Consolidation of virtual machines to reduce energy consumption of data centers by using ballooning, sharing and swapping mechanisms

Simon Lambert
LIP Laboratory
Univ Lyon1, Inria, ENS de Lyon, Ciril GROUP
Lyon, France
simon.lambert@ens-lyon.fr

Eddy Caron
LIP Laboratory
Univ Lyon1, Inria, ENS de Lyon, CNRS
Lyon, France
Eddy.Caron@ens-lyon.fr

Laurent Lefèvre
LIP Laboratory
Univ Lyon1, Inria, ENS de Lyon, CNRS
Lyon, France
laurent.lefevre@inria.fr

Rémi Grivel
Ciril GROUP
Lyon, France
rgrivel@cirilgroup.com

***Abstract*—Data centers have major environmental impacts due to their energy consumption and the manufacturing of equipment. They emit greenhouse gases and consume energy and resources, such as rare earth and water. Efficient computing resource management is therefore a key challenge for Cloud service providers today as they need to meet a growing demand while limiting the oversizing of their infrastructures. Mechanisms derived from virtualization, such as Virtual Machines (VMs) consolidation, are used to optimize resource management and infrastructure sizing, but economic and technical constraints can hinder their adoption. They require prior infrastructure knowledge and usage study to evaluate their potential, involve complex placement algorithms, and are sometimes difficult to implement in hypervisors. In this paper, we propose *ORCA (OuR Consolidation Algorithm)*, a complete consolidation methodology designed to facilitate the production implementation of such mechanisms. This methodology includes the study of VM usage, the use of prediction models, and a VM placement algorithm that takes advantage of resource oversubscription. The choice of relevant oversubscription ratios is also addressed, with a focus on memory overcommitment through the study of memory overcommitment mechanisms:ballooning, page sharing, and swapping. Results from a detailed simulation process and deployment on a production infrastructure are presented. The methodology is also tested in simulation on two production infrastructure datasets, with power consumption reduction as high as 29.8% and without consolidation error. The production deployment using VMWare vSphere and considering fault tolerance requirements reduces the energy consumption by 6.12% without causing any performance degradation.**

***Index Terms*—Cloud, Consolidation, Energy, Virtualization, Memory overcommitment**

I. Introduction

Since 2010, we observe an important growth in Cloud services. This led to an increasing number of data centers (DC) of different topologies: from small company DCs to hyperscale cloud DCs. Despite the rise of cloud users and infrastructures, different optimizations allowed to limit the total DCs energy demand between 220 and 320TWh in 2021. This represents an increase of 10-60% between 2015 and 2021 [28, 37]. In 2021, DCs consumed around 1% of the total energy demand, but numbers vary and studies estimate this number to be as high as 4% [18].

This limit to the growth of DCs electricity consumption can be attributed to improvements at the hardware level, with more efficient cooling systems or computing nodes. Koomey's Law [33] depicts well these optimisations in CPUs, stating that the number of computations per joule of energy dissipated doubled every 1.5 years. Moreover, the last two decades have been marked by the emergence of the cloud, with the democratization of virtualization [11]. Virtualization with a good level of isolation helped containing the growing use of energy by allowing multiple Operating Systems (OS), applications, or workloads to run on a single Physical Machine (PM). It increases resource utilisation on a single server, allows a better resource management and energy efficiency when combined with techniques such as Dynamic Voltage Frequency Scaling (DVFS) [22].

However, the increased utilization of Information and Communication Technologies (ICT) was facilitated by the optimizations made in recent years. The perception of users having unlimited access to resources led to a surge in their usage and resulted in a rebound effect and a backlash. Users often do not consider their resource consumption, leading to infrastructures being oversized to meet peak demands, even though the demand is significantly lower for the majority of the time. This results in a waste of resources that translates into an overconsumption of energy.

To tackle this issue of resource wastage, efforts have been made to find new techniques such as elastic management [1] to

help meeting the resource requirements, or consolidation [21] to improve energy efficiency. On one side, elasticity helps to reduce resource wastage by allocating them only when needed. On the other side, consolidation ensures that resources are allocated efficiently. This subject has been extensively studied by the community [17], with approaches varying in complexity. Despite their benefits, these two approaches can be limited because of technical and economic barriers in their implementation, the performance degradation that they can produce, or the complexity to maintain them.

The aim of this work is to revisit consolidation, using simple algorithms but activating other technical levers to extend the potential energy gains. By accurately evaluating and predicting the usage of VMs [12], a barrier to the implementation of elasticity and consolidation approaches can be removed. Proper evaluation of the usage of the machines or their applications makes it possible to condition the implementation of these techniques only to periods when the Virtual Machines (VMs) are idle. By ensuring that the application is available in a degraded mode during periods of inactivity, while maintaining the ability to react quickly to sudden or unforeseen usage, we can guarantee service continuity and performance levels that meet Service-Level Agreement (SLA) requirements.

In this paper, we examine a consolidation approach that considers the historical usage of a VM to assess the utilization of the application it hosts. Unused VMs are consolidated with oversubscription in order to decrease the number of physical servers needed for placement, while used VMs are consolidated onto servers to meet SLAs. In the long term, it will create a double benefit with i) the ability to reduce the DC energy consumption and ii) modify the capacity planning. With an efficient consolidation algorithm, the absolute number of PMs can be reduced and therefore the DC carbon footprint both on Scope 2 (energy consumption) and Scope 3 (upstream, purchases). It limits on one side the GHG emissions and on the other the need for rare earth in order to manufacture the servers [19].

The main contributions of our work are:

- a complete consolidation methodology containing VMs usage evaluation, the use of prediction models and the study of overcommitment mechanisms such as ballooning, swapping and transparent page sharing;
- a VM placement algorithm using consistent CPU and memory oversubscription ratios to reduce the energy consumption of a virtualization infrastructure;
- a detailed simulation process and a production deployment of the consolidation methodology (ORCA) on a Closed-Source hypervisor: VMWare vSphere;

Section II presents the state of the art and related works. Section III reminds key challenges of implementing consolidation on a virtualization infrastructure. Section IV depicts the proposed methodology and how it addresses the challenges. Simulation and experimental results that validate the relevance of the approach are presented in Section V. Finally, we add further discussions on this work in Section VI and conclude in Section VII.

II. Related Work

A. Virtualization and resource management

Through the possibility to run multiple workloads and OS on a single server, virtualization has played a crucial role during the last 20 years to improve the energy efficiency of cloud data centers [31, 32]. Though it appears that no hypervisor outperforms the others in terms of energy efficiency [30], several methods were used to in different hypervisors to improve this energy efficiency [3, 43]. Among these techniques, elasticity [23, 29], dynamic resource management [14, 36] and consolidation [2, 5, 6, 8, 27, 38] are essential. In [43], Waldspurger introduces different mechanisms to manage memory in VMWare ESXi, including *content-based page sharing* and *memory ballooning*, later implemented in other hypervisors. Memory ballooning is a technique used by a host under memory pressure to retrieve memory. It inflates a *balloon* of inactive memory on one or multiple VMs to be able to allocate it to other VMs. Liu *et al.* [36] compare it with memory hotplug and highlight the complementarity of the two techniques to effectively manage memory. Chashoo *et al.* [9] study the different techniques to reduce the cost of live VM migration. The paper shows that the use of ballooning allows quicker migration as the amount of memory allocated by the host is reduced when using ballooning. Additionally, Hines et Gopalan [25] show that the use of ballooning reduces the number of page faults when migrating a VM. In our work, we use ballooning to increase the number of VMs on a single PM, and we will leverage these performance gains during VM migration.

Further than memory, overcommitment or oversubscription can be performed on other resources. Jacquet *et al.* introduce SweetspotVM, a CPU oversubscription approach that considers per-vCPU performance rather than the conventional per-VM granularity. Their approach demonstrate that oversubscription can perform as well as a non-oversubscribed environment at the scale of an Infrastructure as a Service (IaaS) platform. CPU oversubscription is a feature that is now handled by many hypervisors [40, 42] and even recommended by constructors to densify infrastructures.

B. Consolidation and prediction

VM consolidation consists in optimizing the allocation of VMs on physical servers. One of its goals is to reduce the number of unused or idle PMs in an infrastructure by consolidating VMs onto fewer physical hosts. Many research has been conducted about consolidation [15, 20] as it was and still is a key challenge to reduce DCs energy consumption [13]. Most research about consolidation formalizes the problem in 3 parts: (1) detection of over or under-loaded hosts, (2) selection of VMs to migrate, (3) selection of destination host. Most of the work seeks to optimize one or multiple steps of the process.

The first step of the consolidation process consists in evaluating PM load status based on their resource usage and selecting PMs for which VM migration is needed. Migrating VMs from overloaded PMs will improve quality of service

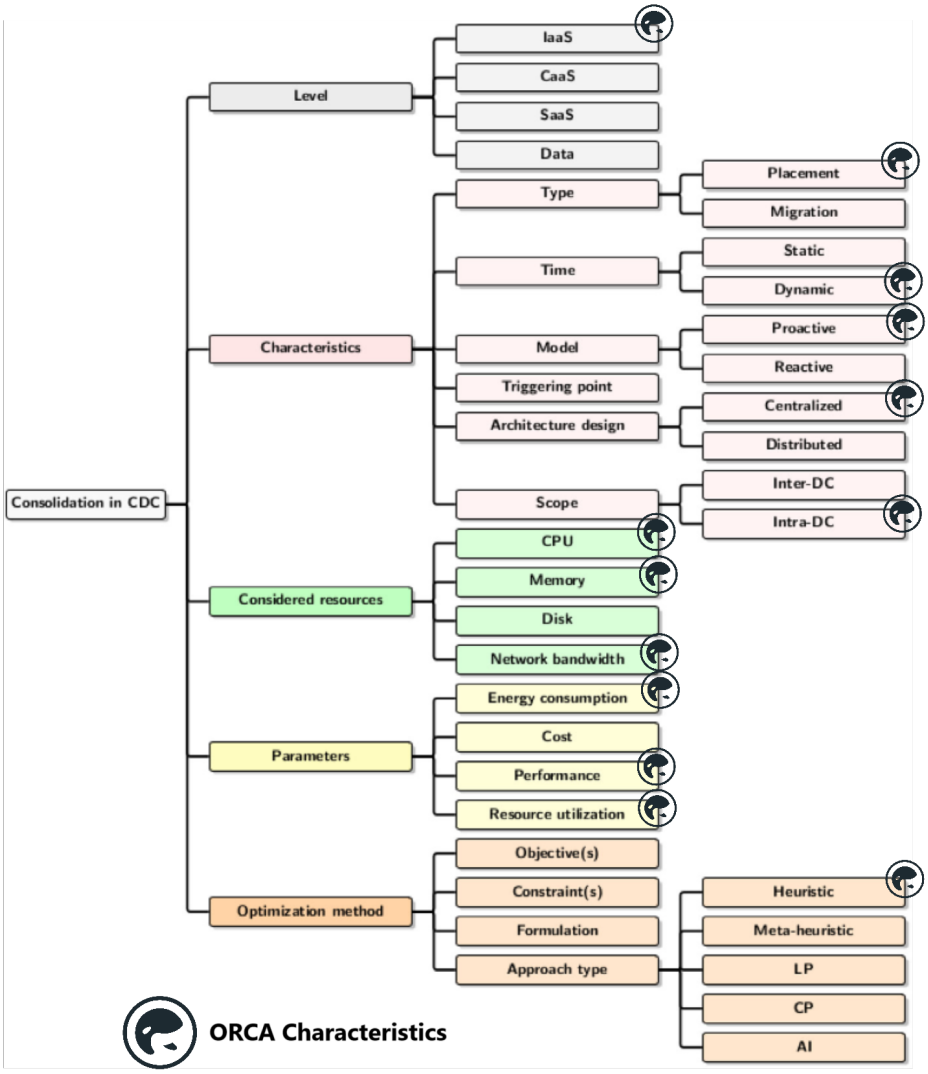


Fig. 1: ORCA methodology characteristics in Helali *et al.* taxonomy [21]

and limit SLA violations. On the other hand, when a PM is underloaded, all the VMs it hosts should be migrated to shut it down, thus enabling energy savings. This consolidation process step can be conducted using both static and dynamic thresholds. Beloglazov *et al.* [7] introduce a Markov chain model for the host overload detection problem in the context of VM consolidation while considering Quality of Service constraints. Minarolli *et al.* [39] uses long-term predictions of resource demand from VMs and performs time series forecasting using machine learning. The authors consider prediction uncertainty and VM live migration overhead which results in enhanced stability, decision-making, and overall performance. Some VMs on underloaded or overloaded hosts will then be selected for migration. Selecting the appropriate VMs to migrate is an important step to reduce the load on PMs, minimizing SLA violations risks and optimizing quality of service. Zeng *et al.* [46] discuss the importance of VM selection and VM placement in the consolidation problem. The authors propose a multiple resource Influence Coefficient (ICVMS) to identify VMs causing host overload in this VM selection process, resulting in the migration of VMs with the greatest impact. Chen *et al.* [10] select VMs to migrate using a combination of historical and future trends, called combined trend, of past used resources and future requested resources by PMs or VMs. VMs with the highest combined trend are selected to be migrated until overloaded hosts are stabilized. The last step of the consolidation process is VM placement, where the selection of destination PMs for the previously selected VMs occurs. This step of the consolidation is a non-deterministic polynomial time-hard (NP-hard) problem usually defined as a bin packing problem. Many different heuristics have been studied to handle this problem, such as classical heuristics like First-Fit decreasing (FFD), Best-Fit decreasing (BFD) or Worst-Fit decreasing (WFD) [17, 44]. Other approaches called meta-heuristics are also studied to solve the consolidation problem. They include genetic algorithms and

evolutionary approaches like Swarm intelligence algorithms such as Particle Swarm Optimization (PSO) [41] or Ant Colony Optimization (ACO) [16, 26], usually used in multi-objective consolidation approaches. Linear Programming (LP), Constraint Programming (CP), and the use of machine learning and deep learning are also possibilities to solve the consolidation problem. While some papers focus only on specific steps of the consolidation problem, others tackle multiple steps and include other dimensions and parameters in their studies, such as the consolidation level, the considered resources, or the evaluated parameters and methods. Considering the Helali *et al.* taxonomy [21], we identified the different characteristics of our approach in Fig. 1.

Beloglazov *et al.* [6] study both VM selection and placement with several VM selection approaches: Minimization of Migrations (MM), Highest Potential Growth (HPG) or Random Choice (RC) and a modification of Best Fit Decreasing (BFD) for VM placement. Bharanidharan et Jayalakshmi [8] propose a *Learning Automata* that uses the environment and previous actions to solve the VM placement problem. Using CloudSim and data from the CoMon project, they demonstrate that their approach provides better results in terms of number of migrations, number of SLA violations, and energy performance than [6]. Mashhadi Moghaddam *et al.* [38] highlights the relevance of using prediction for consolidation by training individual Machine Learning (ML) models to enhance the VM placement algorithms. These works use simulation and generic data to evaluate their results, while our work is based on a private cloud DC with heterogeneous applications and VMs. Hsieh *et al.* [27] forecast CPU utilization using a Markov model to minimize unnecessary VM migrations. Many papers focus on CPU utilization when tackling a consolidation problem as it has the greatest impact on PM power consumption. Our work uses both CPU utilization and network bandwidth to evaluate the idleness of a VM, as well as memory management mechanisms to increase the number of VMs per PM and migration performances. Hieu *et al.* [24] use a multi-usage prediction model based on CPU, memory and disk usage to improve consolidation. We do not consider disk usage in our work as the storage is provided on external storage arrays. Rather, we use network usage, considering that most applications or services rely on a network connection. Wood *et al.* [45] rely on a similar infrastructure with external storage and propose a black-box consolidation method based on CPU, memory and network bandwidth. However, they do not address the issue of energy consumption. Bacou *et al.* [2] propose Drowsy-DC : a management system that predicts an idleness probability for each VM of the infrastructure and places it on PMs accordingly. The idleness probability is calculated with CPU usage and they implement their solution on OpenStack modifying the scheduler. We propose an evaluation of the VM usage based on both CPU usage and network bandwidth and our solution acts as an external plugin because we cannot modify the VMWare scheduler source code.

III. Main challenges

This section aims to underline multiple challenges faced by data center operators to implement consolidation methodologies. Parallels are drawn to bring these challenges into the context of our study.

A. Infrastructures usage and consolidation potential

Consolidation may have one or multiple objectives when applied on a virtualization infrastructure. In our context, which aims to reduce the infrastructures energy consumption, the initial infrastructure resource usage plays a major role in the result potential. An initially low-used infrastructure will eventually result in higher power consumption reduction, as (1) the power consumption of PMs is not proportional to their load usage [4] and (2) there is more room for improvement if the consolidation aims to use PMs at a certain load.

The resource allocation rate on physical machines is also linked to the consolidation potential, as it influences the load usage on these same PMs. Resources oversubscription (overcommitment), both in terms of CPU and memory, can increase the overall infrastructure usage, and therefore save more energy when applying the consolidation. However, Cloud Service Providers are not keen to use resources oversubscription as it has a direct impact on performances and can result in QoS degradation. Finding accurate oversubscription ratios is then a key challenge to ensure a good quality of service while maximizing energy savings.

Individual PM load usage as well as clusters overall usage are therefore dependent on the resources allocation ratio used by data centers operators. But VM individual usage also impacts PMs or clusters load usage. A PM with an aggressive oversubscription ratio hosting solely idle VMs will theoretically itself be underused in terms of load, and therefore will be considered as not energy efficient. Assessing VM usage on infrastructure appears then as another challenge to quantify the consolidation potential on an infrastructure.

Evaluating all these indicators requires an automated, industrial data collection process that can be complex to set up. As the quality of the analysis is highly dependent on the granularity of the data collection and its retention time, implementing this may require the acquisition of specific hardware and software for the collection and storage of these large volumes of information. This represents extra cost for cloud operators and technical knowledge to set up and maintain such systems.

B. Proactive consolidation and prediction models

As VM consolidation is intended to improve the use made of computing resources, it can be used to balance the load between multiple PMs in a single virtualization cluster. This can be done by following the key steps generally used in consolidation approaches: (1) detection of over/under-used PMs, (2) selection of VMs to migrate and (3) selection of destination PMs. These consolidation steps can be applied based on measured values at time t to improve short-term PM

load usage. Conducting this kind of consolidation approach can be defined as *reactive* consolidation.

However, applying consolidation with oversubscription introduces another challenge. If the usage rate of VMs in an infrastructure varies over time, the use of overcommitment can actually lead to performance degradation on oversubscribed VMs. To mitigate these QoS degradations and potential SLA violations that may result in financial penalties, it appears essential to limit the use of overcommitment to underused or idle VMs and to anticipate these periods of inactivity. The use of prediction models is a good response to this challenge, but it also brings its own difficulties.

The use of prediction models require gathering metrics regarding VM usage over a certain period of time. This can represent extra costs in terms of storage for cloud providers depending on the amount of data to collect. Moreover, the training of prediction models may require data pre-processing, the study of different heuristics for the training as well as hyper-parameter tuning to enhance prediction models accuracy, as well as specific hardware for model training. Using prediction models therefore brings extra complexity to the consolidation approach but it also appears as a prerequisite in our context.

C. Consolidation algorithm and parameters

The consolidation problem, and especially the VM placement step, is a NP-complex problem which can therefore be handled with constraint and linear programming solvers. However, the use of such solvers, especially in the case of multiple objective bin packing, is complex and there is no guarantee of finding an optimal solution.

Our context reinforces the difficulties of using such a solver, as various parameters impact or can be impacted by them. The consolidation period, i.e. the frequency at which VM placement is refreshed and VMs are migrated, is crucial here. If too short, the solver may not be able to compile an optimal solution in the given time. This consolidation period also impacts the training of our prediction models, as they may require more data points considering that the sampling rate of our collected metrics will be shorter.

As mentioned, proper oversubscription ratios are also crucial to maximize the energy savings while maintaining optimal performance on the VMs. Combined with VM usage analysis, a combination of oversubscription ratio per VM usage can be studied to achieve the same goal.

D. Production deployment strategies

Implementing a production deployment of a consolidation algorithm or methodology is also challenging for DC operators. First, the multiple parameters discussed above must be carefully studied. Depending on the size, architecture, and criticality of the infrastructure, it is possible to design a system where a limited scope of VMs is migrated to reduce energy consumption. For example, one can choose to only migrate VMs that are considered as idle to ensure a good quality of service on the most used VMs. This will, however, limit the power consumption reduction obtained through consolidation.

Interfacing the algorithm with the infrastructure hypervisor can also be challenging. The developed consolidation algorithm may act as the orchestrator and be responsible for VM migration. The hypervisor orchestrator must therefore be disabled, but the consolidation solver will have to minimize the number of migrations to avoid additional power consumption due to VM migrations. VM migration can also be handled by the hypervisor orchestrator itself, using embedded features of the hypervisor for the consolidation implementation.

Fault tolerance must also be included or anticipated when deploying consolidation in production. Production infrastructures must be able to handle one or multiple PM crashes without service interruption, meaning that resources must be available to host VMs when a crash occurs on a PM. The fault tolerance rate is chosen by the infrastructure operator and depends on the cluster size and the SLAs. A high fault-tolerance rate will guarantee better service availability, but will also reduce the reduction in consumption achieved through consolidation.

All these implementation choices, combined with other discussed parameters, can represent extra costs for DC operators, both in terms of software and hardware.

IV. Methodology

This section presents the proposed consolidation methodology with a modeling of a virtualization cluster and the presentation of the targeted infrastructure. It also presents responses to the key consolidation challenges identified in Section III. The consolidation algorithm is presented as well as its implementation in simulation and experiment.

A. Design of the system

In the current context, as for in many virtualization infrastructures, resources can be divided into various clusters. This allows to meet the challenges of segmentation, license management or resource management.

TABLE I enables us to model a system corresponding to an individual virtualization cluster. It contains several PMs and VMs, with their own configurations. The system also includes variables to help solving the consolidation problem: memory related values as well as 2 matrices containing measured or predicted usage of the VMs of the cluster.

Many equations can model the memory consumption of a VM or PM, and its potential impact on an application performances. We will focus here on memory overcommitment, both at PM and cluster scale:

$$O(t) = \frac{\sum_{j=0}^{|V|} v_j^m(t)}{\sum_{i=0}^{|P|} p_i^m}, O_{p_i}(t) = \frac{\sum_{j=0}^{M_{p_i}} v_j^m(t)}{p_i^m}, with\ v_0^m = 0 \quad (1)$$

Memory overcommitment is here defined as the ratio between the configured memory on VMs within a cluster/host and the available physical memory on that cluster/host. By definition, *memory is overcommitted when the combined working*

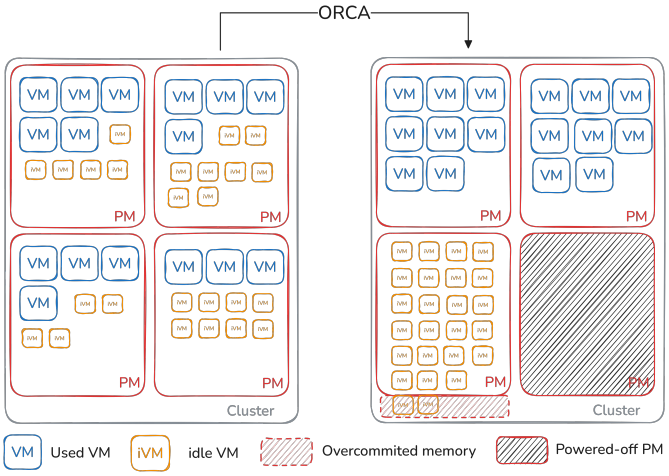


Fig. 2: Example of virtualization cluster before and after the execution of ORCA$_S$

TABLE I: Variables - Elements of a cluster

Variable	Description
P	Set of PMs
V	Set of VMs
p_i^V	Number of VMs on PM p_i
p_i^c	Number of cores on PM p_i
p_i^m	Amount of physical RAM on PM p_i
v_j^c	Number of vCPUs on VM v_j
v_j^m	Configured RAM on VM v_j
$O(t)$	Memory *over*commitment on the system
$O_{p_i}(t)$	Memory *over*commitment on PM p_i
$O_{v_j}(t)$	Memory *over*commitment of VM v_j
U	VMs usage matrix
F	Forecasting matrix

memory footprint of all VMs exceeds the size of the host memory, i.e. the ratio is greater than 1. The term overcommitment is used here regardless it is less than or greater than 1 for simplicity.

Our main objective is to increase the value of $O(t)$, as it correlates with a cluster's energy consumption. The higher the value, the fewer PMs there will be for a given number of VMs. The fewer PMs in a cluster, the lower its energy consumption. Increasing $O(t)$ can be done by decreasing the number of PMs ($|P|$) while keeping the number of VMs ($|V|$) constant, or conversely by increasing $|V|$ with constant $|P|$.

However, these solutions do not consider side effects. Increasing $O(t)$ through these mechanisms will cause performance degradation across the entire cluster. Therefore, the algorithm will split P into multiple PM subsets with a specific overcommitment value. Each VM will be allocated an overcommitment rate $O_{v_j}(t)$ according to its usage, given by Algorithm 1 for each sample t. Given an initial overcommitment O_s, VM and PM groups will be constructed such that, on an iteration t, $O(t) \geq O_s$.

Fig. 2 shows an example of a virtualization cluster with PMs and VMs on which ORCA is applied. VMs considered as used (blue) are migrated together onto 2 PMs, and idle VMs (orange) are consolidated with memory overcommitment onto a single PM. This enables reducing the number of used PMs to 3, increasing the value of $O(t)$ in this cluster.

B. Infrastructure

There are today many data centers types, with their own characteristics that can help classify them. Their size, location and the services they offer can vary a lot, and the implementation of a consolidation methodology can be impacted by these factors. For example, a data center offering services based on virtualization will logically use a hypervisor, which will inevitably influence the implementation of consolidation algorithms.

This study is conducted within a data center called *SynAApS*, operated by *Ciril GROUP*. The infrastructure contains multiple virtualization clusters for both internal and customer needs. The simulations and experiments are realized on the internal virtualization cluster composed of 600 VMs with 400 daily active VMs and 7 heterogeneous PMs. This virtualization cluster runs on the VMWare vSphere hypervisor. VM storage is handled by a centralized storage array, which has an impact on consolidation: when VMs are migrated, only the RAM is transferred from one PM to another. This drastically minimizes migration times to the order of milliseconds or seconds.

Metrics to evaluate the usage and load of the infrastructure VMs and PMs such as CPU, memory and network usage are collected. All the metrics are gathered using a Telegraf[1] agent using the vSphere input plugin. PM and VM usage related metrics are gathered every 30 seconds and stored in a VictoriaMetrics[2] database with a retention period of 1 year. These collection parameters enable a good usage analysis, as specified in Section III-A.

C. VM Usage and consolidation potential

Our methodology will rely on memory overcommitment to enhance power consumption reduction. However, this cannot be done arbitrarily on all the VMs of the infrastructure as it may cause performance degradation. Therefore, we need to evaluate VM usage to identify under-utilized or *idle* VMs on which the use of memory overcommitment is not considered an issue, as explained in Section III-A.

We can for example evaluate the use or not of a standard web application VM with different conditions:

1) The application must not be consulted at a given time.
2) There must be no current queries to the database.
3) There must be no background tasks running on the VM (cronjobs, backups, etc).

These criteria can be monitored by observing i) the communications at the network interfaces of the machine, and ii) the CPU utilization rate of the VM. Application metrics and access to the VM OS are not mandatory a priori, as these metrics

[1]https://www.influxdata.com/time-series-platform/telegraf/
[2]https://victoriametrics.com/

are provided by the hypervisor. Algorithm 1 is proposed to evaluate whether or not a VM is used.

Algorithm 1 VM Usage evaluation algorithm

function IS_USED($vm, t, cpuRate = 1\%, netRate = 0$)
  if cpu_usage(vm, t) $\leq$ $cpuRate$ **and** net_usage(vm, t) $< netRate$ **then**
    return 0
  else
    return 1
  end if
end function

With this algorithm, the state of an idle VM will return 0 and a used VM will return 1. From this, a matrix containing the evaluation of our VM usage with a sampling interval of time t can be constructed. This allows to calculate the mean usage of each VM over a certain period T and the mean usage of all VMs for a given sample t. It is then possible to evaluate the mean usage of one or multiple VMs over a given period of time. The considered thresholds make the evaluation algorithm strict but applicable to many other servers. Considering a 3-tier web application architecture, all the elements usage can be evaluated using Algorithm 1. This algorithm can be extended to other CPU-intensive applications like HPC or network-intensive like video conferences. It does not require prior knowledge of the application hosted on the VM a priori.

D. Memory overcommitment mechanisms and allocation ratios

As the methodology uses memory overcommitment, selecting proper allocation ratios is crucial to find the correct balance between energy savings and quality of service on the VMs. This is a key challenge when it comes to enhance infrastructure usage as explained in Section III-A. This study does not focus on CPU overcommitment as it is already used in all the *SynAApS* clusters. TABLE II gives the configuration limits on VMWare vSphere[3]. This sets a theoretical maximum value of 32 vCPUs per core.

TABLE II: Limits - VMWare vSphere

Limitation	Value
Number of VMs	1024
Number of vCPUs	4096
Number of vCPUS per core	32
Amount of memory	24 TB

On the memory side, multiple techniques are used to handle memory overcommitment. When a PM does not have enough memory to respond to the demand of one or more VMs, it can retrieve memory using different mechanisms, including ballooning, swapping and Transparent Page Sharing (TPS).

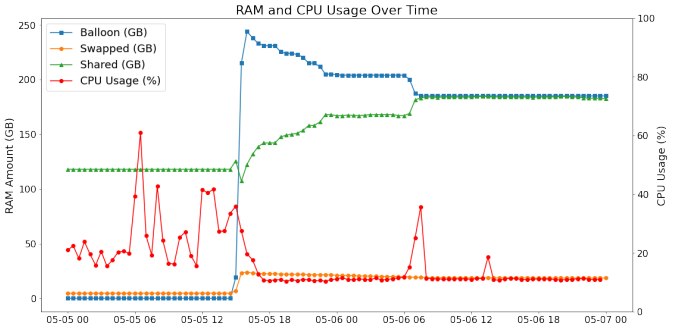
Ballooning happens when a PM has a RAM shortage and it asks VMs, through a specific driver named the memory controller, to free inactive or unused memory to provide it to other VMs.

Swapping occurs in the same context, but this time the PM will write the memory pages allocated to the VMs on an external storage (i.e. its physical disks or a storage controller).

TPS allows the host to scan the memory of the running VMs and identify identical memory pages. For each similar memory page on a specific VM, TPS creates a single copy and shares it with the others. As it may be a security concern, inter-VM TPS is disabled by default and this mechanism occurs within individual VMs.

[3]https://configmax.broadcom.com/guest?vmwareproduct=vSphere&release=vSphere%207.0&categories=2-0

Fig. 3: Measured ballooned, swapped, shared memory and CPU usage on the PM



Both ballooning and swapping enable dealing with erratic memory shortage, but they may create performance degradation on the VMs. TPS runs continuously on the hypervisor and gradually releases memory on PMs.

To identify a consistent memory oversubscription ratio, we select a set of idle VMs and manually migrate duplicates of these VMs onto a single PM. We collect CPU and memory usage metrics of the physical machine. TABLE III shows the gathered metrics of the experiment.

TABLE III: RAM and CPUs observed metrics on a PM after VMs manual consolidation

	Physical RAM PM (GB)	RAM Usage PM (GB)	Sum of VMs configured RAM (GB)	Sum of VMs configured CPUs
Total	384	322.79	1168	**382**

After the migration of 50 VMs to the PM, the assignable number of vCPU per PM core limit is reached. This physical constraint enables us to evaluate an upper bound for memory overcommitment O_{max}, where $O_{max} = \frac{1168}{384} = 3.04$, given the VMs configuration. The high vCPU per PM core ratio is not considered an issue here, given that the VMs consume less than 1% of their CPU capacity according to the usage evaluation algorithm.

Fig. 3 shows the amount of RAM retrieved by the ballooning, swapping and TPS mechanisms, which are a direct response to the memory shortage observed on the server. The

mean CPU usage is also displayed, and the figure shows that the high memory overcommitment does not lead to CPU overconsumption. The CPU consumption of the PM even stabilizes around 15% with solely idle VMs.

E. Prediction Models

To respond to the challenge highlighted in Section III-B, our methodology uses prediction models to conduct our experimentation and deploy it in production on the *SynAApS* clusters.

When forecasting a timeseries, different approaches are considered in the literature: linear regression, ARIMA, SARIMA or deep learning methods such as Long Short-Term Memory (LSTM) Neural Networks (NN) [34, 47]. Logistic regression can also be used to forecast binary timeseries. To assess the simplicity of forecasting a timeseries, an Augmented Dickey-Fuller (ADF) test can be conducted to determine if it is stationary. A stationary timeseries is easier to forecast, as its statistical metrics (mean, variance or covariance) do not change over time. Linear regression is often used to forecast stationary timeseries, whereas LSTM models provide better results with non-stationary timeseries. The heterogeneity of timeseries and VM usage also plays a role in the methodology used: model training and update times can be a key parameter. The amount of resources required to train and store models can be identified as a bottleneck for Cloud operators as identified in Section III-B. A compromise must therefore be found between performance, resource usage, training and inference times.

An ADF test is performed on the VM usage timeseries to assess their stationary condition. Among all the evaluated timeseries, 94.7 % are stationary. Since every VM has different usage data over a year, one model will be trained per VM for better results. In order to reduce training time and the size of prediction models, and given the large number of VMs for which the time series is stationary, machine learning appears as the best method. The prediction models accuracy will also be improved using gradient-boosting decision trees with the XGBoost library[4]. XGBoost enables the use similar objective functions as regular regression, while providing the ability to take advantage of specific features of the timeseries which may require prior knowledge.

The predicted timeseries will ultimately contain the mean usage of each VM per 15-minute period over one week. This 15-minute sampling rate was chosen to both preserve the variation in VM usage and to limit the consolidation algorithm execution frequency. The timeseries values are by definition always positive and smaller than 1. The evaluation metric used to assess the performance of our models is the Mean Absolute Error (MAE), given by Equation (2), where y_i is the prediction and x_i is the true value.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n} \tag{2}$$

[4]https://xgboost.readthedocs.io/en/stable

TABLE IV: Features and average MAE of the models during training

	Features	**MAE**
1	Day of year, hour	0.160
2	*Features 1* + Day of week, month, year, weekend	0.062
3	*Features 2* + Previous 12 hours + Value 24 hours before	0.032

Multiple model trainings were carried out to evaluate the relevance of using varying features, as well as their impact on the models performance. Different feature combinations were evaluated, as depicted in TABLE IV. The sole use of the day of the year and the timestamp hour results in a MAE of 0.16. This value is reduced when adding the corresponding day of the week, month of the year, year and an indication of whether the timestamp falls on a weekend. One can further reduce the MAE to 0.032, i.e. a 3% average difference over all the prediction by using lag features, which are previous values in the timeseries. Using lag features is however only possible when training the models, as it makes inference recursive and therefore more complex. Therefore, temporal features are used and offer an average prediction error of 6.2%.

To obtain consistent consolidation results, the prediction models do not need to forecast the exact usage of the VM over the next 15 minutes but rather a value that places the VM in the correct consolidation group. To evaluate the performance of the prediction models, we compare the result of the group allocation between the measured usage for week n and the predicted values for the same week n, using the measured data up to $n-1$. It is also possible to individually measure VM allocations within their group. Fig. 4 shows the distribution of the average allocation error rate per VM over one week. For 294 VMs, the allocation error rate is between 0 and 10%, meaning they are placed in the correct consolidation group more than 90% of the time. The overall mean error is 12.9% over one week.

From this, it is possible to create a filter for which VMs whose average allocation error exceeds a threshold are not considered in the consolidation. As mentioned above, no VM is literally removed from the consolidation: unconsidered VMs are always consolidated in the minimal memory overcommitment group. This will ultimately minimize the error rate of the consolidation, as well as the potential performance degradation on one side and the energy savings on the other side. The relevance of the threshold use can be verified by a simple calculation. The solving algorithm provided in Algorithm 2 being deterministic, the resulting consolidation for a set of identical consolidation groups will always be similar. Using the threshold, it is possible to calculate the divergence between consolidation groups at a given value. For a threshold set to 0.1, and using the number of VMs given by Fig. 4, the maximum allocation error rate will be set to 6.8%:

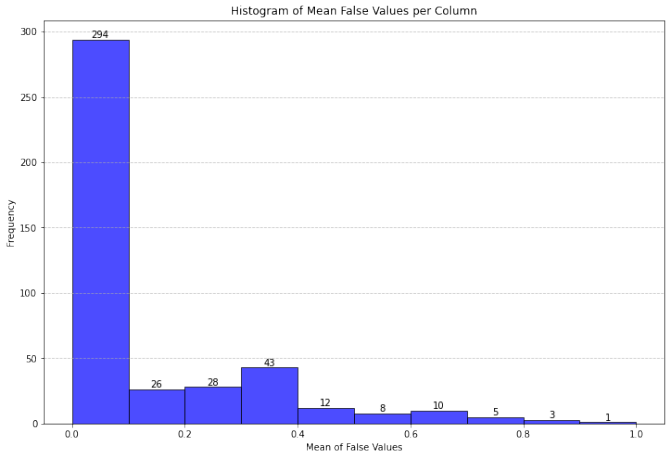


Fig. 4: Histogram of mean false allocation

$$\frac{(294 \times 0.1) + (136 \times 0)}{430} = 0.068 \tag{3}$$

With a threshold set to 0, there will be no consolidation difference between the U and F matrices, as the consolidation groups will always be similar. This value of threshold enables us to calculate the minimal average power consumption reduction, also called the reduction baseline in the rest of the article. This reduction baseline is used to quantify the benefits in terms of consumption that can be achieved solely through the use of consolidation with a new base memory allocation rate, without the use of overcommitment.

F. Consolidation algorithm

To solve the consolidation problem from an algorithmic perspective, two extra variables must be defined: *VMGroups* and *PMGroups* which respectively contain the VM groups and the associated PM groups. A single VM group is defined as follows:

$$< vms, S_v^m(t), S_v^c(t), O_v >$$

with vms the subset of VMs in the group, $S_v^m(t)$ the sum of the configured RAM of vms, $S_v^c(t)$ the sum of the configured vCPUs of vms at time t, and O_v the associated overcommitment value. A PM group is defined as <pms, $S_p^m(t)$, $S_p^c(t)$, O_p>, where pms is the PM subset of the group, $S_p^m(t)$ the amount of memory on all the PMs of the group, $S_p^c(t)$ the amount of cores on all PMs and O_p the associated overcommitment value. VM and PM groups are associated when $O_v = O_p$.

Algorithm 2 presents ORCA$_S$, the proposed solver for the consolidation placement problem. It first sets the PMGroups as an empty dictionary, which will ultimately contain the consolidated PM groups with their sets of *pms*. The VM groups as well as the set of PMs P according to a specific policy. VM groups can be sorted in multiple ways such as by number of VMs ($|vms|$), total size of RAM ($S_v^m(t)$) or overcommitment value (O_v). PMs can also be sorted with

Algorithm 2 Solving Algorithm - ORCA$_S$ (Part.1)

```
1: Input: VMGroups, P, groupMigration
2: Output: groupedPMs
3: procedure ORCA_S(VMGroups, P, groupMigration)
4:     Initialize PMGroups as an empty dictionary
5:     Sort VMGroups according to the proper policy
6:     Sort P according to the proper policy
7:     for each groupname, VMGroup in VMGroups do
8:         Get availablePMs from P
9:         if VMGroup['vms'] is empty then
10:            for each pm in availablePMs do
11:                if pm tag is groupname then
12:                    Untag pm
13:                end if
14:            end for
15:            groupedPMs['pms'] ← []
16:            Skip to next iteration
17:        end if
18:        if vmBuffer is not empty then
19:            VMGroup['vms'] ← vmBuffer['vms]
20:        end if
21:        Get S_v^m(t) and S_v^c(t)
22:        NumRam ← CalcPmRam(S_v^m(t), O_v, availablePMs)
23:        NumCpu ← CalcPmCpu(S_v^c(t), availablePMs)
24:        numPms←max(NumRam, NumCpu)
25:        if numPms == 0 then
26:            vmBuffer ← VMGroup
27:            for each pm in availablePMs do
28:                if pm tag is groupname then
29:                    Untag pm
30:                end if
31:            end for
32:            groupedPMs['pms'] ← []
33:            Skip to next iteration
34:        end if
35:        groupedPMs['pms'] ← availablePMs[:numPms]
36:        for each pm in availablePMs do
37:            if pm ∈ availablePMs[:numPms] then
38:                Tag pm with groupname
39:            else
40:                Untag pm
41:            end if
42:        end for
43:    end for
44:    if groupMigration is True then
45:        MigrateGroups(VMGroups, PMGroups)
46:    end if
47: end procedure
```

different possibilities like the physical amount of RAM (p_i^m) or the number of cores (p_i^c).

For both groups, items can be sorted in either increasing or decreasing order. Different sorting of VM groups or PMs can result in different consolidation, with varying consumption reduction or error rate. The algorithm then loops through all the considered consolidation group and processes different steps. First, the $availablePMs$ is retrieved from the set P (line 6) : this corresponds to PMs already used for this consolidation group during the previous iteration (tagged with the group name) or PMs not used at previous iteration. This ultimately allows us to reduce the number of migrations provoked by the consolidation. Through lines 7 to 14: if the consolidation group contains no VMs, it does not need to be processed. PMs tagged with the corresponding group name will be freed, and therefore usable to consolidate other groups of VMs. For uniformity, the PMGroup dictionary is populated with an empty set of PMs. From line 16, a buffer mechanism is also implemented to handle potential PM shortage when consolidating a group of VMs. If the buffer is not empty, buffered VMs are added to consolidated VMs. At line 19, the amount of CPU and RAM

is then retrieved, and the required number of PMs is calculated with two functions. One function calculates the number of PMs based on the memory requirements (line 20), as in Equation (4) (on page 10). The other function calculates this number based on CPU requirements (line 21), taking into account the limits imposed by vSphere. Equation (5) on page 10 details this, with X being the average number of cores per PM here. As different PM configurations are used, the two functions iterate through the list of servers rather than using an average value to provide a better result. The final number of PMs selected is the maximum value obtained from these two functions.

Lines 23 to 32: if no PM is available to host VMs of the group, VMs are placed in the buffer, the PM set is populated with an empty list and the next VM group is consolidated. Otherwise, from line 33, the PM set is populated with the predefined available PMs and the number of PMs calculated before. For every used PM, the groupname is tagged (lines 35, 36). Remaining PMs are untagged because not used (line 38).

A final function is then conditionally run using the groupMigration parameter (lines 42-44), causing the migration of consolidation groups. Consolidation groups are sorted by decreasing order of overcommitment value, i.e. quality of service. The function loops through the first $n-1$ consolidation groups. For each processed consolidation group, the measured memory overcommitment rate is calculated and compared to the theoretical overcommitment rate of the next consolidation group. If it is lower than this theoretical overcommitment rate, VMs and PMs can be migrated to the next consolidation group without compromising the quality of service. This standardizes the consolidation at the macro level and can even enhance the quality of service.

$$|pms| = \left\lceil \frac{S_v^m(t)}{O_v \times Y} \right\rceil \tag{4}$$

$$|pms| = \frac{S_v^c}{X \times 32} \tag{5}$$

G. Energy savings and Error rate calculation

This section presents the result computation process, both in terms of energy savings and performance degradation.

1) Energy savings: The energy savings obtained thanks to the consolidation methodology are calculated based on measured power consumption of the evaluated cluster PMs. The PM power consumption is retrieved by the hypervisor from internal sensors embedded in the power supply through the Intelligent Platform Management Interface (IPMI) driver.

For simulations, the power consumption reduction is computed by comparing the initial infrastructure power consumption over the simulation period and the sum of the power consumption of every used PM over the simulation period. For the experimentation, the consolidated power consumption is retrieved by measuring the power consumption of all the PMs hosting at least one VM. During the experimentation, the unused PMs are placed in maintenance mode and no VM is able to migrate to theses PMs.

2) Error rate calculation:

Algorithm 3 Solving Algorithm - ORCA$_S$ (Part.2)

```
1: Input: amountCpus, PMS
2: Output: numPms
3: procedure CalcPmCpu(amountCpus, PMS)
4:     numPms ← 0
5:     remainingCpus ← amountCpus
6:     for each p in PMS do
7:         if remainingCpus ≤ 0 then
8:             break
9:         end if
10:        numPms ← numPms +1
11:        remainingCpus ← remainingCpus −32 × p^c
12:    end for
13:    numPms ← min(numPms, |PMS|)
14: end procedure

15: Input: amountRam, aimedOc, PMS
16: Output: numPms
17: procedure CalcPmRam(amountRam, aimedOc, PMS)
18:    numPms ← 0
19:    remainingMem ← amountRam / aimedOc
20:    for each p in PMS do
21:        if remainingMem ≤ 0 then
22:            break
23:        end if
24:        numPms ← numPms+1
25:        remainingMem ← remainingMem −p^m
26:    end for
27:    numPms ← min(numPms, |PMS|)
28: end procedure

29: Input: VMGroups, PMGroups
30: Output: VMGroups, PMGroups
31: procedure migrateGroups(VMGroups, PMGroups)
32:    Sort consolidation groups by decreasing order of overcommitment
33:    for each groupname, VMGroup in VMGroups[:-1] do
34:        Get corresponding PMGroup
35:        Get consolidation index of VMGroup i
36:        Calculate O^i = S_v^m(t) / S_p^m(t)
37:        Get O_v^{i+1}
38:        if O^i ≤ O_v^{i+1} then
39:            Migrate vms to VMGroup i + 1
40:            Migrate pms to PMGroup i + 1
41:        end if
42:    end for
43: end procedure
```

a) Macro scale: Cluster.: Despite the black-box nature of the proposed methodology, the latter must still take into account undesirable performance degradation on VMs. When uncontrolled, these degradations in QoS can lead to SLAs violations and penalties for cloud service providers.

It is assumed that the initial overcommitment rate O_s measured in a cluster does not lead to any QoS degradation. However, the proposed methodology expressly aims to build consolidation groups for which the overcommitment rate is higher than O_s, which by definition means undersized consolidation groups. It is therefore necessary here to separate voluntary performance degradation (applied to barely used or unused VMs) from undesirable, or uncontrolled, performance degradation. The methodology, based on VM usage evaluation and the creation of consolidation groups adapted to different VM usage rates, does not cause undesirable performance degradation. In the following, we therefore only refer to performance degradation when these are not anticipated.

To assess performance degradation, and potential errors, a comparison must be made between the consolidation applied to measured and predicted data. Thus, a consolidation error

occurs if one these two conditions is verified at an iteration t:

- The number of PMs of a group is lower for predicted data than for measured data.
- The total amount of RAM of the PMs of a group is lower for predicted data than for measured data.

As specified above, a consolidation error can result in performance degradation at VMs scale. The overcommitment value of a specific group must however be considered when calculating these performance degradation. Considering two consolidation groups, $G1$ and $G2$, poor predictive model performance can impact the number of VMs in each group. If 70% of VMs considered in group $G2$ by the measured data are positioned in group $G1$ with the predicted data, the consolidation is likely to result in an undersizing of group $G2$. However, if the overcommitment rate of $G1$ is lower than that of $G2$, these same VMs will be assigned a lower overcommitment rate, and will therefore benefit from better performance. This issue can be addressed by weighting each group by its overcommitment rate, and calculating an overall consolidation score $Q(t)$ and a similar score based on memory $Q_m(t)$.

$$Q(t) = \sum_{j=0}^{|PMGroups|} |pms| \times O_p \tag{6}$$

$$Q_m(t) = \sum_{j=0}^{|PMGroups|} |S_p^m| \times O_j \tag{7}$$

With this approach, the methodology allows to quantify the performance degradation considering QoS at the macro scale. For both measured and predicted values, the consolidation score $Q(t)$ or memory consolidation score $Q_m(t)$ can be calculated at each iteration. An undersizing occurs if $Q(t)$ is lower for predicted values than for measured values. Memory undersizing occurs if the same condition is verified for $Q_m(t)$.

b) Micro scale: VMs.: To assess the performance degradation more in depth, it is also possible to calculate it at the scale of each VM. Considering the usage evaluation algorithm to be robust, and by carefully allocating the overcommitment rate of each group according to the associated usage, one can guarantee a good QoS when applying the consolidation on the U matrix, i.e. the measured data. At iteration t, after consolidation on matrices U and F, comparison of the theoretical overcommitment rate applied using the U matrix ($O_{v_j}^U(t)$ for VM j) and the overcommitment rate measured using the F matrix ($O_{v_j}^F(t)$) gives an indicator of whether or not performance has deteriorated for each VM.

The difference between $O_{v_j}^F(t)$ and $O_{v_j}^U(t)$ indicates an individual performance degradation at time t when $O_{v_j}^F(t) > O_{v_j}^U(t)$. Considering an evenly distributed degradation across all VMs of a consolidation group, this provides the memory stress percentage of VM j at time t. When computing this indicator for each VM at each iteration during the whole simulation, it is possible to quantify the overall performance degradation rate at micro-scale. To test different scenarios, the methodology takes into account different tolerance rates (5, 10, 15, 30 and 50%) when calculating the overall rate of performance degradation. A higher tolerance rate will result in a lower overall rate of performance degradation, as a degradation will only be considered if the difference between $O_{v_j}^F(t)$ and $O_{v_j}^U(t)$ exceeds this tolerance rate.

H. Algorithm implementation

1) Simulation process: Fig. 5 shows the process used in simulations to assess both power consumption reduction and potential performance degradation over periods of one week. First the PMs and powered-on VMs are gathered to set the scope of the simulation. Then, measured data is extracted from the usage matrix U for the specific simulation period T, and the forecasting matrix F is computed with the prediction models, using the same period T and sampling rate. 10 threshold values, introduced in Section IV-E, are then tested: from 0.1 to 1 by 0.1 steps. Using both matrices U and F, the error allocation rate is calculated for each VM, and the consolidation process can be applied using every predefined threshold. For each threshold, we apply the consolidation for each timestamp t of the consolidation period T. For each iteration period t, VMs are placed into their consolidation group and Algorithm 2 is executed. The average power consumption reduction as well as the performance degradation indicators are calculated after the execution for the last iteration. Note that Fig. 5 describes the simulation process using the F matrix. To compute performance degradation $Q(t)$ or $Q_m(t)$, the consolidation process must be computed both on F and U matrices simultaneously.

Algorithm 4 ORCA$_C$: ORCA Controller

```
1: Input: U
2: U,V ← vCenter inventory
3: while True do
4:     F ← createForecastMatrix(U, V)
5:     t ← F.index[0]
6:     while t != F.index[-1] do
7:         VMGroups ← evaluateUsage(V, F, t)
8:         PMGroups ← ORCA_S(VMGroups, P)
9:         Buffers ← {VMGroups, PMGroups}
10:        Apply consolidation
11:        st ← t
12:        for i = 0; i < 4; i = i + 1 do
13:            Increase st by 15 min
14:            VMGroupsSim ← evaluateUsage(V, F, st)
15:            if i is 0 then
16:                P_on ← ORCA_S(VMGroupsSim, P)
17:            end if
18:            P_off ← P_off + ORCA_S(VMGroupsSim, P)
19:        end for
20:        Shutdown PMs not in P_off
21:        Power on PMs in P_on
22:        Wait 15 min
23:        Increase t by 15 min
24:    end while
25:    U = updateUsageMatrix()
26: end while
```

2) Experimental setup: To assess the feasibility of the proposed solution, we need to develop and implement it on the experimental infrastructure. To implement the consolidation on the hypervisor, we use embedded features of VMWare vSphere. As we cannot modify the source code of the hypervisor, we propose an implementation using an additional

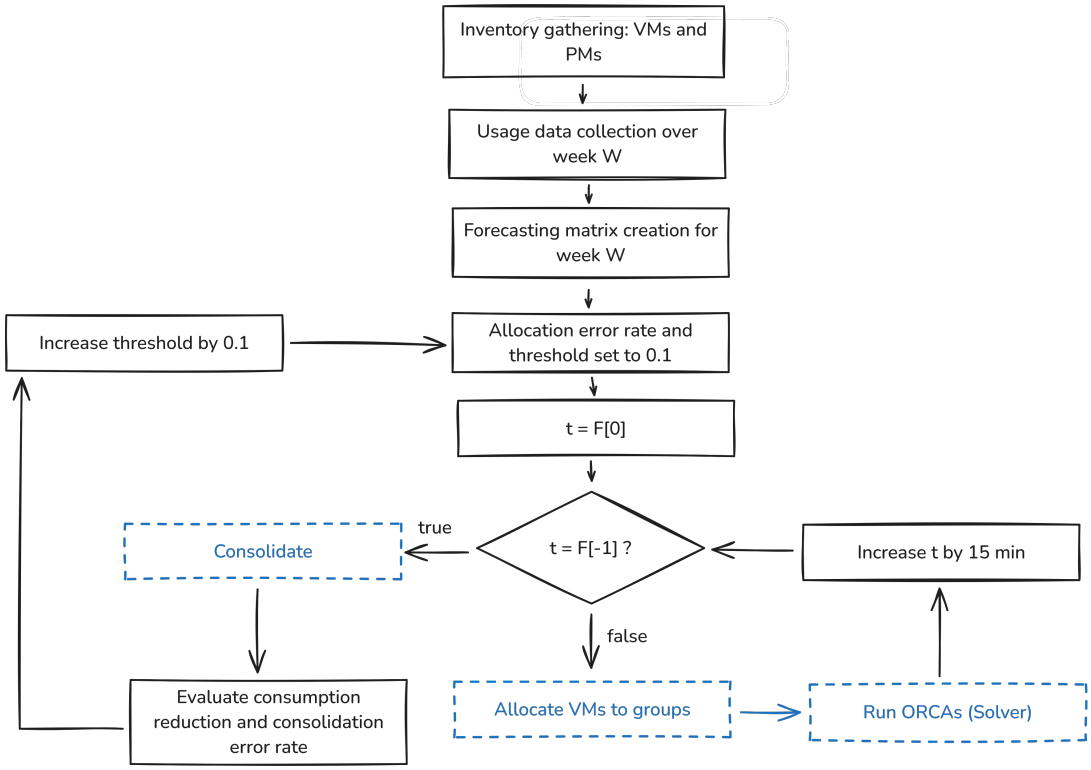


Fig. 5: Simulation process

layer based on PowerCli[5]. The consolidation relies on a feature called *VM Host Affinity rules*, which enables us to map groups of VMs to groups of PMs. Each rule takes for parameter a VM group, a PM group and one of the following *specification*[6] :

- *Must run on hosts in group*. VMs in VM Group 1 must run on PMs in PM Group 1.
- *Should run on hosts in group*. VMs in VM Group 1 should, but are not required, to run on PMs in PM Group 1.
- *Must not run on hosts in group*. VMs in VM Group 1 must never run on PMs in PM Group 1.
- *Should not run on hosts in group*. VMs in VM Group 1 should not, but might, run on PM in PM Group 1.

Using this feature presents multiple advantages. First of all, this greatly simplifies VM migration management. By using the Distributed Resource Scheduler (DRS) rules, we transfer the VM migration operations to hypervisor scheduler, assuming it is already configured to limit the number of migrations. We use the *Should Run* specification for our rules, enabling us to be reactive to potential breakdowns on our PMs. This specification level allows us to achieve high allocation rates while remaining flexible in VM management. If a VM experiences significant performance degradation visible from the hypervisor, the scheduler can place it on another PM.

We propose a simple software implementation representing the ORCA Controller, showed in Fig. 6. ORCA$_C$ is a Python implementation of Algorithm 4, which takes as input the Usage matrix. It gets the list of VMs and PMs of the cluster from the vCenter on line 1. From this, it creates the Forecasting matrix over one week, sampled every 15 minutes, using the prediction models previously trained. The consolidation process starts at the first index of matrix F. At each iteration, the VM consolidation groups are populated using the data in F. The solver depicted in Algorithm 2 then computes the corresponding PM groups. Once done, the implementation populates the text buffers with the consolidation groups (line 9). A PowerShell script using PowerCli creates or updates the vCenter VM and Host groups, as well as the VM Host Affinity Rules to apply the consolidation in the vCenter (line 10). After applying the consolidation, the algorithm will simulate the next four iterations (lines 11 to 19) to determine PMs to shutdown. In the current context, this corresponds to PMs unused for the following hour. The first simulated iteration also enables us to determine which PMs to power on. ORCA$_C$ then handles PM power on lines 20 and 21. This process is executed every 15 minutes.

A production implementation of the methodology needs to take into account fault tolerance to ensure continuous service on every VM. A fault tolerance rate can therefore be implemented, which defines the number of servers that

[5]https://developer.broadcom.com/powercli

[6]https://techdocs.broadcom.com/us/en/vmware-cis/vsphere/vsphere/7-0/vsphere-resource-management-7-0/using-drs-clusters-to-manage-resources/using-affinity-rules-with-vsphere-drs/vm-host-affinity-rules-with-vsphere-drs.html

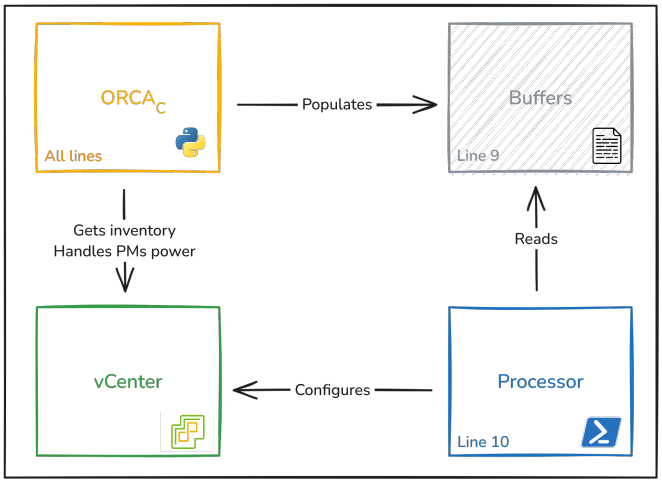


Fig. 6: Software implementation of the ORCA Controller

can fail without compromising service continuity. As there are heterogeneous server configurations in the experimental infrastructure, this fault tolerance rate is set to $2N$, N being the configuration of the biggest server in the infrastructure. At iteration t, the amount of resource available on all servers except the biggest must be greater than or equal to $2N$. This will ultimately reduce the power consumption benefits of consolidation, as some servers initially unused for a specific iteration could end up being used to ensure fault tolerance. In the implementation, PMs in this case are consolidated into the highest QoS consolidation group. Our methodology here addresses the multiple challenges raised in Section III-D. Considering the size and architecture of the infrastructure, we choose to migrate all the VMs in our consolidation strategy. VM migration cost is not considered an issue here, as we use centralized storage arrays. The VM migration process is handled by the hypervisor orchestrator using VM/Host Affinity rules. To prevent a high number of migrations, our solver keeps track of the PMs used in the different consolidation groups at each consolidation step. Finally, fault tolerance is considered to ensure service continuity.

V. Results

This section presents the results obtained thanks to our consolidation methodology. The complete methodology is tested in simulation based on measured and predicted values in the infrastructure, and compared to an optimal Bin Packing Solver. We also test the methodology using the same simulation process on another bigger cluster to evaluate its scalability. We finally conduct a production deployment in the infrastructure to validate the methodology and the hypotheses.

A. Simulation with real usage values

We now analyze the performance of the methodology on the Section IV-B infrastructure using the simulation protocol presented in Section IV-H1.

The ORCA$_S$ algorithm is applied over measured and predicted values, and compared to the initial cluster power consumption over a period of one week. The consolidation is applied at each iteration (every 15 minutes). VMs are separated into 3 groups using the parameters in Table V. ORCA is applied both on predicted and measured values to evaluate the accuracy of the algorithm. As in Section IV-E, the results are compared between measured usage for a week n and the predicted values for the same week n.

TABLE V: Simulation parameters for 3 groups resolution

Group name	**S**	**M**	**L**
Usage rate (r)	r≤20%	20%<r<80%	r≥80%
Commitment ratio	3	1.5	1

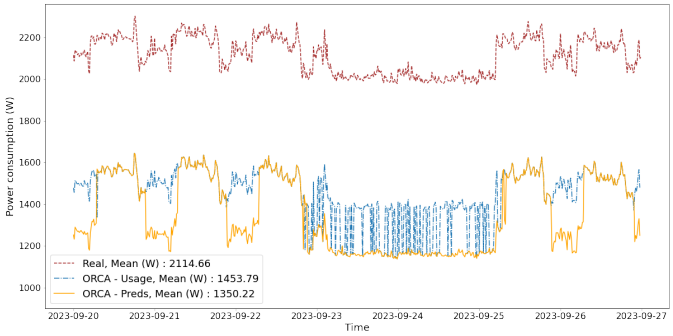


Fig. 7: ORCA$_S$ resulting power consumption using predicted (orange) and measured (blue) values

Fig. 7 shows the power consumption of the cluster PMs after applying the ORCA consolidation on both predicted (orange) and measured (blue) values. The power consumption after consolidation is always lower than the initial power consumption (red), with an average 36.1% consumption reduction on predicted values. Nevertheless, our 2 consolidated power consumption differs, meaning there are differences between the predicted and measured values and the resulting consolidation groups. The calculated error rate is 59.8%, focusing solely on PM group undersizing without weighting. This behavior is already underlined in Section IV-E. To mitigate these errors, we can apply the previously identified filter: the threshold. By using the 0.2 value for the threshold, we obtain the results of Fig. 8. Using our filter, we are able to lower our error rate to 0%, as the consumption graphs for predicted and measured values merge. We observe a drawback in terms of power consumption reduction as it falls to 29.8%.

To validate the proposed error rate mitigation, simulations are run using ten values of threshold over multiple weeks. Fig. 9 shows how ORCA performs for a specific week using multiple values of threshold in terms of power consumption reduction and errors, using all the macro error rate ratios presented in Section IV-G2a. $Q(t)$ and $Q_m(t)$ correspond respectively to the *Undersizing* and *RAM Undersizing* graphs, which coincide on the figure. Increasing the threshold value naturally increases the error rate of the consolidation, without proportionally reducing the power consumption. The figure

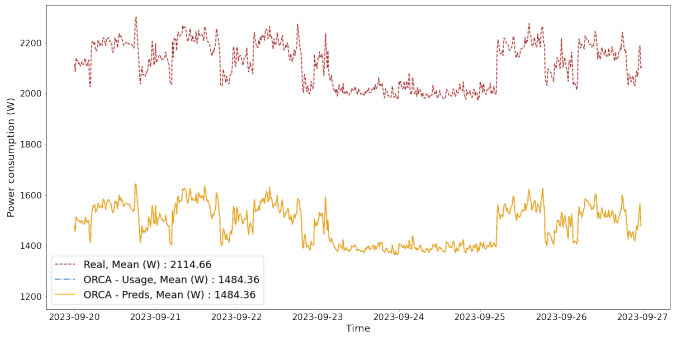


Fig. 8: ORCA$_S$ resulting power consumption using predicted (orange) and measured (blue) values after allocation filter

presents the base power consumption, as well as the consolidated consumption and a reduction baseline. The reduction baseline corresponds to the minimal power consumption reduction obtained with the sole use of consolidation: it is calculated by using a threshold of 0, meaning that all VMs are consolidated into the highest QoS consolidation group. In every scenario, the ORCA solver produces an additional power consumption reduction, enabled by the use of memory overcommitment. A reduction baseline of 0% here means that the cluster is originally well-sized in terms of memory requirements.

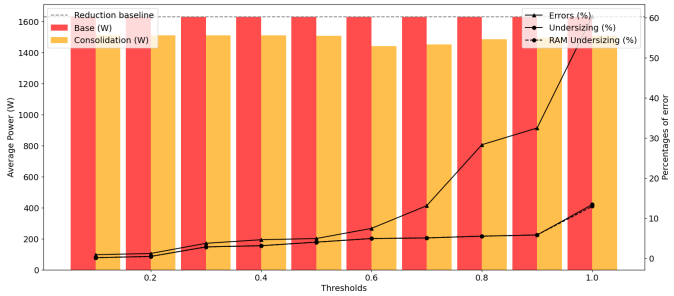


Fig. 9: Consolidation results using multiple thresholds values

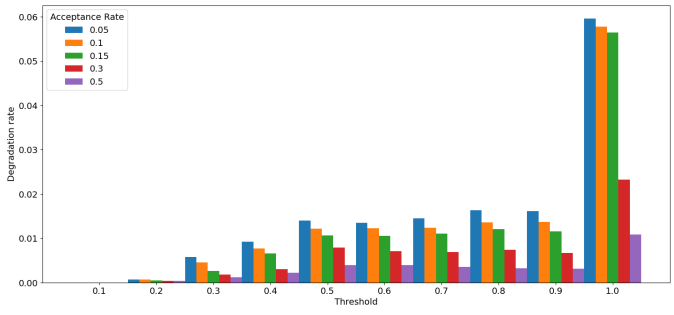


Fig. 10: Micro scale degradation - Performance degradation rate

Fig. 10 shows that threshold increases have an impact on the global performance degradation, especially when it is set to 1. As the value of the threshold increases, the micro scale degradation rate also increases, no matter the tolerance rate. This tolerance rate also influences the results, mitigating degradation as it increases. The simulation produces encouraging results here, particularly for the first two threshold values, for which the performance degradation rate is less than 1%.

Numerous versions of the ORCA solver have been tested and analyzed, varying the consolidation group and PM set sorting algorithms, as well as enabling and disabling buffer and migration mechanisms. This enabled us to validate the methodology and extract a high-performance approach for the experimental cluster. Once done, this specific version of ORCA$_S$ was tested over multiple weeks, allowing us to get a threshold value that both reduces the power consumption over the reduction baseline and mitigates the error rates.

B. Comparison with existing algorithm

To measure the performance of ORCA$_S$ against another optimization algorithm, we implemented an optimal Bin Packing Solver and compared the resulting power consumption. Our Bin Packing Solver objective is to minimize the number of used PMs and uses multiple constraints:

- VMs must fit onto the PMs in terms of vCPUs.
- VMs must fit onto the PMs in terms of memory.
- VMs with different overcommitment values cannot be on a similar PM.

The results are presented in Fig. 11. Both ORCA$_S$ (orange) and the optimal Bin Packing Solver (blue) offer lower power consumption than the initial measured consumption of the cluster (red). Over one week, ORCA$_S$ provides better results than the Bin Packing Solver.

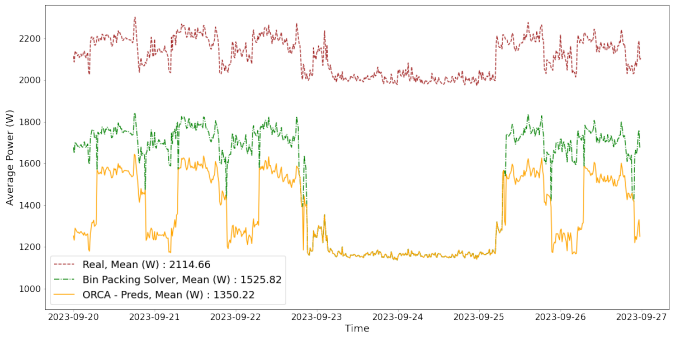


Fig. 11: ORCA$_S$ (orange) and Bin Packing (Green) power consumption after consolidation

During simulations, the average execution time of both solvers is measured. ORCA$_S$ outperforms the Bin Packing Solver as it executes in 1.1ms on average, compared to 1.01s for the Bin Packing Solver. ORCA$_S$ offers an average 29% power consumption reduction with an error of less than 0% on our experimental cluster, meaning it provides better results than an optimal Bin Packing Solver.

Our methodology does not aim to solely evaluate the performance of a VM placement heuristic. It tackles all steps of the consolidation process and delegates management of final

VM positioning to the hypervisor orchestrator. The selection of VMs to migrate at each consolidation step is directly dependent on the predicted usage value of the VM. All the VMs are considered at each step of the consolidation, and the selection of VMs to migrate is not dependent on an algorithm that can be compared to the state of the art. Then, the number of hosts to be used is reduced by computing the fewest number of required PMs for each consolidation group. Other consolidation solver approaches, such as meta-heuristic or genetic algorithms, might be used to find the best VM placement for each consolidation group. This will not reduce, in our case, the number of PMs used, as it is computed based on the amount of resources (CPU and RAM) needed by each consolidation group. VM placement is therefore delegated to the hypervisor, which will ensure that VMs are placed on a host of the same consolidation group. This results in better completion times for solver execution, which can be a limitation when scaling, without compromising power consumption reductions.

C. Scalability of the algorithm

To perform a first evaluation of the scalability of the methodology, the simulation process defined in Section IV-H1 is tested on another *SynAApS* cluster composed of more than 1000 VMs and 16 PMs. This cluster uses the same hypervisor and a centralized storage array, meaning it presents similarities with the targeted infrastructure. Usage metrics are gathered with the same collection rate and retention periods, and prediction models are trained using the same combination of parameters and features.

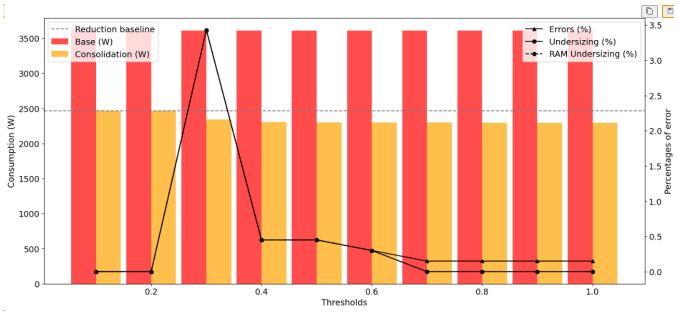


Fig. 12: Consolidation results using multiple thresholds values on a different cluster

Fig. 12 shows how ORCA performs on this other cluster for a specific week using multiple values of threshold. Here, the reduction baseline is significantly lower than the cluster base power consumption, meaning that this infrastructure is oversized. The threshold has less impact on the potential performance degradation at the macro scale : all threshold values keep the degradation rate below 3.5%. This means that our prediction models perform even better on this infrastructure. The micro scale performance degradations do not exceed 1.6% and are below 1% for most threshold/tolerance rate combinations.

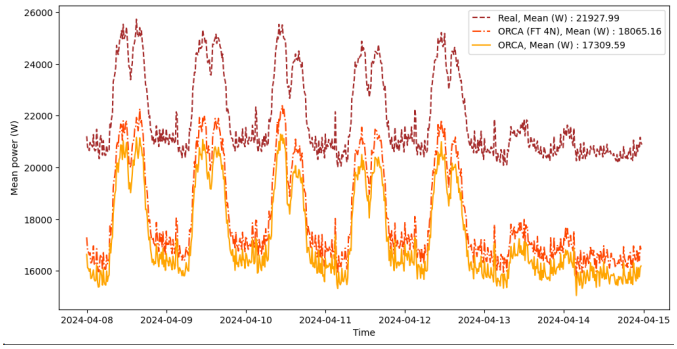


Fig. 13: Consolidation results at scale with and without fault tolerance

Further simulations were conducted based on usage values of this cluster. We simulated an infrastructure with 5520 VMs and 85 PMs, on which we also implemented fault tolerance. Fig. 13 The execution of ORCA without fault tolerance leads to a 20.2% power consumption reduction at scale. A fault tolerance rate of $4N$ is experimented here, meaning that the equivalent of 4 PMs can crash without causing service interruption. With this fault tolerance rate, we obtain a 16.68% power consumption reduction. Last, we conducted similar simulations on a simulated cluster of 11040 VMs and 170 PMs. Without fault tolerance, the power consumption is reduced by 20.97% over one week. With a $4N$ fault tolerance rate, we can reduce power consumption by 19.16%. This validates the use of the proposed methodology on a bigger virtualization cluster with heterogeneous VMs having different applications and usage. This also highlights the impact of using fault tolerance in the results of the consolidation. At scale, it has less impact as the base power consumption is significantly higher.

D. Deployment on the production infrastructure

A deployment of the ORCA$_S$ algorithm through the ORCA Controller is conducted in the targeted production infrastructure over a one-week period, using a threshold of 0.2.

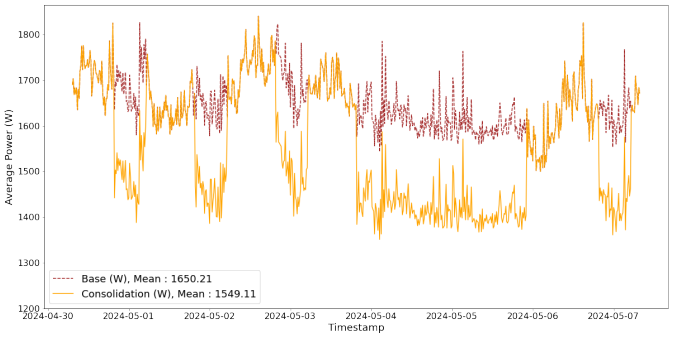


Fig. 14: Base and consolidated consumption during deployment

Fig. 14 shows the base and consolidated average power consumption of all the PMs of the infrastructure. Thanks to the consolidation, the power consumption is reduced here by

6.12%. This result, which is lower than simulation estimates, can be explained by 2 reasons. First, the cluster base power consumption is significantly lower than in simulation. This is due to a better cluster sizing that can be partially explained by exchanges with the infrastructure team after demonstrations of the simulation results. Second, a fault tolerance of $2N$ is used to prevent service interruption if 2 PMs crash during the deployment. As shown in Section V-C, this has an impact on the power consumption reduction, especially with an infrastructure of 7 PMs.

Fig. 15: Sum of ballooning, swapping and page sharing on each PM during deployment

To validate the hypothesis that memory overcommitment enables these consumption reductions, ballooning, swapping, and page sharing are also measured on PMs. Fig. 15 highlights a significant use of memory overcommitment mechanisms on one of the PMs of the infrastructure. PM names correspond to the anonymized names of the production cluster PMs. These mechanisms take place over the same periods as the power consumption reduction shown in Fig. 14, which are periods of inactivity: at night and during weekends. By comparing these observations with the deployment execution logs, we were able to validate the use of a PM for VMs not in use during these periods.

Thanks to the simulations, the methodology was tested, demonstrating that it is possible to achieve significant reductions in power consumption without causing undesirable performance degradation. The deployment also validated a production implementation of the methodology, with smaller reductions in terms of power consumption as it must consider fault tolerance. The deployment finally validated the hypothesis of memory overcommitment mechanisms utility, and did not cause performance degradation on the infrastructure. During the deployment, no incidents were reported affecting VM performance.

VI. Discussions

The methodology aims to be adapted to different infrastructure topologies, especially by using different sorting techniques for groups of VMs and PMs. Of course, varying the number of consolidation groups, the allocated overcommitment values, or deciding whether or not to use the group migration function may lead to completely different results. All these parameters need to be studied and selected carefully as we proposed in the methodology.

The relatively small number of PMs in the experimental infrastructure encouraged us to use a small number of consolidation groups. However, given that this infrastructure is used for internal *Ciril GROUP* needs, it is relatively easy to include users to enhance consolidation results. Works derived from ORCA that include and analyze user behavior were conducted in the infrastructure, enhancing the power consumption reduction from 6.12 to 12.58% [35].

The production deployment was not conducted on the second *SynAApS* cluster as it hosts *Ciril GROUP* customer's applications and VMs. Performance degradation on such VMs may result in SLA violation and financial penalties for the cloud operator as mentioned in Section I. More experimental validations of the methodology on non-critical infrastructure may help promote its adoption. Scalability represents multiple challenges for the consolidation implementation. First, the number of metrics to gather may increase proportionally to the infrastructure size, which may cause storage difficulties. The number of prediction models to train as well as the consolidation solver completion time are also directly affected by the size of the infrastructures. While some parameters in our implementation can be revised for larger infrastructures, many of the choices made already answer these challenges: small prediction models without the use of Deep Learning, a consolidation Solver that does not rely on an Optimal Bin Packing Solver and gives similar results in lower completion time.

During the various simulations and the preparation of the experiments, discussions were held with the team responsible for managing the experimental infrastructure. The first simulation results led the infrastructure team to naturally reduce the sizing of the experimental infrastructure. Coupled with the fault tolerance mechanisms, this explains the difference between the first simulation results and the obtained experimental results. However, this reinforces the idea that having a comprehensive methodology with preliminary studies of the use of VMs for example, or other mechanisms presented in the paper, can include different stakeholders and be beneficial in reducing DCs power consumption. It may increase users and operators awareness to further reduce other data centers environmental impacts such as Abiotic Depletion Potential (ADP), Global Warming Potential (GWP) or Primary Energy (PE) usage.

VII. Conclusion and Future Works

Efficient resource management is today a challenge for many Cloud service providers, as it is a lever to reduce a data center environmental impacts. The growing usage of digital solutions, associated with rebound effects, generates a perception of infinite resources among users and results in oversized infrastructures. To meet these challenges, DC operators can use methods such as elasticity or VM consolidation, but financial and technical restrictions can hinder their adoption.

This paper proposes a complete consolidation methodology based on black-box observations of VM resources consump-

tion to evaluate the mean utilization of VMs over time. We separate VMs into different subsets based on their usage and consolidate them on PMs with a defined overcommitment ratio. The least-used VMs are grouped together on PMs with a high overcommitment ratio and a high VM density. Contrarily, the most-used VMs are consolidated to avoid performance degradation and offer the best quality of service. The use of a high memory overcommitment ratio for idle VMs is enabled by mechanisms like ballooning, swapping, or TPS. The consolidation methodology was tested in simulations based both on predictions and observations made from two virtualization clusters and provides power consumption reduction as high as 29.8% without errors. Simulations enabled us to test multiple consolidation methods, using different sorting techniques for VMs and PMs groups. With a good combination of parameters experimented on over multiple weeks, a threshold parameter was identified for a production implementation, whose purpose is to limit consolidation errors or undersizing while guaranteeing power consumption reduction attributable to memory overcommitment mechanisms. Lastly, a production implementation was carried out to validate a proof of work of our methodology, providing a 6.12% power consumption reduction when considering a fault tolerance factor of $2N$. For future work, we aim to test and deploy our methodology on other bigger virtualization clusters with more than 100 PMs, and use projections to estimate not only the power consumption reductions but other DCs environmental impacts reduction using a Life Cycle Assessment (LCA) approach.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data can be available on request. Please contact authors.

Acknowledgements

This work was funded by ANRT (CIFRE N°2021/1297).

References

[1] Yahya Al-Dhuraibi, Fawaz Paraiso, Nabil Djarallah, and Philippe Merle. Elasticity in Cloud Computing: State of the Art and Research Challenges. *IEEE Transactions on Services Computing*, 11(2):430–447, March 2018.

[2] Mathieu Bacou, Grégoire Todeschi, Alain Tchana, Daniel Hagimont, Baptiste Lepers, and Willy Zwaenepoel. Drowsy-DC: Data center power management system. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, page 825, May 2019.

[3] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, October 2003. Association for Computing Machinery.

[4] Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan. Energy and Power Efficiency. In Luiz André Barroso, Urs Hölzle, and Parthasarathy Ranganathan, editors, *The Datacenter as a Computer: Designing Warehouse-Scale Machines*, pages 99–127. Springer International Publishing, Cham, 2019.

[5] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Future Generation Computer Systems*, 28(5):755–768, May 2012.

[6] Anton Beloglazov and Rajkumar Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420, 2012.

[7] Anton Beloglazov and Rajkumar Buyya. Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints. *IEEE Transactions on Parallel and Distributed Systems*, 24(7):1366–1379, July 2013.

[8] G Bharanidharan and S Jayalakshmi. Predictive Virtual Machine Placement for Energy Efficient Scalable Resource Provisioning in Modern Data Centers. In *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 299–305, March 2021.

[9] Saima Fayaz Chashoo and Deepti Malhotra. VM_mig_framework: Virtual Machine Migration With and Without Ballooning. In *2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pages 368–373, December 2018.

[10] Yuxuan Chen, Zhen Zhang, Yuhui Deng, Geyong Min, and Lin Cui. A Combined Trend Virtual Machine Consolidation Strategy for Cloud Data Centers. *IEEE Transactions on Computers*, 73(9):2150–2164, September 2024.

[11] Susanta Nanda Tzi-cker Chiueh. A Survey on Virtualization Technologies. *RPE Report*, 142, 2005.

[12] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167, Shanghai China, October 2017. ACM.

[13] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes. Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment. *IEEE Network*, 29(2):56–61, March 2015.

[14] Praniali P. Deshmukh and S.Y. Amdani. Virtual Memory Optimization Techniques in Cloud Computing. In *2018 International Conference on Research in Intelligent and Computing in Engineering (RICE)*, pages 1–4, August 2018.

[15] Alexandre H. T. Dias, Luiz. H. A. Correia, and Neumar Malheiros. A Systematic Literature Review on Virtual Machine Consolidation. *ACM Computing Surveys*, 54(8):176:1–176:38, October 2021.

[16] Fahimeh Farahnakian, Adnan Ashraf, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Ivan Porres, and Hannu Tenhunen. Using Ant Colony System to Consolidate VMs for Green Cloud Computing. *IEEE Transactions on Services Computing*, 8(2):187–198, March 2015.

[17] Tiago C. Ferreto, Marco A. S. Netto, Rodrigo N. Calheiros, and César A. F. De Rose. Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems*, 27(8):1027–1034, October 2011.

[18] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon S. Blair, and Adrian Friday. The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations. *Patterns*, 3(8), August 2022.

[19] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Chasing Carbon: The Elusive Environmental Footprint of Computing. *arXiv:2011.02839 [cs]*, October 2020.

[20] Najet Hamdi and Walid Chainbi. A survey on energy aware VM consolidation strategies. *Sustainable Computing: Informatics and Systems*, 23:80–87, September 2019.

[21] Leila Helali and Mohamed Nazih Omri. A survey of data center consolidation in cloud computing systems. *Computer Science Review*, 39:100366, February 2021.

[22] Sebastian Herbert and Diana Marculescu. Analysis of dynamic voltage/frequency scaling in chip-multiprocessors. In *Proceedings of the 2007 international symposium on Low power electronics and design (ISLPED '07)*, pages 38–43, August 2007.

[23] Nikolas Herbst, Samuel Kounev, and Ralf Reussner. Elasticity in Cloud Computing: What it is, and What it is Not. In *10th International Conference on Autonomic Computing (ICAC 2013)*, pages 23–27, San Jose, CA, June 2013.

[24] Nguyen Trung Hieu, Mario Di Francesco, and Antti Ylä-Jääski. Virtual Machine Consolidation with Multiple Usage Prediction for Energy-Efficient Cloud Data Centers. *IEEE Transactions on Services Computing*, 13(1):186–199, January 2020.

[25] Michael R. Hines and Kartik Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '09, pages 51–60, New York, NY, USA, March 2009. Association for Computing Machinery.

[26] Md. Kaviul Hossain, Mutasimur Rahman, Azrin Hossain, Samin Yeaser Rahman, and Md. Motaharul Islam. Active & Idle Virtual Machine Migration Algorithm- a new Ant Colony Optimization approach to consolidate Virtual Machines and ensure Green Cloud Computing. In *2020 Emerging Technology in Computing, Communication and Electronics (ETCCE)*, pages 1–6, December 2020.

[27] Sun-Yuan Hsieh, Cheng-Sheng Liu, Rajkumar Buyya, and Albert Y. Zomaya. Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. *Journal of Parallel and Distributed Computing*, 139:99–109, May 2020.

[28] IEA. Data Centres and Data Transmission Networks – Analysis, July 2023.

[29] Saikishor Jangiti, V.S. Shankar Sriram, and R. Logesh. The role of cloud computing infrastructure elasticity in energy efficient management of datacenters. In *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, pages 758–763, September 2017.

[30] Congfeng Jiang, Yumei Wang, Dongyang Ou, Youhuizi Li, Jilin Zhang, Jian Wan, Bing Luo, and Weisong Shi. Energy efficiency comparison of hypervisors. *Sustainable Computing: Informatics and Systems*, 22:311–321, June 2019.

[31] Yichao Jin, Yonggang Wen, and Qinghua Chen. Energy efficiency and server virtualization in data centers: An empirical investigation. In *2012 Proceedings IEEE INFOCOM Workshops*, pages 133–138, March 2012.

[32] Tarandeep Kaur and Inderveer Chana. Energy Efficiency Techniques in Cloud Computing: A Survey and Taxonomy. *ACM Computing Surveys*, 48(2):22:1–22:46, October 2015.

[33] Jonathan Koomey, Stephen Berard, Marla Sanchez, and Henry Wong. Implications of Historical Trends in the Electrical Efficiency of Computing. *IEEE Annals of the History of Computing*, 33(3):46–54, March 2011.

[34] Jitendra Kumar. Long Short Term Memory Recurrent Neural Network (LSTM-RNN) Based Workload Forecasting Model For Cloud Datacenters. *Procedia Computer Science*, page 7, 2018.

[35] Simon Lambert, Eddy Caron, Laurent Lefèvre, and Rémi Grivel. S-ORCA : A social-based consolidation approach to reduce Cloud infrastructures energy consumption. In *15th IEEE International Conference on Cloud Computing Technology and Science - Cloudcom 2024*, pages 119–126, Abu Dhabi, United Arab Emirates, December 2024. Khalifa University.

[36] Haikun Liu, Hai Jin, Xiaofei Liao, Wei Deng, Bingsheng He, and Cheng-zhong Xu. Hotplug or Ballooning: A Comparative Study on Dynamic Memory Management Techniques for Virtual Machines. *IEEE Transactions on Parallel and Distributed Systems*, 26(5):1350–1363, May 2015.

[37] Eric Masanet, Arman Shehabi, Nuoa Lei, Sarah Smith, and Jonathan Koomey. Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986, February 2020.

[38] Seyedhamid Mashhadi Moghaddam, Michael O'Sullivan,

Cameron Walker, Sareh Fotuhi Piraghaj, and Charles Peter Unsworth. Embedding individualized machine learning prediction models for energy efficient VM consolidation within Cloud data centers. *Future Generation Computer Systems*, 106:221–233, May 2020.

[39] Dorian Minarolli, Artan Mazrekaj, and Bernd Freisleben. Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing. *Journal of Cloud Computing*, 6(1):4, February 2017.

[40] Openstack. Overcommitting CPU and RAM, September 2022.

[41] Neeraj Kumar Sharma and G. Ram Mohana Reddy. Multi-Objective Energy Efficient Virtual Machines Allocation at the Cloud Data Center. *IEEE Transactions on Services Computing*, 12(1):158–171, January 2019.

[42] VMWare. CPU Virtualization Basics, May 2019.

[43] Carl A Waldspurger. Memory resource management in VMware ESX server. *ACM SIGOPS Operating Systems Review*, December 2002.

[44] Andreas Wolke, Boldbaatar Tsend-Ayush, Carl Pfeiffer, and Martin Bichler. More than bin packing: Dynamic resource allocation strategies in cloud data centers. *Information Systems*, 52:83–95, August 2015.

[45] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Computer Networks*, 53(17):2923–2938, December 2009.

[46] Jing Zeng, Ding Ding, Kaixuan Kang, HuaMao Xie, and Qian Yin. Adaptive DRL-Based Virtual Machine Consolidation in Energy-Efficient Cloud Data Center. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2991–3002, November 2022.

[47] Yonghua Zhu, Weilin Zhang, Yihai Chen, and Honghao Gao. A novel approach to workload prediction using attention-based LSTM encoder-decoder network in cloud environment. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):274, December 2019.