CloudFactory: An Open Toolkit to Generate Production-like Workloads for Cloud Infrastructures

Pierre Jacquet, Thomas Ledoux, Romain Rouvoy

CLOUDFACTORY: An Open Toolkit to Generate Production-like Workloads for Cloud Infrastructures

Pierre JACQUET
Inria, Univ. Lille, CRIStAL,
UMR CNRS 9189, France
pierre.a.jacquet@inria.fr

Thomas LEDOUX
IMT Atlantique, Inria, LS2N,
UMR CNRS 6004, France
thomas.ledoux@inria.fr

Romain ROUVOY
Univ. Lille, Inria, CRIStAL,
UMR CNRS 9189, France
romain.rouvoy@inria.fr

***Abstract*—Cloud infrastructures are large-scale and complex platforms designed to host a wide diversity of applications and workloads. Given these complexity and scale factors, simulators and benchmarks are broadly adopted *in vitro* to study their behaviors, prototype new software components and heuristics, and evaluate their effective performances.**

However, both state-of-the-art simulations and benchmarks may suffer from a representativeness problem, as the reported results can vary depending on their input workloads. For example, a *Infrastructure-as-a-Service* (IAAS) platform aims to host *Virtual Machine*s (VMs), whose characteristics (resource configurations, workload intensity, arrival/departure rate, etc.) can greatly differ depending on Cloud providers and public/private deployments. Addressing this IAAS representativeness thus requires Cloud providers to share production-scale datasets, which might be considered sensitive. Moreover, Simulations and benchmarks require a specific experiment scenario that cannot be easily generated from Cloud providers characteristics.

To address these issues, this paper introduces CLOUDFACTORY, a IAAS workload generator. Our contribution is first composed of a library that can be used by Cloud providers to share IAAS statistics, instead of raw datasets. Then, we introduce a generator designed to produce realistic VM workloads that match these statistics. CLOUDFACTORY is made available as open-source software that can be adopted by Cloud providers and researchers to foster the evaluation of new contributions.

As an example, we perform an analysis on scheduling evolution for different IAAS workload intensity of two different Cloud providers: Microsoft Azure and Chameleon. We also report on OVHcloud statistics computed from CLOUDFACTORY and compare them to other Cloud providers.

***Index Terms*—Benchmark, Cloud, generator, IAAS, representativeness, testbed, workload**

I. INTRODUCTION

For more than a decade, Cloud computing keeps being an active field of academic and industrial research, encompassing multiple challenges and contributions that are continuously delivered to improve the state of the art. Nevertheless, assessing effective contributions in the domain of Cloud computing is notoriously hard [1], in particular, because of the difficulty to reproduce production-scale IAAS deployments.

To overcome this limitation, Cloud simulation platforms are commonly considered in the literature to evaluate new contributions for Cloud infrastructures [2]. Alternative contributions to this field include datasets and benchmarks that can be released and published by both academic and industrial practitioners. Unfortunately, Cloud providers may be reluctant to share production-scale datasets, due to the possibility of user identity exposure through trace information [3]. Furthermore, both Cloud-based simulations and experimental deployments tend to suffer from the same limitations in terms of representativeness. In particular, the replay of benchmarks and raw datasets may be difficult to apply when lacking production-scale infrastructures, while raw traces may be truncated to deal with the limitation of a testbed infrastructure, hence questioning the representativeness of the resulting experiments. Furthermore, the applicability of developed contributions in a production-scale environment may be challenged by this limitation.

The main objective of this paper, referred to as CLOUDFACTORY, is to enhance the evaluation and validation of Cloud contributions by allowing to use production-like Cloud workloads. More specifically, CLOUDFACTORY addresses the challenge of reproducing IAAS workloads by leveraging representative infrastructure-scale models extracted from datasets of production-scale execution traces. To achieve this, CLOUDFACTORY combines two key components that can be used independently by Cloud practitioners and researchers to share insightful statistics that can be exploited to generate IAAS workloads at any scale—from single worker nodes to clusters—and apply them in simulations or experimental deployments. In particular, thanks to the workload analyzer that we made available, Cloud providers can share production-scale metrics without disclosing their raw datasets. Then, computer scientists can import these statistics into the workload generator that we packaged as a separate tool to produce various IAAS workloads that they can easily use and further share to evaluate their contribution, thanks to the bindings we deliver for several state-of-the-art evaluation platforms. Throughout the paper, we illustrate the value of CLOUDFACTORY by reporting on the analysis of the Microsoft Azure dataset [4] and the generation of IAAS workloads for the CLOUDSIMPLUS simulation environment [5] and the CBTOOL benchmarking tool [6].

In the remainder of this paper, we report on related works (see Section II), present an overview of CLOUDFACTORY architecture (see III) before diving into its design (see Sec-

tion IV, Section V, and Section VI). Then, we explore a case study enabled by our contribution and share CLOUDFACTORY generated statistics from OVHcloud context (see Section VII). We finally discuss limits (see Section VIII) before conclude on this work and its perspectives (see Section IX).

II. RELATED WORK

When evaluating Cloud-based systems, researchers may adopt two complementary approaches. The first involves *simulations* to synthesize various system configurations and workloads, thereby avoiding the cost of resource-intensive and time-consuming real-world experiments. The second approach involves *benchmarks*, in which real-world workloads are applied to a system under test, which may be either a generic system or a copy of an existing one. Unfortunately, both of these approaches often face a representativeness challenge, where injected workloads must be as faithful as possible to real-world conditions. Cloud metrics reported by practitioners are a key insight in this regard. This section outlines the state of the art for Cloud simulators, *System under Test* (SUT) benchmarks, workload representativeness, as well as relevant Cloud datasets.

A. Cloud simulators

CLOUDSIM [7] is an extensible software framework to simulate a Cloud-based infrastructure. Its flexible nature enables users to create customizable platforms under various workload scenarios, such as VM (Virtual Machine) distribution, arrival rate, CPU usage, etc. CLOUDSIMPLUS [5] was introduced as a fork, enhancing its capabilities. Crossref[1] records indicate that the original framework is widely adopted by the research community, with over $2,700$ citations.

While other Cloud data center simulators exist, like closed-source MDCSIM [8] and GREENCLOUD [9], simulators expect developers to encode synthetic deployment scenarios rather than providing guidelines on realistic Cloud workloads. For instance, the MDCSIM validation protocol relied on a custom scenario to compare its resource utilization to a physical infrastructure hosting a web-oriented application under a specific workload. Similarly, GREENCLOUD and CLOUDSIM used specific evaluation prototypes as examples.

SIMGRID [10] is a more generic solution than can be applied to the Cloud via its S4U interface. It can run a VM-based scenario and evaluate scheduling decisions on physical resources. VMPLACES framework [11] leverages it to inject realistic VMs usage. However, it does not consider VM lifespan and arrival rate.

B. Cloud System under Test

In the context of Cloud computing, state-of-the-art benchmarks typically build on the combination of two key components. First is the *System under Test* (SUT), which is derived from production environments, and can either be the copy of an existing system or a system considered representative of real ones. Second is the input workload, generally described as the usage pattern submitted to the SUT. We now discuss common SUT studied in a Cloud context, while input workload is further discussed in the following section (see Section II-C).

1) Application under Test: Hosted applications are the smallest SUT unit in a Cloud experimentation [12]. Metrics of interest include response time, request rates, errors, etc. [13].

Due to the wide adoption of Cloud infrastructures, many types of applications can be hosted. A non-exhaustive list of legacy applications includes relational database structures proposed by TPC [14], CPU-intensive tasks, such as those provided by SPEC CPU [15], mail server [16], and others.

Additionally, web-oriented architectures are also frequently considered and tested using various projects, such as the OLIO monolithic application [17] and other microservice architectures implemented in the DEATHSTARBENCH [18]. Key-value stores are also widely deployed and can be evaluated using the YCSB benchmark [19].

2) VM under Test: Assessing VM performance for a given application is beneficial to Cloud clients. In this configuration, the application and the workload generator may not be co-located, to evaluate the platform induced latency.

The choice between hosting options can be made by evaluating the balance between performance and cost for various VM sizes and providers. Work targeting providers benchmark includes MOSAIC [20, 21], CLOUDCRAWLER [22] framework, and C-METER [23] where the monitored application can be configured. Other approaches compare application metrics on provided benchmarks [6, 17, 24].

3) Platform under Test: Cloud platform consists of physical servers, known as *nodes*. These nodes can be organized as *clusters*, which are groups of servers used for a specific Cloud workload, such as hosting VM with the same premium policy. Conducting experiments at the scale of a node can provide significant benefits for both providers and researchers, as it can evaluate various platform configurations and architecture choices. In the context of multiple VMs, the node represents the smallest SUT where deployments can be studied. In this context, a deployment is defined as the provisioning of a VM on a node according to its requested resources.

SPECVIRT benchmark [25] offers a mean of measuring end-to-end performances, including hardware, virtualization platform, guest operating system on various application types. While being behind a paywall, they also do not target workload representativeness. They use an increasing workload until *Quality of Services* (QoS) are violated.

Another available option is the VMMARK benchmark, provided by VMWare [26]. This benchmark deploys simultaneously 6 VMs hosting different workloads: mail server, java server, idle workload, web server, database server, and a file server before evaluating their performance under stress. They do not seem to take into account larger deployments, VM lifespan, departure, and arrival rates.

The management complexity associated with studying multiple Cloud nodes has led to the development of dedicated frameworks. A prominent example is CBTOOL [12], which has become the *de facto* standard for describing an IAAS

[1] https://www.crossref.org/

scenario through a given number of nodes, VM, and usage parameters. Interpreted scenarios can be applied to both private clusters and public providers. Single-node experiments are also an option.

C. Input Workloads

Cloud-like workloads applied to SUT aim to be as representative as possible. Two strategies are typically used: *replaying recorded traces* or *using models*.

Replaying traces [27, 28] involves resubmitting production requests, which provides a comprehensive representation of the workload, but may not be available in all cases.

Alternatively, workload models can be used to represent an input workload behavior. Basic usage may rely on a targeted rate (requests per second), while other generators include patterns and burst mechanisms [29].

Workload representativeness is typically focused on the application or VM level. To the best of our knowledge, few previous works provide workloads at the node or cluster scale, even if it introduces additional metrics, such as VM arrival rate, VM lifespan, used resources, etc. One exception is [30], which implements a specific workload scenario on CBTOOL, although the adopted parameters are not publicly communicated due to a paywall.

Our work aims to improve the representativeness of Cloud node and cluster workloads by leveraging provider datasets or statistics.

D. Cloud datasets

Different Cloud datasets are made available by Cloud providers. Azure IAAS workload characteristics has been provided in a previous study [4]. It includes, amongst others, VM memory and vCPU configuration options distribution, arrival rate patterns, VM lifespan, and CPU usage daily pattern. Despite lacking crucial figures, such as memory usage, this dataset stands as a valuable contribution, given its status as the only publicly available Cloud dataset to the authors' knowledge.

However, it is important to acknowledge that Cloud workloads may significantly differ from one another due to their specific contexts. For instance, research-oriented Cloud operator CHAMELEON [31] published 6-month traces in 2020 where half of their deployments imply VMs having at least 4 vCPUs, while Azure reports that less than 2 vCPUs VMs are required in 80% of their deployments. Publication of new datasets must therefore be encouraged.

In the realm of Cloud services, IaaS VMs are a prevalent offering. Managed services, like *Platform-as-a-Service* (PAAS) and *Function-as-a-Service* (FAAS), are typically internally built upon IAAS offerings. Furthermore, *Container-as-a-Service* (CAAS) offers comes with a higher level of management, as the host is responsible for managing the kernel. In this CAAS context, traces from Alibaba [32] and Google Borg [33] can be studied. However, there is a scarcity of datasets specifically focused on IAAS infrastructure and datasets from CAAS cannot be directly applied to a VM-based context. For instance, containers generally have shorter lifetimes compared to VMs.

E. Synthesis

The review of the state of the art, therefore, motivates a novel approach, referred to as CLOUDFACTORY, which aims to complement current progress in Cloud workloads by incorporating representative platform-scale workloads. Concretely, CLOUDFACTORY improves upon the existing state-of-the-art application generators by enabling their scaling at both node and cluster levels, while also adhering to high-level specifications of Cloud datasets. In this regard, our solution enables the generation of more realistic scenarios—*i.e.*, a set of specifications for a given Cloud platform—from a workload analyzer. This approach promotes the utilization and sharing of non-sensitive data, thereby enhancing the generalizability of the generator.

III. CLOUDFACTORY OVERVIEW

Figure 1 depicts an overview of the CLOUDFACTORY key components. In particular, we focus on generating IAAS workloads that are representative of production-scale deployments but can be provisioned at any scale—from single compute nodes to cluster-wide platforms or simulators.

To do so, we propose to convert raw datasets collected by Cloud providers into a compact set of IAAS statistics that are sufficient to deploy a production-like workload. Interestingly, this approach does not impose the disclosure of sensitive information from the perspective of Cloud providers, who can simply publish such cluster-wide statistics. It is the goal of the workload analyser component detailed in section IV.

The workload generator leverages these statistics to generate a usable experiment workload, with the appropriate VM set and usage models. Implementation is detailed in section V.

Finally, different binding options are available to exploit the results on several state-of-the-art evaluation platforms. Generated experiments workload may be executed on a simulator or on a physical system. Exporting capabilities are detailed in section VI.

IV. COMPUTE HIGH-LEVEL STATISTICS

After deployment of a client-selected configuration, a VM enters its usage phase, during which resource utilization can vary substantially. This variability can be observed in different ways, ranging from complete idleness to periodic activity, bursty behavior, short lifespan, etc. To account for VM behaviors, we introduce our workload analyzer. This library builds a set of VM usage profiles from a Cloud dataset.

The realism of workloads is contingent upon their ability to replicate the behavior of actual systems. While replay-based IAAS workloads meet this requirement, they are often impractical to implement—due to privacy restrictions that limit sharing options, which in turn impacts the reproducibility of experiments. Furthermore, production-scale Cloud clusters cannot be easily reproduced for SUT experimental purposes, thus posing a material constraint.

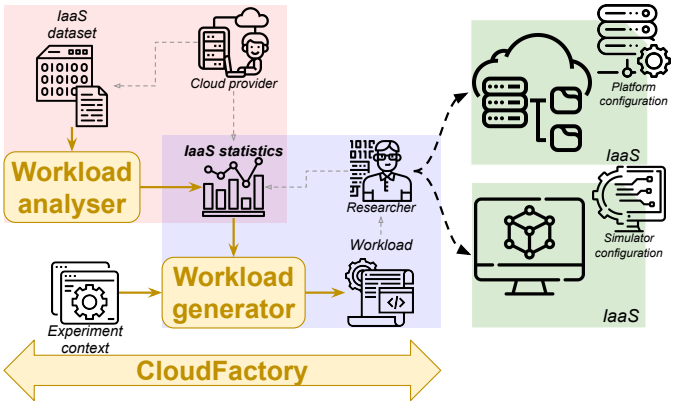

Fig. 1. Overview of CLOUDFACTORY

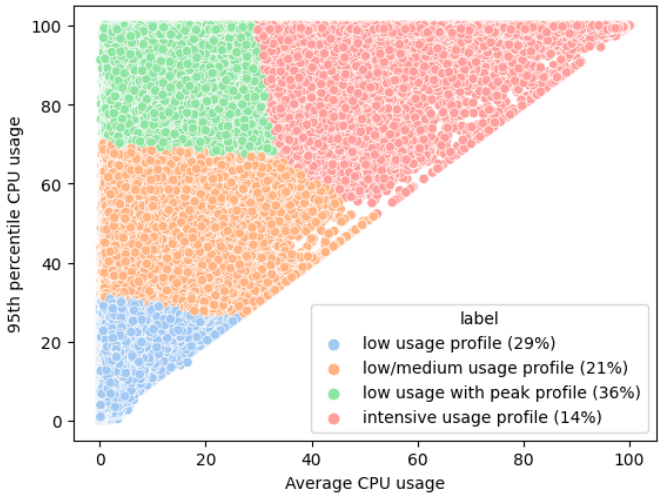

Fig. 2. 4 VM usage profiles captured from Azure 2017 using k-means clustering ($k = 4$)

A. Statistics identification

Metrics describing the SUT must be carefully selected to represent as faithfully as possible Cloud context specificities. Our identification of relevant statistics was also motivated by metrics availability from currently published datasets. Furthermore, this set of statistics is voluntarily kept as small as possible to facilitate computation and sharing from undisclosed Cloud datasets.

CLOUDFACTORY statistics can be classified in two categories. At the micro level, VMs resource usages are considered. At the macro level, management of a dynamic set of VMs is studied.

VM resource usage is considered from the CPU perspective. In addition to CPU utilization bounds (see Section IV-B), we also take into account usage periodicity (see Section IV-C).

To account for VM management, CLOUDFACTORY considers VM configuration options distribution (see Section IV-D) and departure/arrival ratios (see Section IV-E).

At the exception of VM configuration options distribution, statistics are computed according to profiles. A first step applies clustering to VMs, based on their usage similarities, to describe more accurately observed behaviors, as per-profile statistics. Options distribution is treated separately as offered configuration options may significantly change from one Cloud provider to another, which limits comparisons and exports options.

B. Computing usage

The workload analyzer first classifies VMs based on their average and percentile CPU usage metrics to identify resource utilization profiles. We have selected the k-means clustering algorithm, due to its capability to capture a given number of clusters. This capability allows Cloud providers to fine-tune the granularity of information exposed to researchers by customizing the number of profiles through the parameter k. Figure 2 illustrates this classification on the Azure dataset VMs on four profiles ($k = 4$). Different VMs clusters are deduced, from those with a low average and percentile usage to those with high average and percentile usage. Each cluster is subsequently translated into a profile, consisting of specific bounds on the observed average and percentile values. For instance, the blue cluster in Figure 2 is composed of VMs having an average CPU usage below 26% (X-axis), and peaks below 32% (captured by their 95th percentile, on Y-axis). VMs that match this profile are typically considered oversized. It is noteworthy that approximately 29% of the VMs observed in the Azure dataset belong to this particular category.

C. Periodicity ratio

VM periodicity is an important factor for scheduling strategies. A recurrent usage pattern can be leveraged by Cloud providers to identify interactive workloads [4]. In each cluster, we compute the ratio of VMs reporting a recurrent pattern considering a specified scope (set by default to one day). As in [4], we used a *Fast Fourier Transform* (FFT) algorithm to evaluate each VM behavior. Differences may remain, however, as the authors did not communicate the implementation details. Our implementation is described in algorithm 1. Specifically, we used a periodogram to estimate *Power Spectral Density* (PSD) frequencies. A VM is classified as *periodic* over a given window if the associated window frequency PSD exceeds a user-specified percentile in distribution. In the remainder of this document, the 99th percentile has been considered. On each profile, the periodicity is evaluated on a subset of VMs matching a lifespan duration condition, before being extrapolated to the overall quantities.

D. Computing VM distribution

VM configuration options distribution is an important statistic due to its impact on server consolidation. CHAMELEON and Azure datasets underline the disparity between the observed VM distributions. Therefore, we calculate the ratio of VM configuration, in terms of vCPU and vRAM, observed in Cloud datasets.

Algorithm 1 Perodicity detection algorithm

Input time series, period, threshold
Output Perodicity state

1: $\mathcal{M}_{periodogram} \leftarrow$ Generate periodogram from time series
2: $value \leftarrow \mathcal{M}_{periodogram}(period)$
3: $max \leftarrow percentile(\mathcal{M}_{periodogram}, threshold)$
4: **if** $value > max$ **then**
5: **return** PERIODIC
6: **end if**
7: **return** NOT PERIODIC

a) Avoiding bias: Counting the VM configuration occurrences at the dataset scale biases the results, due to the short lifespan of some deployments. Instead of considering the number of configurations based on deployments, we compute the average VMs distribution presence in nodes (see Equation 2).

$$\text{configuration}_{\text{freq}}(c) = \frac{\sum_{s \in S} \text{Frequency}(c, s)}{|S|} \quad (1)$$

This later facilitates experiments' representativeness in down-scaled clusters, where server configuration and arrival rates may be significantly lower than the ones observed in production, hence impacting the hosted distribution accuracy. The provided dataset is therefore considered through time windows of a chosen duration. Distribution presence is computed as the average value observed from all considered time slices (S). With a sufficiently short time-slice duration, this approach allows us to compute the average distribution of each VM configuration option at any given time, mitigating the impact of heterogeneous lifespan. The rest of this paper considered a one-hour time-slice duration.

b) Azure example: Interestingly, applying this method to the Azure dataset reveals significant differences with their values, as they adopted the "deployment approach". For instance, the 14 GB configuration option, which is claimed to account for 11% of the VMs, is found to be half as present, accounting for only 6% of VMs over a typical hour. The configuration has indeed the lowest 90th percentile lifespan in the dataset. VM-creation test workloads are reported to be common on VMs belonging to Azure [4] and may be based on this configuration, leading to a reduce average lifespan.

E. Departure & arrival ratios

The rate at which new VMs are deployed is a crucial metric for a IAAS, as it bears a significant influence on the provisioning capabilities of the platform. Conversely, the lifespan of VM deployments may be viewed as a complementary micro-level indicator. In light of our concentration on IAAS workloads, we opt for the macro-level equivalent, namely the departure ratio. Consequently, both arrival and departure ratios are computed for each identified VMs profile.

To analyze these metrics, we compute the number of newly arrived VMs relative to those who left within a default one-day window through all the analyzed datasets. With Equations 2 and 3, we transform this quantities into average ratio observed in the specific context. If the arrival ratio is greater than the departure ratio, the profile is considered as *increasing*, while the opposite reflects a *decreasing* profile. In the Azure dataset, we observe that low utilization workloads have a lower degree of dynamism, with reduced departure and arrival ratios compared to more intensive workloads.

$$\dot{\text{arrival}}_{\text{IAAS}} = \frac{\sum_{s \in S} \frac{|departure_{\text{VM}}(s)|}{|\text{VM}(s)|}}{|\text{S}|} \quad (2)$$

$$\dot{\text{departure}}_{\text{IAAS}} = \frac{\sum_{s \in S} \frac{|arrival\text{VM}(s)|}{|\text{VM}(s)|}}{|\text{S}|} \quad (3)$$

F. Profiles examples on Azure dataset

The profiles built by our workload analyzer include the following statistics:

- *Profile weight in Cloud dataset*,
- *Average CPU usage bounds*,
- *95th percentile CPU usage bounds*,
- *Arrival & departure ratio*,
- *Periodicity ratio*.

The IAAS statistics computed for the 4 profiles illustrated in Figure 2 are summarized in Table I.

G. Generated statistics.

The workload analyzer data related to VM distribution and usage are converted to YAML files. The YAML format was chosen due to its user-friendly nature that facilitates comprehension. These files are not expected to disclose any sensitive information and to enable sharing with the wider community. Furthermore, they can also be modified by developers seeking to explore specific conditions. Finally, they can be packaged with any Cloud experiment to offer a reproductible Cloud environment for researchers who would like to compare their contributions with previous works published with this toolkit.

V. GENERATE PRODUCTION-SCALE WORKLOADS

We now detail how to generate workload matching high-level specifications. Beyond importing statistics, generating a production-scale workload requires describing the infrastructure considered for the SUT. This also involves specifying experiment duration and temporality patterns.

Long-run experiments are simplified by CLOUDFACTORY through the temporality concepts of slices and scope. A slice represents the minimum duration considered in the experiment, while scope refers to a multiple of slices. This can be used in two ways. The first is to accelerate real experiments by changing the virtual day's duration (i.e., scope) and a virtual hour (i.e., slice). The second is to study the effects of predictability on various windows (e.g., diurnal pattern, and hourly patterns).

The workload generator combines the IAAS statistics shared by Cloud providers (or computed from our workload analyzer) with user input on desired workload size and on temporality to build a consistent set of VMs. The set is composed of VM behaviour attached to a VM configuration. A VM configuration is defined by characteristics related to provisioning:

TABLE I
DETAILED METRICS OF THE 4 USAGE PROFILES COMPUTED BY THE WORKLOAD ANALYSER ON AZURE DATASET

Usage Profile	Percentage	Avg. CPU bounds	95th percentile CPU bounds	Arrival rate	Departure rate	Periodicity rate
Low	29%	[0.0;25.9]	[0.0;31.2]	15%	15%	0.4%
Low/Medium	21%	[0.1;51.9]	[26.0;70.2]	32%	32%	0.8%
Low with peak	36%	[0.2;33.3]	[66.6;100.0]	60%	60%	1.1%
Intensive	14%	[29.3;100.0]	[52.1;100.0]	57%	57%	0.8%

- *number of vCPUs*,
- *size of allocated vRAM*,
- *time of provisioning* (as ticks),
- *lifespan* (as ticks/slices/slots).

While, the VM behaviour is characterized by:

- hosted *application binary*,
- VM *workload*.

A. On VM configuration generation

CLOUDFACTORY first generates a set of VM configuration according to requested ratios for the SUT. The user can request a specific number of VMs to be generated. A more practical option—especially when studying the effects of load—is to request workloads in terms of the amount of vCPU and memory. In the latter case, CLOUDFACTORY applies an operational search approach where vCPU allocation is maximized while respecting constraints on the specified amount but also on the VM configurations distribution considered.

Each VM is randomly assigned a usage profile that conforms to the proportions reported by the workload analyzer.

If the arrival rate for a given profile is not zero, additional VMs is generated after the initial set is deployed, following a heavy-tail Gaussian to match the real pattern observed [4].

During this process, a lifespan is attributed to each VM so that the number of VMs departures on each scope meets the targeted departure ratio of each profile.

B. On VM behaviour generation

From the profile assigned to each VM, an activity workload must be generated. Computed profiles include bounds on average and percentiles values. A random selection is made within the specified limits for both the targeted average value and percentile value of the CPU. Therefore, a Gaussian distribution that aligns with these two specific characteristics is generated. Gaussian distributions were chosen for their similarities with VM CPU usage [11].

For a given VM, the obtained distribution is considered as a list of potential CPU usage. Based on experiment temporality, values from the distribution are selected and assigned to each slice as its targeted CPU usage. For VMs that exhibit periodic behavior, the values attributed to the first scope are repeated on the subsequent scopes until the VM reaches the end of its lifecycle.

C. A few words on reproducibility

CLOUDFACTORY provides the ability to dump and load an experiment in JSON format, allowing researchers to reproduce the same workload on different systems or temporality settings.

This feature is particularly useful in peer review processes, as the JSON dump can be shared along with the generated scenario scripts to facilitate communication on the evaluation process.

VI. EXPORTERS

CLOUDFACTORY can currently generate 3 export types, leading to 3 possible bindings.

A. CLOUDSIMPLUS

1) Overview: In a simulation-based experiment, a CLOUDSIMPLUS scenario can be generated as a Java program that incorporates the VM workloads. While this generated Java template file is readily usable, researchers may wish to customize it to evaluate specific clusters (e.g., number of data centers, number of nodes, size of nodes) or algorithms (e.g., schedulers).

2) Implementation details: CLOUDSIMPLUS uses various object representations to simulate the execution of a Cloud environment. VM objects are instantiated according to generated configuration. Internal applications are modeled using Cloudlet objects. We create a per-VM Cloudlet implementing a custom utilization model loaded from a Java properties file. To account for VM lifespan, a maximum Cloudlet duration is specified, and a broker destroys VM without any Cloudlet being assigned to it. The submission delay feature is used to manage arrival rates.

B. Bash

1) Overview: In the case of a physical testbed, our tool can generate VM workloads in the form of libvirt-based bash scripts.

CLOUDFACTORY outputs VMs deployment script jointly with a VMs workloads script to deliver a reproducible behaviour.

To mimic the heterogeneity of the Cloud workload, different images with different applications can be considered. The generated scripts transform targeted CPU levels into benchmarking tool parameters using editable configuration files. Additionally, constraints on VM distribution can be expressed at the generation level to prevent, for instance, memory-intensive benchmarks from being deployed on lightweight VM configurations.

Although this bash exporting option requires an initial setup involving the use of pre-generated qcow2 images with SSH keys already injected, it remains a simple way to deploy a large number of VMs on a worker node.

The resulting workload scripts can be used both locally and remotely, with *Network Address Translation* (NAT) management offered with the latter option.

2) Implementation details: Various applications are used to represent the heterogeneity of the workload in the Cloud.

Idle VMs are characterized as workloads with no specific CPU activity, except for the guest operating system. Batch-oriented VMs simulate CPU-intensive workloads using the StressNG load test with variable CPU usage. Database workloads are represented with VMs hosting a POSTGRESQL instance serving a TPC-C workload where request rates are adapted. Static websites are represented with a Wordpress instance serving various request rates using wrk2 benchmarking tool. Microservices architectures are also considered using DEATHSTARBENCH [18], a social network serving various requests rates using wrk2 benchmarking tool.

The reports obtained from the various benchmarking tools are stored as text files. To facilitate the exploration of application performance, we provide bash scripts that can generate a CSV format from these reports.

C. CBTOOL

1) Overview: To facilitate more complex deployments, the IBM CBTOOL framework can be used. From the generated CBTOOL script, VM workloads can be deployed on various Cloud environments, including GCP, AWS and OpenStack clusters, with proper framework configuration.

2) Implementation details: The notion of a CBTOOL virtual application is used to model VMs and their associated load levels. Prior to deploying any new VM, a baseline virtual application is customized to match the desired load level, load duration, and VM size. CLOUDFACTORY slices are represented using "`waitfor`" instructions, with VM deployments and destructions at each occurrence to simulate the departure and arrival rate obtained from the modeling phase. The generated script is configured for the CBTOOL simulation mode, making it ready to use. It can be adapted by developers to their deployment context.

D. Others exporters

CLOUDFACTORY has been designed with compatibility considerations in mind. Others formats can be implemented through custom exporters, typically written in about 200 lines of code.

To ensure maximum compatibility with other formats, exporters should utilize the CPU usage list generated by CLOUDFACTORY instead of custom solution models. This approach helps to maintain homogeneity across different scenarios.

VII. CASE STUDY

While CLOUDFACTORY can be used for multiple purposes, we demonstrate its added value to study cluster sizing issues, given multiple workloads characteristics. Cloud physical resources remain underused, hence severely impacting the cost and efficiency of these infrastructures at large [4, 34]. To overcome this inefficiency, IAAS providers usually compensate for oversized VMs by offering more virtual resources than are physically available on a host. However, studying oversubscription strategies implies considering realistic VM usage. In this section, we highlight the distribution impact on the sizing of a IAAS cluster, based on the study of Chameleon and Microsoft Azure datasets.

A. Generate distributions

Chameleon-like and an Azure-like distributions were computed using the workload analyzer from each dataset. The Azure 56 GB memory configuration, which represents 2% of VMs, has no equivalent in the Chameleon dataset. It was disregarded to facilitate comparisons. The computed distribution is detailed in Tables II and III.

TABLE II
CPU DISTRIBUTION USED FOR EXPERIMENT

vCPU	Azure	Chameleon
1	**51%**	26%
2	25%	20%
4	17%	12%
8	7%	**42%**

TABLE III
MEMORY DISTRIBUTION USED FOR EXPERIMENT

vRAM (GB)	Azure
0.75	4%
1.75	**47%**
3.50	25%
7.00	16%
14.00	8%

vRAM (GB)	Chameleon
0.5	2%
2.0	24%
4.0	20%
8.0	12%
16.0	**42%**

B. Generate usage profiles

As the Chameleon dataset does not contain usage metrics, Azure ones—computed from four usage profiles inferred from workload analyzer—were applied to both contexts. Furthermore, two deployment strategies were tested: a *static* and a *dynamic* deployment. In the static deployment case, all profiles were assigned arrival and departure ratios equal to `0`. The initial set of VMs is therefore deployed immediately, and run on all experiment duration, with no new VMs arriving. In the case of dynamic deployment, the computed departure and arrival ratios are left unchanged for each profile. Once the initial set of VMs is deployed, some of them are leaving and others arrive continuously, while maintaining a similar overall quantity. In this scenario, the provisioning capabilities were evaluated over a one-week window.

C. Experiment

Considering a hosting target of VMs, and given a VM distribution and deployment type, we compute the minimal amount of servers required to host the targeted number of VMs. To do so, the workload generator is used to obtain a CLOUDSIMPLUS workload for each combination, using a server baseline of 64 CPUs and 256 GB. The configuration baseline was chosen due to the prevalence of 64-core servers in at least one Cloud dataset [32] and an assumption that

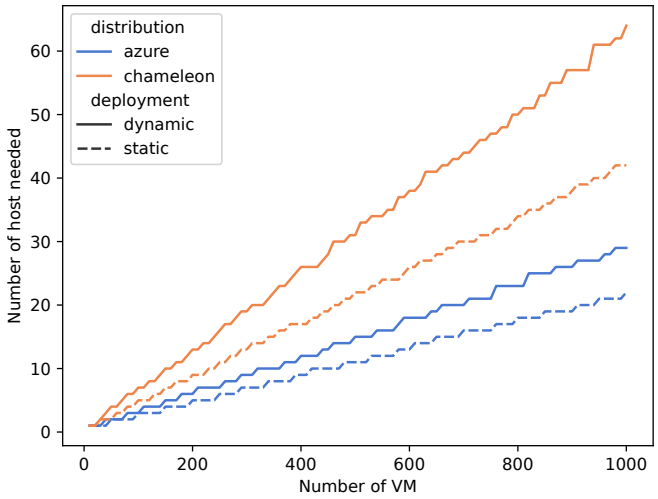

Fig. 3. IAAS hosting capacities required by VM distribution and deployment conditions

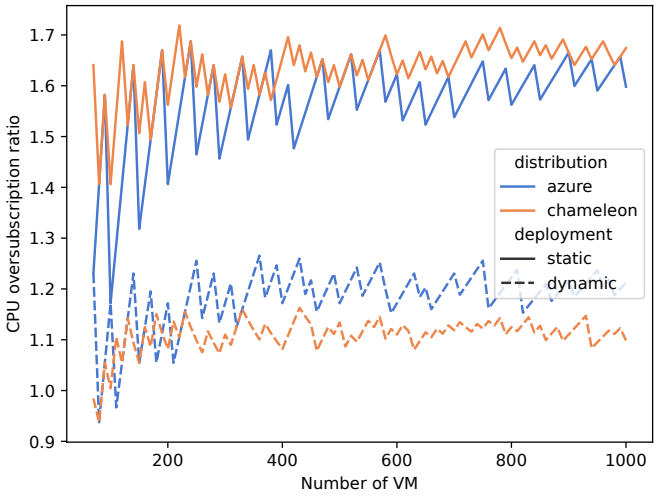

Fig. 4. CPU oversubscription ratio based on VM distribution and deployment condition

an allocation of 4 GB memory per core would be representative. Given a distribution and deployment type combination, CLOUDSIMPLUS simulations are run sequentially, with an increasing number of hosts, until the minimum amount of servers required is found. This is done by verifying that the default VM scheduler was able to schedule all the provisioned VMs. CPU oversubscription was supported in this experiment through CLOUDSIMPLUS MIPS notion, by implementing a per-VM MIPS baseline weighted by their maximum usage, allowing the VMs scheduler to allocate resources based on VM usage instead of VM provisioned resources. This mechanism can be described as an oracle-based oversubscription mechanism, as maximum usage is known prior to deployment. However, it remains pessimistic, as maximum usage of VMs is unlikely to happen at the same time, leading to potential gains, non-exploited here. A more realistic approach requires the deployment of a custom VM scheduler, which is out of the scope of this paper. We did not consider live migration in this experiment.

D. Results

Results are shown in Figure 3. Compared to the static deployment, the flexibility required by a fluctuating number of provisioned VMs implies a significant overhead for both datasets, on average 1.31 more servers for Azure and 1.46 for Chameleon. While being relatively stable on an increasing hosting objective, this factor underlines that heavy-tail arrival rates, reflecting grouped VMs deployments, impose an infrastructure cost. Conversely, in a system where the arrival rate can be managed based on, for example, a queuing mechanism, the cost associated with the same VM hosting objective is lower. Cloud providers can leverage this by offering their customers an incentive to choose a queued deployment instead of the state-of-the-art "as fast as possible" strategy.

The Chameleon distribution requires VMs with larger amounts of resources, on average. Therefore, the same VMs hosting objective results in provisioning more resources, compared to the Azure distribution. Concretely, an average deployment requires about 2.25 vCPU and 4 GB of memory in the Azure context versus 4.5 vCPU and 9 GB for Chameleon. Chameleon needs on average 1.89 additional hosts for static deployments and 2.1 additional hosts for dynamic deployments, due to the amplified impact of VMs arrival bursts on newly provisioned resources.

The achieved CPU oversubscription ratio, defined as the ratio between the vCPUs offered and those physically available, can also be estimated. Provisioned CPUs are extrapolated from the average deployment size applied to the number of VMs. Physical resources are computed from the number of hosts and their respective configurations. In Figure 4, one can observe that, while being higher than 1.0, oversubscription ratios remain significantly lower for dynamic deployments, compared to static ones. While dynamic deployments have the same VM hosting objective, the number of hosted VMs may be significantly higher at a given point, due to fluctuation in departure and arrival rate, reducing oversubscription to almost no gain on average (1.11 ratio for Chameleon, 1.18 for Azure). Due to the same usage being applied during CLOUDFACTORY workload generation, oversubscriptions ratios are similar on static deployments (1.63 and 1.57 for Chameleon and Azure respectively).

The key results of the experiment can be seen in Table IV.

E. Adoption by the Cloud industry

Beside generating realistic IAAS workloads for Cloud experiments, CLOUDFACTORY aims to foster sharing of high-level IAAS statistics to and from the Cloud ecosystem. In

TABLE IV
EXPERIMENT RESULTS SUMMARY

Infrastructure Configuration	Servers to host 1,000 VMs	CPU oversubscription ratio
Static Azure	22	1.57
Dynamic Azure	29	1.18
Static Chameleon	42	1.63
Dynamic Chameleon	64	1.11

particular, OVHcloud[2] is a public Cloud provider, being recognized as the largest Cloud operator in Europe [35].

Thanks to the availability of the workload analyzer of CLOUDFACTORY, OVHcloud accepted to share the statistics of their *public cloud* offer. Interestingly, we can compare these statistics with the ones of alternative providers, like Azure and Chameleon. Figures 5 and 6 are, thus, reporting on the distributions of vCPU and vRAM, respectively. Due to the diversity of configurations, we grouped some close vRAM configurations.

The findings from the VM distribution analysis suggest statistically significant differences among the three Cloud service providers.

Specifically, Azure exhibits a light CPU context in comparison to both Chameleon and OVHcloud. The configuration options that account for less than 1% have been excluded from the analysis. It is worth noting that the OVHcloud distribution includes the largest configuration option, with 16 vCPU (3% of VMs). Conversely, the Chameleon Cloud service provider records the highest average CPU option, reflecting its research-oriented objective. This outcome is attributed to the 8 vCPU option, which is the most prevalent among the Chameleon cluster VMs.

On the memory aspect, Azure is also predominantly characterized by small configurations. While Chameleon is a memory intensive Cloud workload compared to Azure, yet it is exceeded by OVHcloud, primarily due to its percentage of options equal to or greater than 30 GB (8% of VMs). In contrast, Azure's most significant memory option is the 56 GB configuration, which accounts for only 2% of the total VMs. Notably, OVHcloud does not offer any memory options less than 2 GB.

The three distributions are available in our repository and can directly be used by our workload generator. CLOUDFACTORY generated usage metrics for OVHcloud will also be available in our repository in a near future, as an alternative to Azure set of statistics.

VIII. LIMITATIONS

CLOUDFACTORY aims to capture and reproduce global behaviors observed in IAAS environments. However, the use of statistical techniques introduces an inherent information loss, potentially leading to the omission of specific behaviors. In particular, our modeling of CPU usage patterns is based on

[2]https://www.ovhcloud.com/

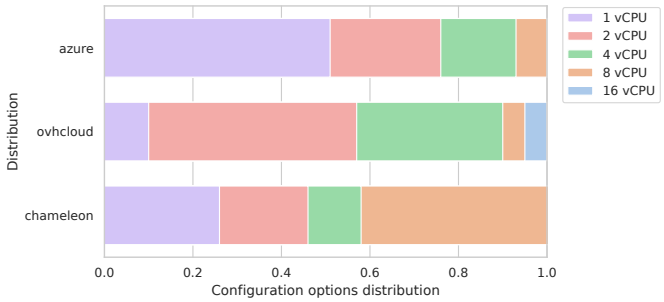

Fig. 5. Comparison of vCPU distributions across Cloud providers

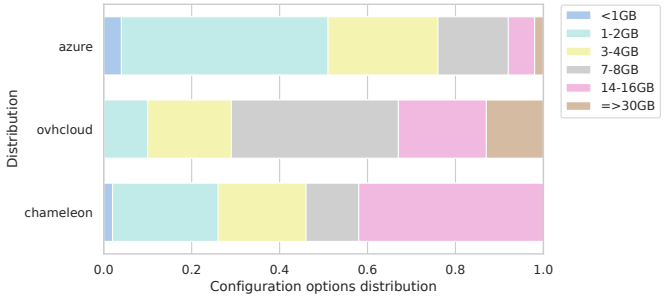

Fig. 6. Comparison of vRAM distributions across Cloud providers

Gaussian distributions, which may not accurately represent the behavior of all VMs under real-world conditions.

While we consider grouped deployments by utilizing a heavy-tail distribution to capture arrival rates, this subsequently treats individual VMs as independent entities. However, in reality, sets of VMs can be provisioned to host distributed applications, resulting in network traffic between them.

Furthermore, CLOUDFACTORY's usage profiling focuses on CPU usage and neglects other types of resources. This decision was made for two main reasons: the widespread availability of CPU usage in datasets and the presumed correlation between CPU usage and the utilization of other resources, based on the specific application being used. However, we acknowledge that certain benchmarks could benefit from incorporating information about the usage of other computing resources.

We offer the capability to scale a IAAS workload down to a single node. However, this functionality relies on the assumption that the workload is evenly distributed among nodes, which may not always hold in practice. For example, Cloud providers may manage dedicated clusters for different premium levels or service tiers.

IX. CONCLUSION

In this paper, we introduced CLOUDFACTORY, a realistic IAAS workload generator. We first provide a Cloud dataset analyzer, able to compute context-based metrics on VM configuration options distribution and usage. These statistics can

be shared with the wider community, enabling researchers to consider multiple Cloud-based contexts. They also can be used by our generator to compute a workload scenario in a simulated context, leveraging CLOUDSIMPLUS capabilities, or using a physical testbed using bash or CBTOOL deployments. A case study example was presented with the cluster sizing example, underlying the importance of considering representative workload to produce coherent results.

As future work, we would like to consider other VM metrics on deployments, such as disk and network activities, which were not included in this version, due to the lack of available datasets. We would also like to include more complex VM deployments, where instances can jointly work. This perspective may allow CLOUDFACTORY to consider container-based Cloud solutions by permitting, for example, Kubernetes deployments.

Toolkit availability

CLOUDFACTORY is an open-source toolkit written in Python.[3] Clustering is implemented using the scikit-learn library, while periodicity signal analysis is based on SciPy. All the experiments generated for our case study are made available in JSON format in our repository.

ACKNOWLEDGMENTS

This work is supported by the "*FrugalCloud*" Inria and OVHcloud partnership, and the ANR Distiller grant (ANR-21-CE25-0022).

REFERENCES

[1] J. Byrne, S. Svorobej, K. M. Giannoutakis, D. Tzovaras, P. J. Byrne, P.-O. Östberg, A. Gourinovitch, T. Lynn *et al.*, "A review of cloud computing simulation platforms and related environments." *CLOSER*, pp. 651–663, 2017.

[2] N. Mansouri, R. Ghafari, and B. M. H. Zade, "Cloud computing simulators: A comprehensive review," *Simulation Modelling Practice and Theory*, vol. 104, p. 102144, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X20300836

[3] S. M. Ali and G. Kecskemeti, "Sequal: an unsupervised feature selection method for cloud workload traces," *The Journal of Supercomputing*, pp. 1–19, 2023.

[4] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms," in *SOSP*. ACM, 2017, pp. 153–167.

[5] M. C. Silva Filho, R. L. Oliveira, C. C. Monteiro, P. R. M. Inácio, and M. M. Freire, "Cloudsim plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 400–406.

[3]https://github.com/jacquetpi/cloudfactory/

[6] M. B. Chhetri, S. Chichin, Q. B. Vo, and R. Kowalczyk, "Smart cloudbench – automated performance benchmarking of the cloud," in *2013 IEEE Sixth International Conference on Cloud Computing*, 2013, pp. 414–421.

[7] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms," *Softw. Pract. Exper.*, vol. 41, no. 1, p. 23–50, jan 2011.

[8] S.-H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. R. Das, "Mdcsim: A multi-tier data center simulation, platform," in *2009 IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1–9.

[9] K. Anusuya and V. Krishnapriya, "Green cloud:"a pocket-level simulator with on-demand protocol for energy-aware cloud data centers," *International Journal of Science and Research (IJSR)*, vol. 3, no. 2, 2014.

[10] H. Casanova, A. Legrand, and M. Quinson, "Simgrid: A generic framework for large-scale distributed experiments," in *Tenth International Conference on Computer Modeling and Simulation (uksim 2008)*. IEEE, 2008, pp. 126–131.

[11] A. Lebre, J. Pastor, and M. Südholt, "Vmplaces: A generic tool to investigate and compare vm placement algorithms," in *Euro-Par 2015: Parallel Processing: 21st International Conference on Parallel and Distributed Computing, Vienna, Austria, August 24-28, 2015, Proceedings 21*. Springer, 2015, pp. 317–329.

[12] M. Silva, M. R. Hines, D. Gallo, Q. Liu, K. D. Ryu, and D. d. Silva, "Cloudbench: Experiment automation for cloud environments," in *Proceedings of the 2013 IEEE International Conference on Cloud Engineering*, ser. IC2E '13. USA: IEEE Computer Society, 2013, p. 302–311. [Online]. Available: https://doi.org/10.1109/IC2E.2013.33

[13] B. Sun, B. Hall, H. Wang, D. W. Zhang, and K. Ding, "Benchmarking private cloud performance with user-centric metrics," in *2014 IEEE International Conference on Cloud Engineering*, 2014, pp. 311–318.

[14] T. P. P. Council, "Tpc benchmark c standard specification," 1990.

[15] C. SPEC. (2017) Spec cpu® 2017. [Online]. Available: https://www.spec.org/cpu2017/

[16] M. SPEC. (2009) Specmail2009. [Online]. Available: https://www.spec.org/mail2009/

[17] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, A. Fox, and D. Patterson, "Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0," in *Proc. of CCA*, vol. 8. Citeseer, 2008, p. 228.

[18] Y. Gan, Y. Zhang, D. Cheng, A. Shetty, P. Rathi, N. Katarki, A. Bruno, J. Hu, B. Ritchken, B. Jackson, K. Hu, M. Pancholi, Y. He, B. Clancy, C. Colen, F. Wen, C. Leung, S. Wang, L. Zaruvinsky, M. Espinosa, R. Lin, Z. Liu, J. Padilla, and C. Delimitrou, "An Open-Source Benchmark Suite for Microservices and Their Hardware-

Software Implications for Cloud & Edge Systems," in *ASPLOS*. ACM, 2019, pp. 3–18.

[19] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proceedings of the 1st ACM Symposium on Cloud Computing*, ser. SoCC '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 143–154. [Online]. Available: https://doi.org/10.1145/1807128.1807152

[20] M. Rak and G. Aversano, "Benchmarks in the cloud: The mosaic benchmarking framework," in *2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. IEEE, 2012, pp. 415–422.

[21] G. Aversano, M. Rak, and U. Villano, "The mosaic benchmarking framework: Development and execution of custom cloud benchmarks," *Scalable Computing: Practice and Experience*, vol. 14, no. 1, pp. 33–46, 2013.

[22] M. Cunha, N. Mendonca, and A. Sampaio, "A declarative environment for automatic performance evaluation in iaas clouds," in *2013 IEEE Sixth International Conference on Cloud Computing*. IEEE, 2013, pp. 285–292.

[23] N. Yigitbasi, A. Iosup, D. Epema, and S. Ostermann, "C-meter: A framework for performance analysis of computing clouds," in *2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009, pp. 472–477.

[24] B. Varghese, L. T. Subba, L. Thai, and A. Barker, "Container-based cloud virtual machine benchmarking," in *2016 IEEE International Conference on Cloud Engineering (IC2E)*, 2016, pp. 192–201.

[25] SPEC. (2021) Specvirt datacenter 2021. [Online]. Available: http://spec.org/cloud_iaas2018/

[26] V. Makhija, B. Herndon, P. Smith, L. Roderick, E. Zamost, and J. Anderson, "Vmmark: A scalable benchmark for virtualized systems," Technical Report TR 2006-002, VMware, Tech. Rep., 2006.

[27] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "Towards understanding cloud performance tradeoffs using statistical workload analysis and replay," *University of California at Berkeley, Technical Report No. UCB/EECS-2010-81*, 2010.

[28] E. F. Boza, C. San-Lucas, C. L. Abad, and J. A. Viteri, "Benchmarking key-value stores via trace replay," in *2017 IEEE International Conference on Cloud Engineering (IC2E)*, 2017, pp. 183–189.

[29] J. Yin, X. Lu, X. Zhao, H. Chen, and X. Liu, "BURSE: A bursty and self-similar workload generator for cloud computing," *IEEE Trans. Parallel Distributed Syst.*, vol. 26, no. 3, pp. 668–680, 2015.

[30] I. SPEC. (2018) Spec cloud iaas 2018. [Online]. Available: http://spec.org/cloud_iaas2018/

[31] J. Mambretti, J. Chen, and F. Yeh, "Next generation clouds, the chameleon cloud testbed, and software defined networking (sdn)," in *2015 International Conference on Cloud Computing Research and Innovation (ICCCRI)*, 2015, pp. 73–79.

[32] Alibaba. (2018) The alibaba clusterdata201708 trace data. [Online]. Available: https://github.com/alibaba/clusterdata

[33] Google. (2019) Borg cluster traces. [Online]. Available: https://github.com/google/cluster-data

[34] N. Bashir, N. Deng, K. Rzadca, D. Irwin, S. Kodak, and R. Jnagal, "Take It to the Limit: Peak Prediction-Driven Resource over-committment in Datacenters," in *EuroSys*. ACM, 2021, p. 556–573.

[35] OVHcloud. (2023) Who are we? [Online]. Available: https://www.ovhcloud.com/en/about-us/