.31 Joules
32 Gigabytes	83.13 Seconds	12487.41 Joules
64 Gigabytes	157.30 Seconds	25161.75 Joules

an important carbon footprint. In the context of a lack of electronic components, the delivery time of a compute node may exceed several months. Moreover, the order of the compute nodes has a significant carbon cost. We must consider the carbon footprint of manufacturing and transporting the compute node, which is frequently estimated and provided by the manufacturer. For this scenario, the compute node delivery time is estimated to be 3 weeks from order placement. The carbon footprint of the manufacturing and transport phases of a Dell PowerEdge R640 server was taken as a reference. The footprint is estimated by the manufacturer at 1306.37 $kgCO_2eq$. This value is used as the carbon cost of a compute node order leverage.

c) Compute node deployment leverage: Compute node deployment time may vary depending on the cloud provider's internal deployment process. In this scenario, we do not consider the costs of deploying the compute nodes and assume them to be zero.

d) Compute node decommissioning leverage: The decommissioning of compute nodes has an impact in terms of carbon emissions. This impact depends on how the compute node is decommissioned. After decommissioning, the compute node or its components can be reused elsewhere or simply destroyed. All of this has an effect on the carbon footprint of the compute node decommissioning process. In this scenario, we estimate this cost to be 1% of the compute node's estimated overall carbon footprint.

10) Results: In this section, we discuss the results obtained by using the leverage management framework with the environmental Gantt chart to minimize the global carbon footprint of a cloud provider's infrastructure management process described earlier.

First, we simulated the standard infrastructure management process used by cloud providers with our leverage management framework. The standard management only includes compute node decommissioning and increasing workload management processes. The compute nodes are decommissioned by the cloud provider on an annual basis. A supplementary node is added to each node pool every week to handle the increasing workload. We calculated the carbon footprint of standard management processes over a 10-year period.

Then, we simulated the carbon footprint of the minimization scenario management by adding pool re-balance and idle pool management leverages to the standard management process. We also changed the increasing workload management process by adding a single node to the production pool, instead of adding a node to each pool. This measure avoids overprovisioning while allowing management of the increasing workload. As for the standard management process, we calculated the carbon footprint of the minimization scenario management over a 10-year period.

In Figure 10, we show the evolution of the carbon footprint of standard and minimization scenario management techniques. We can point out that the carbon footprint of the minimization scenario is significantly smaller. The majority of savings are attributable to the less aggressive addition of compute nodes to handle the increasing workload. Figure 11 illustrating the evolution of the carbon footprint for a period of 1 year can be used to confirm this. We can mention multiple nonlinear increments presenting the carbon impact of deploying new compute nodes to manage the increasing

workload. The sharp increase at 2024-01 timestamp is due to the process of compute node decommissioning. As this process is identical between the standard and minimization scenario management, the increase is also the same. Figure 12 highlights the savings due only to pool re-balance and idle pool management leverages added in the minimization scenario. Even if they are much less significant, they are far from negligible (58.57 $TonsCO_2eq$).

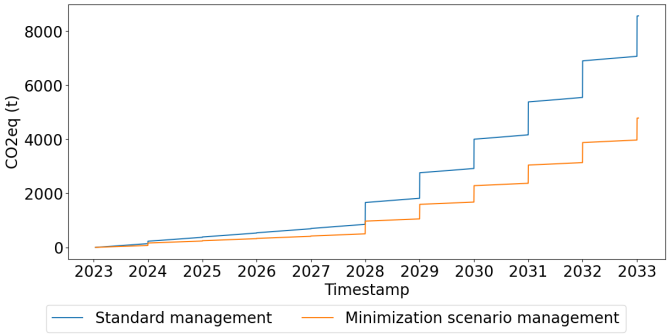

Fig. 10. Evolution of carbon footprint over a 10-year period. Comparison between standard and minimization scenario management techniques.

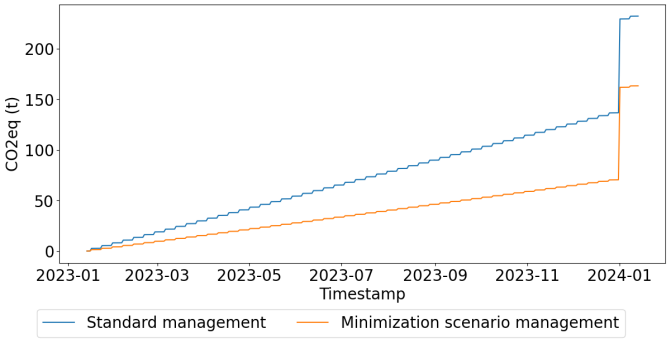

Fig. 11. Evolution of carbon footprint over a 1-year period. Comparison between standard and minimization scenario management techniques.

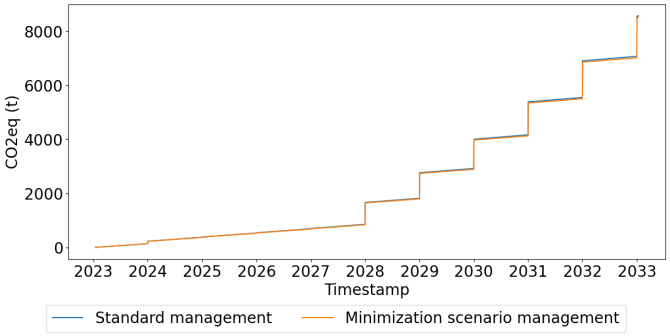

Fig. 12. Evolution of carbon footprint over a 10-year period. Minimization scenario with pool re-balance and idle pool management leverages only.

11) Added value/discussion/limits on explored scenario: In this scenario, we have demonstrated how the leverage management framework can be used to minimize the global carbon footprint of cloud infrastructure management. To accomplish this, the leverage management framework places a combination of technological and logistical environmental leverages on the environmental Gantt chart. The environmental Gantt chart provides an overview of the cloud provider's infrastructure as well as every action performed on it throughout its entire life cycle. This overview can then be used not only for accounting purposes but also for life cycle assessments and even for further infrastructure management process optimization.

In order to validate our approach, we chose a simplified version of the cloud provider management process in this scenario. We made some assumptions as the size of compute node pools, the compute node delivery times, the workload variation, and decommissioning periods that cannot be generalized to all cloud providers. These assumptions must be validated and adapted according to each cloud provider.

The cloud management process is also much more intricate in the real world, involving more actions and having an impact on more components. Due to the flexibility of the proposed leverage management framework and environmental Gantt chart, it can effectively manage and depict the majority of cloud provider actions.

X. Conclusion and Future Works

While large-scale ICT infrastructures like data centers must enter in a new era of reduced environmental impacts, providers face challenges in dealing between technological and logistical capabilities. The proposed approach opens the door to some large-scale facilities to integrate, plan, map, monitor and orchestrate heterogeneous leverages.

In this work, we proposed and validated a framework for reducing the carbon footprint of cloud infrastructure management and for temporary power capping by applying heterogeneous leverages at the scale of an entire data center.

We have demonstrated that by implementing the proposed management framework, we can optimize infrastructure management and, as a result, significantly reduce the carbon footprint of cloud infrastructure.

Using the proposed management framework for temporary power capping, we were able to demonstrate the possibility of reducing the power consumption of the entire data center by a substantial amount (50% in our example) by only using RAPL power limiting leverage without powering off compute nodes.

The leverage management framework bases its decisions on leverages impacts that have been experimentally evaluated on the nodes of the *Gros* cluster of the Grid'5000 testbed. In this work, we evaluated the RAPL power limiting, shutdown, and workload migration leverages.

We have introduced an environmental Gantt chart concept which is useful for accounting, life cycle assessment, and management process analysis purposes.

a) Future works: This work opens up a myriad of future work possibilities. First, we plan to validate the temporary power capping scenario on a real data center infrastructure containing hundreds of nodes. This validation will be done

considering not only the compute node part but also other data centers components such as cooling and power supply. For this scenario, we also intend to conduct a more thorough analysis of the RAPL power limiting leverage by separating the CPU and DRAM limitation constraints and analyzing the performance impacts on the workload. Then, we intend to integrate the proposed framework with already-existing management systems like SLURM Yoo et al. (2003) and OpenStack[3] for workload management and prediction. Finally, we plan to validate the minimizing of the global carbon footprint scenario by integrating into the leverage management framework each step of the infrastructure management process of an existing cloud provider.

Acknowledgments

This work is supported by the "FrugalCloud" Inria and OVHcloud partnership. Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several Universities as well as other organizations (see https://www.grid5000.fr).

References

Ayanoglu, E. (2019). Energy efficiency in data centers. *IEEE ComSoc Technical Committees Newsletter*, pages 1–8.

Bailey, D. H., Barszcz, E., Barton, J. T., et al. (1994). The NAS parallel benchmarks. Technical report, RNR-94-007, NASA Ames Research Center.

Balouek, D., Amarie, A. C., Charrier, G., et al. (2013). Adding virtualization capabilities to the Grid'5000 testbed. In *The 2nd International Conference on Cloud Computing and Services Science (CLOSER) 2012, Porto, Portugal, 18-21 April 2012*, pages 3–20.

Benoit, A., Lefèvre, L., Orgerie, A.-C., et al. (2017). Shutdown policies with power capping for large scale computing systems. In *23rd International European Conference on Parallel and Distributed Computing (Euro-Par) 2017, Santiago de Compostela, Spain, 28 August-1 September 2017*, pages 134–146.

Benoit, A., Lefèvre, L., Orgerie, A.-C., et al. (2018). Reducing the energy consumption of large-scale computing systems through combined shutdown policies with multiple constraints. *The International Journal of High Performance Computing Applications*, 32(1):176–188.

David, H., Gorbatov, E., Hanebutte, U. R., et al. (2010). RAPL: memory power estimation and capping. In *ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED) 2010, Austin, Texas, USA, 18-20 August 2010*, pages 189–194.

Diouri, M. E. M., Glück, O., Lefèvre, L., et al. (2013). Your cluster is not power homogeneous: Take care when designing green schedulers! In *International Green Computing Conference (IGCC) 2013, Arlington, VA, USA, 27-29 June 2013*, pages 1–10.

[3]OpenStack is a free, open standard cloud computing platform. (https://www.openstack.org/)

Haidar, A., Jagode, H., Vaccaro, P., et al. (2019). Investigating power capping toward energy-efficient scientific applications. *Concurrency and Computation: Practice and Experience*, 31(6):e4485.

Hamdi, N. and Chainbi, W. (2019). A survey on energy aware VM consolidation strategies. *Sustainable Computing: Informatics and Systems*, 23:80–87.

Heinrich, F. C., Cornebize, T., Degomme, A., et al. (2017). Predicting the energy-consumption of MPI applications at scale using only a single node. In *IEEE International Conference on Cluster Computing (CLUSTER) 2017, Hawaii, USA, 5-8 September 2017*, pages 92–102.

Khan, K., Hirki, M., Niemi, T., et al. (2018). RAPL in action: experiences in using RAPL for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 3(2):1–26.

Kim, W., Gupta, M. S., Wei, G.-Y., et al. (2008). System level analysis of fast, per-core DVFS using on-chip switching regulators. In *IEEE 14th International Symposium on High Performance Computer Architecture (HPCA) 2008, Salt Lake City, Utah, USA, 16-20 February 2008*, pages 123–134.

Krzywda, J., Ali-Eldin, A., Carlson, T. E., et al. (2018). Power-performance tradeoffs in data center servers: DVFS, CPU pinning, horizontal, and vertical scaling. *Future Generation Computer Systems*, 81:114–128.

Masanet, E., Shehabi, A., Lei, N., et al. (2020). Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986.

Mytton, D. and Ashtine, M. (2022). Sources of data center energy estimates: A comprehensive review. *Joule*, 6(9):2032–2056.

Orgerie, A.-C., Assuncao, M. D. d., and Lefèvre, L. (2014). A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Computing Surveys*, 46(4):1–31.

Rais, I., Balouek-Thomert, D., Orgerie, A.-C., et al. (2019). Leveraging energy-efficient non-lossy compression for data-intensive applications. In *International Conference on High Performance Computing & Simulation (HPCS) 2019, Dublin, Ireland, 15-19 July 2019*, pages 463–469.

Rais, I., Boutigny, M., Lefevre, L., et al. (2018). Building the table of energy and power leverages for energy efficient large scale systems. In *International Conference on High Performance Computing & Simulation (HPCS) 2018, Orleans, FR, 16-20 July 2018*, pages 284–291.

Raïs, I. (2018). *Discover, model and combine energy leverages for large scale energy efficient infrastructures*. PhD thesis, Universite de Lyon, FR.

Raïs, I., Lefèvre, L., Orgerie, A.-C., et al. (2018a). Exploiting the table of energy and power leverages. In *Algorithms and Architectures for Parallel Processing: 18th International Conference (ICA3PP) 2018, Guangzhou, China, 15-17 November 2018*, pages 175–185.

Raïs, I., Orgerie, A.-C., Quinson, M., et al. (2018b). Quantifying the impact of shutdown techniques for energy-efficient

data centers. *Concurrency and Computation: Practice and Experience*, 30(17):e4471.

Rountree, B., Ahn, D. H., de Supinski, B. R., et al. (2012). Beyond DVFS: A first look at performance under a hardware-enforced power bound. In *IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW) 2012, Shanghai, China, 21-25 May 2012*, pages 947–953.

Suleiman, D., Ibrahim, M., and Hamarash, I. (2005). Dynamic voltage frequency scaling (DVFS) for microprocessors power and energy reduction. In *4th International Conference on Electrical and Electronics Engineering (ELECO) 2005, Bursa, Turkey, 7-11 December 2005*, pages 1–5.

Yoo, A. B., Jette, M. A., and Grondona, M. (2003). Slurm: Simple linux utility for resource management. In *9th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP) 2003, Seattle, WA, USA, 24 June 2003*, pages 44–60.

Yu, S., Yang, H., Wang, R., et al. (2017). Evaluating architecture impact on system energy efficiency. *PLOS One*, 12(11):e0188428.

Zhang, H. and Hoffman, H. (2015). A quantitative evaluation of the RAPL power control system. *Feedback Computing*, 6.

AUTHOR BIOGRAPHIES

Vladimir Ostapenco is a second-year PhD student in computer science at Ecole Normale Supérieure of Lyon and Inria (the French Institute for Research in Computer Science and Control). He is a member of the Avalon team (Algorithms and Software Architectures for Distributed and HPC Platforms) from the LIP laboratory in Ecole Normale Supérieure of Lyon, France. He works on energy efficiency and environmental impacts of cloud data center infrastructures as part of the Inria-OVHcloud (FrugalCloud) challenge.

Laurent Lefevre is a permanent researcher in computer science at Inria (the French Institute for Research in Computer Science and Control). He is a member of the Avalon team (Algorithms and Software Architectures for Distributed and HPC Platforms) from the LIP laboratory in Ecole Normale Supérieure of Lyon, France. He has organized several conferences in high-performance networking and computing. He has co-authored more than 150 articles published in refereed journals and conference proceedings. Since more than 15 years, he works on energy efficiency and environmental impacts of large-scale systems (HPC centers, data centers, clouds, and big wired networks).

Anne-Cécile Orgerie received a PhD degree in computer science from Ecole Normale Superieure de Lyon, France, in September 2011. She has been a full time researcher at CNRS in the IRISA laboratory, Rennes, France, since October 2012. Her research is in the broad area of distributed systems, and in particular focuses on energy efficiency and environmental impact of cloud computing infrastructures and smart grids.

Benjamin Fichel has been working for OVHcloud as a Telecom and Network engineer for the last 15 years. Within the infrastructure group, he leads several transformation projects in order to modernize the control plane and change the paradigms of infrastructure consumption through the use of standardized APIs.