Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A one-year measurement of quota-induced waste in private clouds

Paola Marques
Universidade Federal de Campina Grande
Francesco Desimoni
francesco.desimoni@uniupo.it
Università degli Studi del Piemonte Orientale “Amedeo Avogadro”
Francisco Brasileiro
Universidade Federal de Campina Grande
Massimo Canonico
Università degli Studi del Piemonte Orientale “Amedeo Avogadro”
Thiago Emmanuel Pereira
Universidade Federal de Campina Grande

Research Article

Keywords: Private cloud, Quotas, Resource hoarding, Resource inefficiency

Posted Date: November 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11484189/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A one-year measurement of quota-induced waste in private clouds

Paola Marques[1], Francesco Desimoni[2*], Francisco Brasileiro[1], Massimo Canonico[2], Thiago Emmanuel Pereira[1]

[1]Department of Computing and Systems, Federal University of Campina Grande, Av. Aprígio Veloso, 882, Campina Grande, 58109-970, PB, Brazil.

[2*]Department of Sciences and Technological Innovation, University of Piemonte Orientale, Viale Teresa Michel, 11, Alessandria, 15121, Italy.

*Corresponding author(s). E-mail(s): francesco.desimoni@uniupo.it;
Contributing authors: paola.kathrein.marques@ccc.ufcg.edu.br; fubica@computacao.ufcg.edu.br; massimo.canonico@uniupo.it; temmanuel@computacao.ufcg.edu.br;

Abstract

In contrast to public clouds, where pricing regulates resource access, private clouds rely on administrative constraints such as quotas. Typically, users pay nothing for allocated resources; the quota simply caps how much they can provision. Because quotas are oversold in practice, it is unclear whether physical capacity or the quota itself limits provisioning. We analyze one year of operational data from a private cloud, comprising quota assignments, per-project allocation, and per-virtual-machine (VM) utilization sampled every five minutes, totaling over 64 million records. We find this mechanism prone to inefficiencies. Quota exhaustion is a persistent constraint independent of available capacity: at least half of projects cannot provision even the smallest available VM for some period; about 14% remain in this state throughout their entire observed periods, and approximately 27% remain quota-bound even when aggregate memory allocation is between 70% and 80% of physical capacity. Compounding this, overall VM usage is very low: measured by virtual CPU usage, VMs are idle more than twice as long as they are meaningfully used, and although about 50% reach first meaningful use on the day they are created, a long tail remains unused for weeks. We argue this is not coincidental: users request quotas sized for peak demand,

while administrators oversell capacity, betting that not everyone peaks simultaneously. With no guarantee of obtainability, users provision in advance and leave resources idle until needed. We conclude by discussing the incentive structure that plausibly sustains this gap and its implications for quota design in private clouds.

Keywords: Private cloud, Quotas, Resource hoarding, Resource inefficiency

1 Introduction

Cloud providers must impose limits to prevent a single client from monopolizing the available infrastructure. Both public and private clouds address this issue by restricting the amount of resources that a single client can allocate simultaneously—although the notion of a client varies across environments (e.g., in OpenStack-based clouds, a client corresponds to a project, an organizational unit that groups users and VMs).

These limits are typically enforced through quotas, which can be defined either with or without overbooking. In a conservative approach, quotas are set so that the sum of all quotas does not exceed the total physical resources available. In contrast, an aggressive approach allows overbooking, in which case the aggregate quotas exceed the underlying physical capacity.

When quotas are defined without overbooking, they effectively act as resource reservations: as long as a client has enough quota remaining, it is guaranteed to allocate resources when requested. However, this approach increases the likelihood that clients will reach their quota limits, potentially leading to inefficient infrastructure utilization. In such cases, a client may be denied additional resources even though capacity is available.

Public clouds mitigate these inefficiencies by combining overbooked quotas with multiple allocation models. Some of these models allow clients to reserve resources in

advance, guaranteeing availability when needed. As a result, quotas prevent monopolization, overbooking reduces the likelihood of clients exhausting their quotas, and reservation mechanisms provide guarantees for critical workloads.

In contrast, middleware platforms for managing private clouds, such as OpenStack, typically do not offer native resource reservation mechanisms. In these environments, the only way to guarantee resource availability is to adopt conservative, non-overbooked quotas. This approach, however, increases the risk of underutilization and makes quota management more complex, as quotas must be continuously adjusted to match changing client demands.

Using overbooked quotas in private clouds alleviates some of these issues, but it comes at the cost of losing guarantees that resources will be available when needed. Additionally, private clouds often lack billing or pay-as-you-go models, which further exacerbates inefficiencies. Since clients are not charged for resource usage, they are incentivized to allocate resources as early as possible—even when not immediately needed—and to retain them after use in anticipation of future demand. In practice, this behavior turns early allocation into a *de facto* reservation mechanism, contributing to persistent resource hoarding and reduced overall efficiency.

In this paper, we investigate and measure the inefficiency of quotas in private clouds by analyzing the data collected from a private cloud infrastructure used by researchers from the Federal University of Campina Grande (UFCG), Brazil. Our measurements quantify the extent of these inefficiencies. At the cluster level, full RAM saturation occurs on only 3.1% of days. Yet at least half of the projects experience periods in which their remaining quota is insufficient to provision even the smallest VM, and about 14% remain in this situation throughout their entire observed periods. Overall, approximately 38% of the provisioned RAM quota remains unallocated. Inefficiency also extends to resources that have already been allocated. VMs spend more than

twice as much time idle as they do in meaningful use, and although approximately 50% reach their first meaningful use on the day of creation, some remain unused for weeks.

Overall, the main contributions of this work are: i) a one-year longitudinal characterization of a production private cloud, including not only host utilization but also per-project allocation; ii) the definition of potential failure and the finding that quota exhaustion persists even when there is available capacity; and iii) the quantification of resource hoarding, including idleness across the VM lifetime and the time until first use. The rest of the paper is organized as follows: in Section 2, we discuss a selection of previous works related to ours; Section 3 describes the infrastructure of our system and how we collected the data; in Section 4, we present the methodology and results of our work, further discussed in Section 5. Finally, Section 6 provides concluding remarks and outlines future research directions.

2 Related work

Resource waste in cloud computing environments has attracted considerable attention from the research community, with most contributions approaching the problem from either workload characterization or resource optimization perspectives. Below, we survey the most relevant works, paying particular attention to evidence of how much allocated cloud capacity remains idle or underutilized over time.

On public large-scale clusters, Reiss et al. [1] analyzed the first publicly available trace from a large Google compute cluster (over 12,500 machines, 29 days). Their analysis observed under-utilization despite over-commitment of resources, difficulties in scheduling high-priority constrained tasks, and a lack of dynamic adjustment to user allocation requests. The study revealed significant heterogeneity in resource demand and flagged the consequent inefficiency of traditional slot-based schedulers, establishing a foundational reference point for subsequent measurement studies. Building on this, Cortez et al. [2] at Microsoft introduced Resource Central, characterizing Azure's

VM workload across distributions of VM lifetime, deployment size, and resource consumption, and demonstrating that certain VM behaviors are fairly consistent over multiple lifetimes, making history an accurate predictor of future behavior. The implication was that a substantial share of allocated resources follows predictable low-utilization patterns that are exploitable for reclamation.

More recently, Reidys et al. [3] performed an extensive characterization of over one million opaque VMs across ten Azure clusters and seven regions. The traces were collected at 5-minute intervals and covered resource allocation, deallocation times, and maximum utilization for CPU, memory, network, and storage, establishing a lower bound for underutilization. The study confirmed that the CPU is usually the most underutilized resource and that many VMs exhibit complementary temporal patterns — some peaking at noon, others at night — which are predictable and can be harnessed by cloud platforms to colocate oversubscribed VMs.

In private and enterprise clouds, Khan et al. [4] studied workload characterization in a production private cloud supporting hundreds of enterprise customers, each running between 100 and 5,000 VMs. The study found that workloads on these VMs are often driven by repeatable business behavior and exhibit temporal and spatial correlations, including time-of-day effects, while configuration details for customer VMs are often unavailable to the operator. This opacity over actual resource consumption is precisely what makes measuring waste in private cloud settings particularly difficult. Along similar lines, a 2025 empirical study on the SAP Converged Cloud [5] — a private managed cloud based on OpenStack, introduced in 2017 — analyzed scheduling and placement over a real production environment for 30 consecutive days. The data revealed that some compute nodes showed consistently high CPU availability, suggesting underutilization, while others exhibited potential resource contention. This imbalanced distribution of workload within the data center may be intentional, depending on the chosen scheduling algorithm. The authors further noted that most

VMs appear to be overprovisioned and consequently consume only a fraction of the allocated CPU resources.

Guo et al. [6] analyzed an 8-day production trace from Alibaba's co-located cluster of 4,034 machines and revealed that latency-critical applications reserved massive CPU and memory resources in case of performance spikes, and that static memory partitioning prevented stranded resources from being reallocated to other jobs, greatly affecting the resource efficiency of the datacenter. Additionally, batch-processing applications were restricted from harvesting available spare resources during the daytime, even though remarkable spare capacity existed. Together, these findings document a structural form of resource idleness that persists for extended intervals, driven by conservative allocation policies rather than actual demand.

A complementary data-driven study by Everman et al. [7] analyzed Microsoft Azure 2019 traces covering nearly 2.7 million VMs created by 6,687 users over 30 days. The results revealed that a large portion of VMs were under-utilized or over-provisioned for resources, and that 15% of users were classified as "red users" who habitually over-allocate, while around 36% were "green users" who utilize resources efficiently. The study introduced the concept of Cloud Waste Points as a quantitative measure of per-VM waste but focused on public IaaS rather than on the dynamics of privately managed project-based quota systems.

For idle VM identification in production, the CloudGC system [8] identified idle VMs by collecting and analyzing server-side metrics in a production cloud environment. The system flagged idle VMs and provided suggestions to their owners, and upon confirmation, the idle VMs were deleted or archived. However, the work targeted reactive cleanup of obviously idle resources and did not attempt a longitudinal quantitative study of how long allocated resources remain idle prior to detection, nor did it distinguish between resources that are idle because they belong to inactive projects and those that are idle due to scheduling inefficiency.

Concerning utilization measurement and VM consolidation, Beloglazov and Buyya [9] proposed energy-aware VM consolidation heuristics for cloud data centers, observing that idle VMs consume about 70% of their maximum power, making idleness costly not only in terms of capacity waste but also energy. Similarly, dynamic VM consolidation schemes [10] proposed detecting underloaded hosts through utilization prediction and switching them to a low-power state. However, they evaluated the magnitude of waste through simulation rather than through direct observation of real allocation-versus-usage gaps. Marshall et al. [11] addressed low utilization in IaaS clouds by proposing backfill VMs to exploit idle cloud nodes, reporting that utilization in an IaaS cloud increased from 37.5% to 100% during a portion of their evaluation trace, confirming substantial idle capacity in typical deployments.

In conclusion, despite the above contributions, the literature reveals a consistent limitation: most measurement and characterization studies focus on public cloud providers (Google, Azure, Amazon Web Services, Alibaba) or simulation environments and address resource underutilization at the infrastructure level — looking at physical host utilization — rather than at the allocation level, i.e., quantifying the gap between resources formally assigned to a project's quota and resources actually consumed over time. The private cloud context introduces additional complexity: resources are typically allocated to projects under long-running quotas negotiated with platform administrators, creating conditions in which a project may hold a large allocation for months or years while consuming only a fraction of it intermittently. To the best of our knowledge, a systematic and intensive empirical investigation focused specifically on quantifying how much allocated capacity remains idle — and for how long — within private cloud platforms managed under project-based quota models is still missing from the literature.

3 Infrastructure and Data Collection

This study was conducted using data collected over approximately 12 months, from May 2024 to May 2025 [12], from a private production cloud operated by the Distributed Systems Laboratory at the Federal University of Campina Grande, Brazil. The cloud is based on the OpenStack platform and serves multiple user groups with heterogeneous workloads. The cloud services include core Infrastructure as a Service (IaaS; that is, a cloud computing model providing virtualized computing resources—servers, storage, and networking over the internet) offerings such as compute, networking, storage, identity, and image management [13].

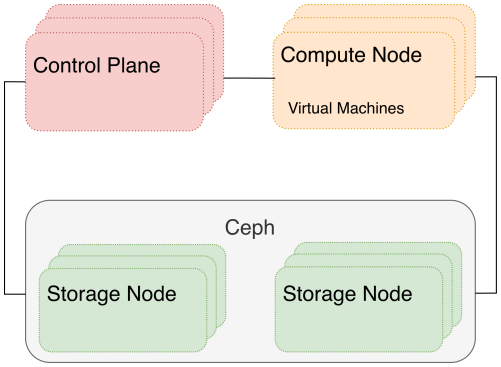


Fig. 1 Overview of a typical OpenStack-based private cloud architecture, including control-plane, compute, and storage (Ceph) components.

As shown in Figure 1, an OpenStack cloud is deployed on three different groups of machines: (1) control-plane nodes responsible for orchestration and management services; (2) compute nodes that host VMs; and (3) storage nodes providing persistent (block and blob) storage; in particular, we use *Ceph* [14] as the back-end storage method. The aggregated capacity of the computing nodes allows for the allocation of 1,024 GB of RAM, with a memory overcommit factor of 1, and 1,536 virtual CPUs (vCPUs), already accounting for a CPU provisioning overcommit factor of 8.

Data were obtained by querying a monitoring service using OpenStack APIs for metadata on projects, users, VMs, and predefined VM hardware configurations, as well as quota definitions and current resource allocations. Hypervisor-level VM utilization metrics were obtained from Prometheus endpoints exposed by the libvirt exporter. In total, the dataset contains more than 64 million records, 161 users, 34 projects, and 1,193 VMs. Users are authenticated identities that may belong to one or more projects. Provisioned VMs and other resources are associated with a project and with the user who created them.

The dataset describes how resources are provisioned and effectively allocated. This includes the quota of resources, such as RAM and CPUs, assigned to each project over time. For every quota assignment change, the dataset records a new entry. In addition to quota assignments, the dataset also captures the amount of resources effectively allocated within each project at regular 5-minute intervals, allowing for the analysis of how quotas are consumed over time.

At the VM level, the dataset contains resource utilization data. In addition to the resource capacity of the VM (the predefined hardware configurations that determine the number of vCPUs, the amount of memory, and the block storage attributes assigned at VM creation), we tracked VM utilization metrics as timestamped measurements of vCPU and memory usage sampled at 5-minute intervals.

The dataset uses native OpenStack UUIDs to identify users, projects, and VMs; their names and IP addresses are not included. Marques et al. [12] describe the collection methodology and privacy considerations in detail.

Although the data collection process operated almost continuously, minor interruptions occurred due to operational events such as scheduled maintenance or temporary monitoring system failures. These gaps account for approximately 0.39% of the total expected period, indicating near-complete temporal coverage [12].

4 Data analysis and results

This section presents the methodology used to process the data and the results of our study. We discuss how we pre-process the data (Section 4.1), then we analyze the two core problems investigated in this work: the quota exhaustion problem (Section 4.2) and the resource hoarding problem (Section 4.3). Each problem is analyzed through a set of metrics designed to capture different aspects of resource allocation and usage in the system.

4.1 Data and pre-processing

We analyze the two problems mentioned above at different granularities. The quota exhaustion study works at the project level, whereas the hoarding study works at the level of individual VMs, so each requires its own pre-processing of the dataset.

For the quota exhaustion analysis (Section 4.2), we use different project subsets depending on the metric. The dataset contains 34 projects with provisioned quota. One project ran its VMs entirely on a reserve host outside the production cluster under study and is not part of the analysis, leaving 33 projects for the characterization of provisioned quota. Four of these projects did not provision any VMs during the observation period. These projects are still included in the characterization of provisioned quota, but do not contribute to analyses that rely on VM allocation, since no allocation data exist for them. As a result, the potential-failure and quota-utilization analyses cover 29 projects.

The quota assignments are logged as events, with a new entry recorded only when a project's quota changes. To obtain a continuous per-project quota series aligned with the 5-minute allocation samples, we expand this sparse log onto the allocation timestamp grid using last-observation-carried-forward, starting from the first recorded snapshot. Summing across projects at each timestamp yields the cluster-level provisioned quota, which we normalize by physical capacity and reduce to daily resolution

by retaining its end-of-day value. We reconstruct the allocation series by summing, at each timestamp, the configured sizes of all VMs in each project. We then reduce this series to daily resolution using the same end-of-day convention and summarize each project's size as its mean allocated RAM over the study period.

Turning to the hoarding analysis (Section 4.3), we combine per-VM vCPU and RAM utilization sampled at 5-minute granularity with each VM's configured size, recovering the total vCPUs and RAM allocated to each VM. The raw vCPU samples typically range from zero to the vCPU count multiplied by 100, since they report the summed utilization across all vCPUs rather than a normalized figure. Therefore, we divide each sample by its vCPU count to obtain a per-VM utilization percentage. Usage values are normalized using the resources allocated to each VM at each point in time to account for the few cases where a VM was resized. A small fraction of normalized values still exceeds 100% (0.32% of rows, concentrated in a handful of VMs), and we treat these as missing. We then align each VM's series to a regular 5-minute grid, rounding timestamps to the nearest mark, collapsing any duplicates to their mean, and inserting empty rows for absent samples. We fill the resulting gaps, together with the cross-VM gaps noted in Section 3, by linear interpolation, but only when they span at most 4200 seconds (70 minutes, or 14 consecutive samples). This limit covers 92.77% of all gaps, and doubling it to 8400 seconds would raise coverage to only 96.13%, so we keep the lower value to limit the number of synthetic samples introduced. The samples filled by interpolation amount to 0.28% of the entire preprocessed dataset. Longer gaps are left unfilled, and the resulting time deltas between consecutive rows act as natural breakpoints when segmenting each VM's timeline into usage blocks.

4.2 The quota exhaustion problem

To understand the factors that limit resource provisioning, we analyze whether the inability of projects to allocate additional resources is driven by physical scarcity or by

static project-level quotas. In particular, we assess whether provisioning constraints emerge even when physical capacity remains available at the cluster level.

In OpenStack-based private clouds, resource allocation is controlled through project-level quotas, which define upper limits on the resources a project can provision across different dimensions, such as compute, storage, and networking. This work focuses on compute quotas, specifically vCPU and memory. Figure 2 shows the distribution of these quotas across projects, normalized by the smallest assigned quota. Quotas vary substantially across projects; the largest project has 40 times as much quota as the smallest one. Most projects (29 of 33) have quotas that preserve the 2 GB of RAM per vCPU ratio of the smallest VM configuration. Although users may request quota increases in arbitrary amounts, the provider typically assigns project quotas as multiples of this VM configuration.

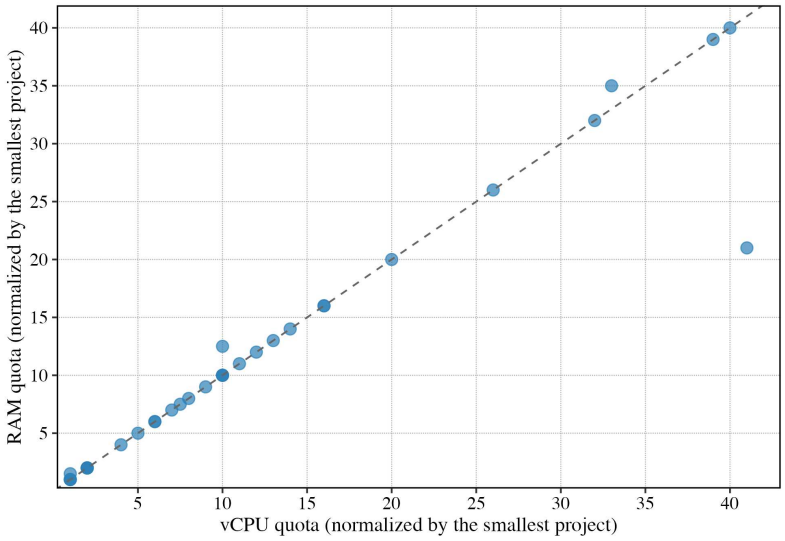


Fig. 2 Distribution of assigned project-level quotas for vCPU and RAM in the studied private cloud.

As Figure 3 shows, the physical cloud capacity is oversold. The sum of the provisioned (not necessarily allocated) RAM and CPU quotas is higher than the physical

cloud capacity throughout the entire observation period. Cloud overselling allows administrators to grant quotas whose sum exceeds physical capacity, based on the assumption that not too many projects are likely to allocate their entire quota at the same time. Additionally, it is important to note that there is a trend to increase the provisioned quotas as new projects join the system. Therefore, since the quotas are greater than capacity, the resource allocation might lead to unwanted consequences. First, even though it is doable, when the allocation of CPU quotas exceeds the physical cloud capacity, CPU contention may occur. Second, and more critically, when aggregate allocated RAM reaches the available capacity, no more VMs can be allocated since memory is not overbooked.

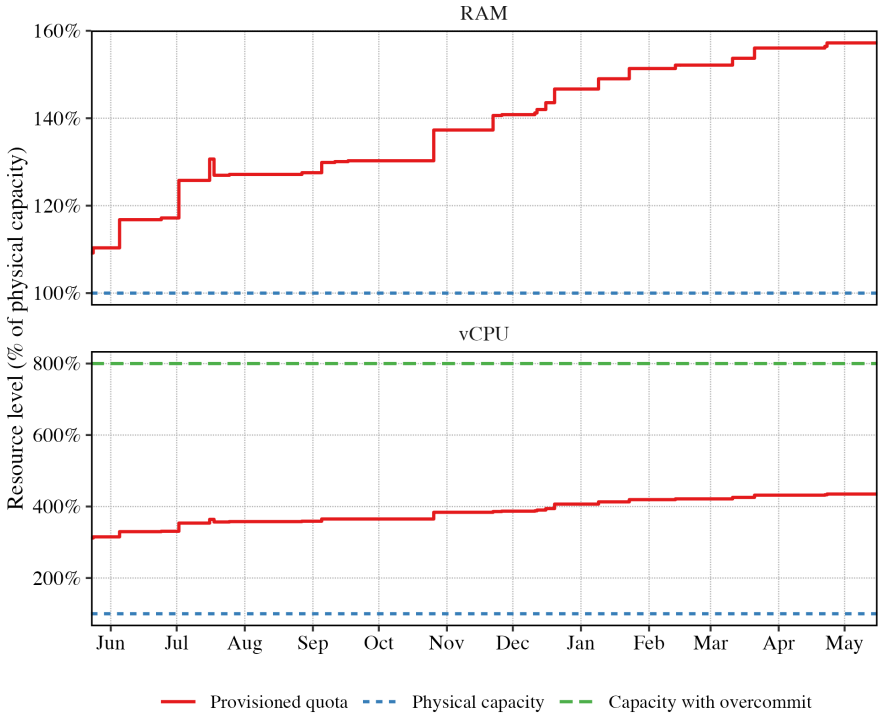


Fig. 3 Aggregate quotas normalized by physical capacity.

While project-level quotas define allocation limits, the effective use of infrastructure is determined by how the resources are allocated in running VMs. Figure 4 shows the aggregate allocated RAM and vCPU (solid lines) over the observation period, normalized by the cloud total physical capacity (dotted lines). The dashed line indicates the capacity under the configured CPU overcommit factor. Full physical saturation of RAM (at or above 100%) occurs on only 3.1% of the days. Memory allocation remains below 95% of physical capacity on 70% of the days. CPU allocation remains consistently below the configured overcommit limit throughout the observation period. Together, these results indicate that, most of the time, there is room for resource allocation despite capacity overselling.

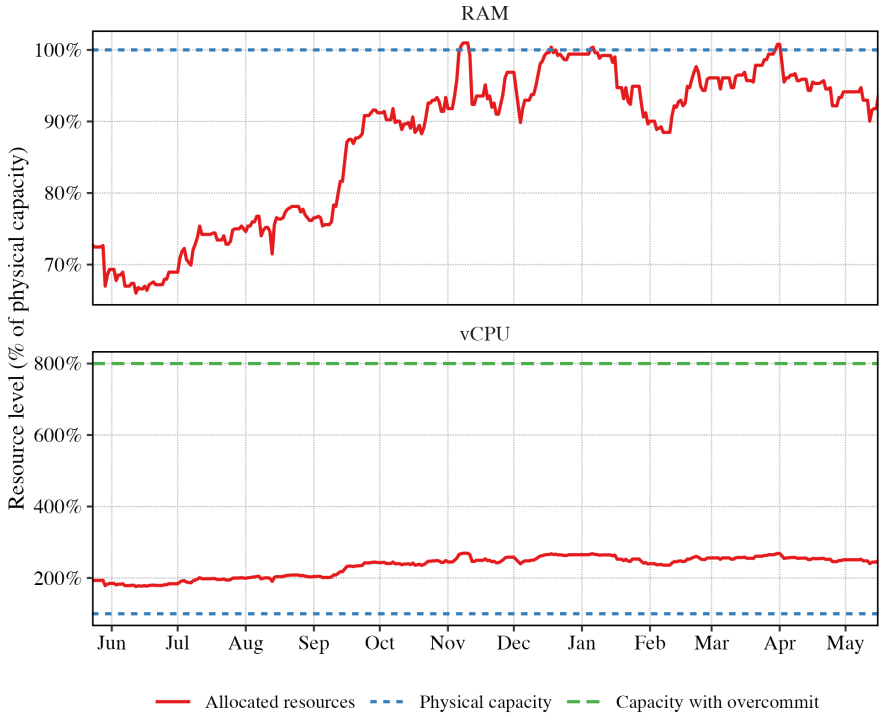


Fig. 4 Aggregate allocated resources normalized by physical capacity.

4.2.1 Quota exhaustion

In the cloud we analyzed, as we described earlier, quotas are oversold, and although the demand for resources is high, for the most part, there is room available to allocate more resources. In this section, we analyze whether a project may become unable to provision additional VMs once its quota is exhausted, independently of cluster-wide headroom. To characterize this constraint, we define *potential failure* as a period during which a project does not have sufficient remaining quota to allocate at least one VM of the smallest available size (1 vCPU and 2 GB of RAM). This condition captures a provisioning limit imposed by quota saturation. In practice, any provisioning attempt during such periods would fail due to the exhaustion of quota limits. We measure the prevalence of potential failure by computing, for each project, the fraction of days spent in this state during the observation period.

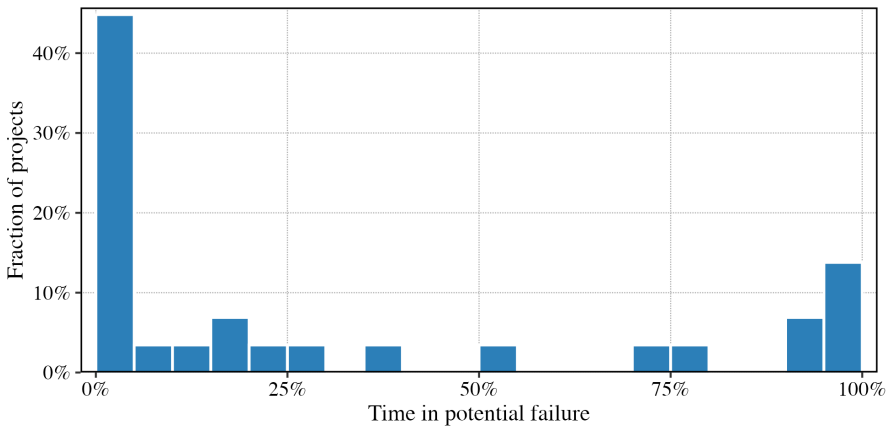


Fig. 5 Distribution of projects by fraction of days spent in potential failure.

Figure 5 shows the resulting distribution. At least half of the projects spend some time in a potential failure state, while 4 remain (about 14% of the projects) in this state throughout their entire observed periods. Figure 6 shows, for each project, the fraction of days spent in potential failure as a function of its mean allocated RAM over the

observation period (as we described earlier, overbooking and allocation pressure are predominantly observed for RAM rather than vCPU). Exposure to potential failure is highest for several small projects and for the largest project, and lower for those with intermediate allocations.

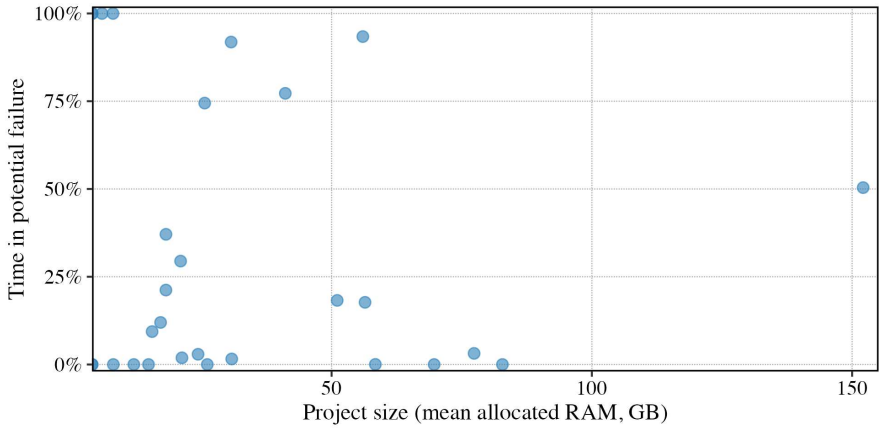


Fig. 6 Project size vs. exposure to potential failure.

To better understand whether the potential failure to allocate is really tied to quota exhaustion instead of resource depletion, Figure 7 relates the daily fraction of projects in potential failure to the aggregate allocated RAM, normalized by physical capacity. Each point represents one day of observation, and the fraction is calculated over projects with allocated resources on that day, whose number varies from 21 to 29 over the observation period. There is a trend of rising potential failure as aggregate memory allocation increases. However, lower and higher allocation levels correspond to earlier and later periods, respectively, with different project populations. Nevertheless, projects continue to experience potential failure even when substantial physical headroom is available at the cluster level. If physical saturation were the primary driver, exposure would increase sharply only near full allocation. Instead, a substantial fraction of projects remains in potential failure across a broad range of allocation levels. Even when total memory allocation remains between 70% and 80% of physical

capacity, approximately 27% of projects operate under quota exhaustion. Although exposure rises as allocation approaches the physical limit, a substantial baseline level of constraint persists well before saturation.

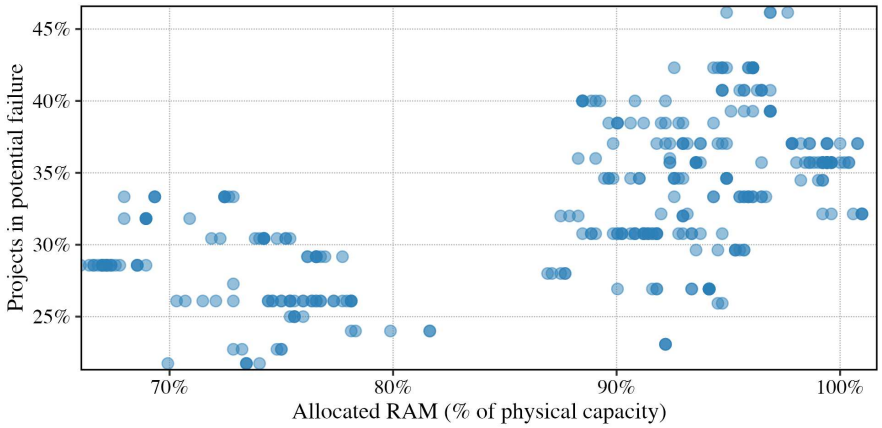


Fig. 7 Fraction of projects in potential failure as a function of aggregate allocated RAM.

Figure 8 complements this analysis by showing that the mean fraction of RAM quota allocated by each project ranges from approximately 37% to 100%, with a median of 70%. Seven projects allocate less than half of their quota on average, whereas four allocate nearly all of it, showing that some projects operate near their quota ceiling while others have a large share of their quota unallocated. The median project leaves approximately 30% of its RAM quota unallocated. At the aggregate level, approximately 38% of the provisioned RAM quota remains unallocated.

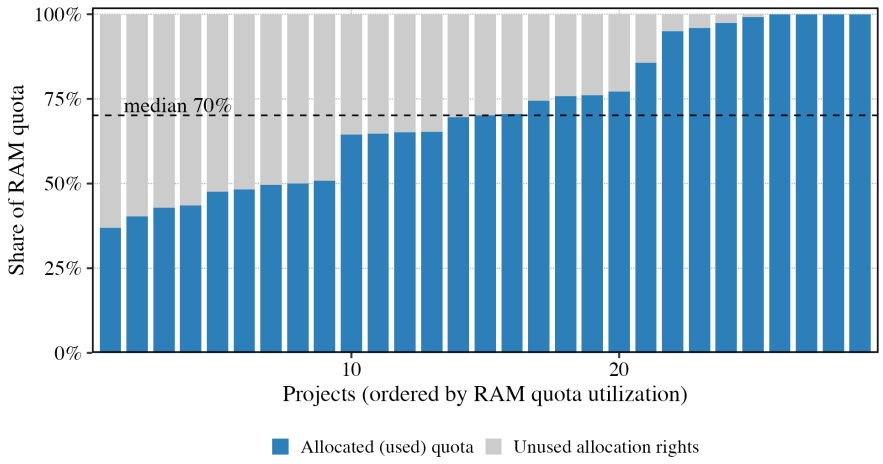


Fig. 8 Mean fraction of RAM quota allocated per project over the observation period.

4.3 The resource hoarding problem

Resource hoarding refers to the tendency of users to retain allocated VMs during extended periods of inactivity or to provision them well in advance of actual need. In the studied infrastructure, VMs consume allocated resources from the moment of creation, regardless of actual workload activity, and those resources count against a project's quota for as long as the VMs exist. The analysis in Section 4.2 shows that some projects are persistently unable to provision additional VMs, even when physical headroom exists at the cluster level. In this section, we ask whether this quota exhaustion reflects genuine demand for resources or VMs held idle against future needs. If hoarding is prevalent, then VMs left idle by some users occupy quota capacity that could otherwise be available to others, and the binding allocation constraints observed in Section 4.2 would arise not from real consumption but from allocation rights held in reserve. Answering this requires going beyond aggregate utilization metrics and examining how each VM's lifetime is actually spent: specifically, how much of it corresponds to meaningful activity versus idleness, and how long VMs sit unused after provisioning.

Before hoarding can be quantified, we need an operational definition of when a VM is in meaningful use and when it is idle. We build it bottom-up, from hours to days to blocks of days, using vCPU utilization as the signal. We rely on vCPU alone and not on memory because vCPU usage rises only when computation is performed, whereas memory usage mostly reflects state retained by the guest, so a VM that holds data in memory without serving requests is idle in the sense relevant to hoarding. Network and disk I/O are not part of the dataset (Section 3) and could not be used. A meaningful block is a contiguous sequence of meaningful days of the same VM, allowing short idle gaps (defined below), and captures a period of genuine workload activity. An idle block is a contiguous period without meaningful use, during which a VM occupies quota without contributing useful work.

At hourly granularity, resampling with the 95^{th} percentile (P95) of vCPU utilization within each hour to stabilize short-term fluctuations, the number of hours that would be classified as meaningful falls sharply as the activity threshold rises and then flattens into a long plateau, with a clear elbow at 10% (Figure 9). Beyond this point, raising the threshold removes comparatively few additional hours, indicating that meaningful hours are concentrated at the low end of the utilization range rather than spread evenly across it. This threshold is consistent with comparable low-CPU cutoffs used elsewhere to separate idle from meaningfully used VMs [7]. We accordingly treat all hours above 10% utilization as meaningful.

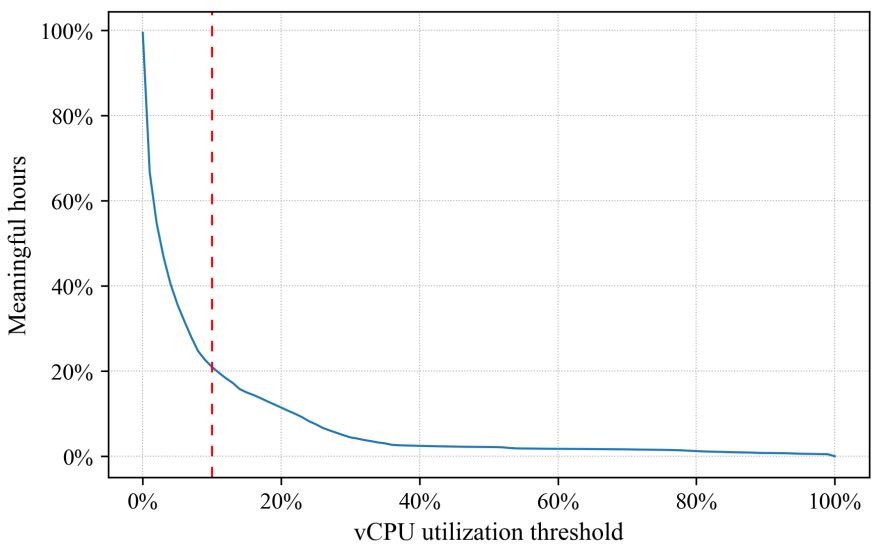


Fig. 9 Proportion of meaningful hours as a function of the activity threshold. The curve exhibits a clear elbow around 10% (dashed red line), after which further increases in the threshold yield diminishing reductions in meaningful hours, motivating the choice of 10% as the classification boundary.

The same pattern reappears at a daily granularity. Weekly usage patterns reveal a second platform-level regularity: one-hour utilization spikes at 3 AM (UTC−3) across many VMs (Figure 10), consistent with automated maintenance tasks rather than user-driven workload. At daily resolution, the count of meaningful days again drops steeply and then flattens, with an elbow at 2 meaningful hours per day (Figure 11). Beyond this point, requiring additional meaningful hours captures comparatively few further days, indicating that meaningful daily usage is concentrated at the low end of the meaningful-hour range rather than spread evenly across it. We label a day as meaningful when it contains at least 2 meaningful hours, a requirement that also keeps the one-hour 3 AM spike from making a day meaningful on its own.

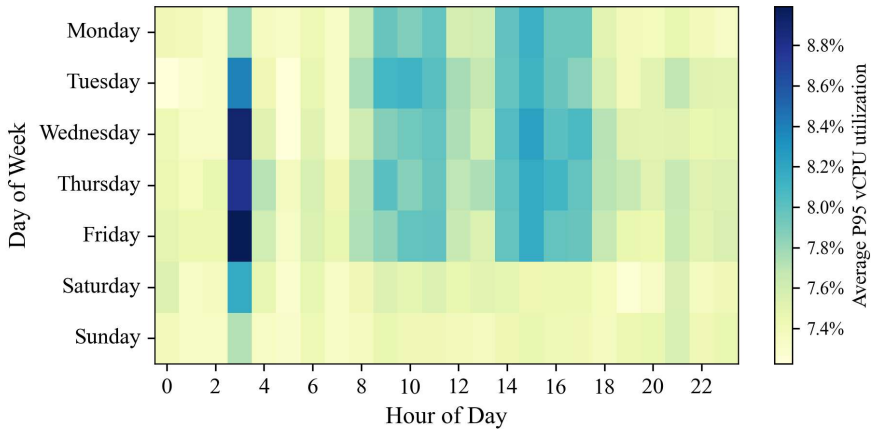


Fig. 10 Heatmap of weekly average 95^{th} percentile (P95) vCPU utilization by day of week and hour of day (UTC−3). A pronounced spike is visible at 3 AM, weaker on Sundays and Mondays. Utilization is otherwise generally higher during business hours and lower on weekends.

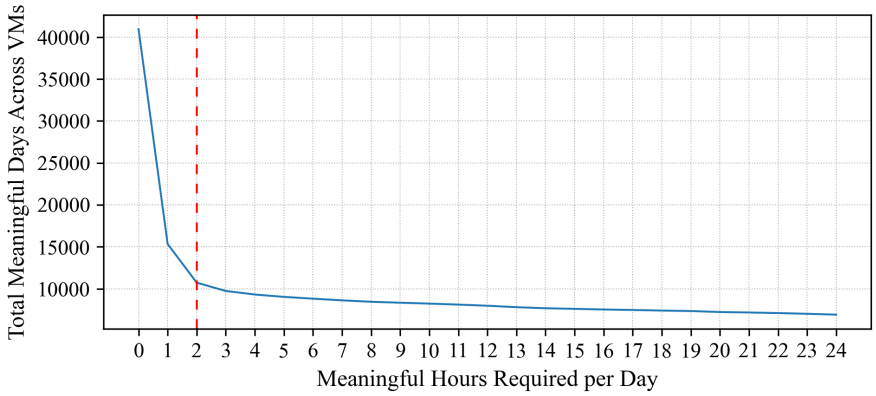


Fig. 11 Number of meaningful days across VMs as a function of the daily meaningful hours required. The curve drops sharply between 0 and 2 hours, after which the rate of decrease slows considerably, motivating the choice of 2 meaningful hours as the threshold for classifying a day as meaningful.

Meaningful days are then grouped into blocks; that is, contiguous sequences of days sharing the same VM and label (meaningful or idle). Short pauses, such as weekends, should not break a period of genuine use, so we bridge idle gaps of at most 3 days

between two meaningful days—a tolerance long enough to cover a weekend or a long weekend. Bridged gaps are merged into the surrounding block and counted as meaningful time. The number of meaningful blocks decreases as more idle days are tolerated (Figure 12), reflecting that short interruptions of a few days are common. Among the bridged gaps, 28% are two-day weekends, more than twice the share expected if gaps were placed on the calendar at random, which shows that weekends are a recurring cause of these interruptions. Unlike the previous two thresholds, the elbow of this curve is not as sharp, so we support the choice of 3 days, the point where the curve flattens, with the sensitivity analysis in Section 4.3.1, which shows that the results do not change substantially when the tolerance is moved between 2 and 5 days.

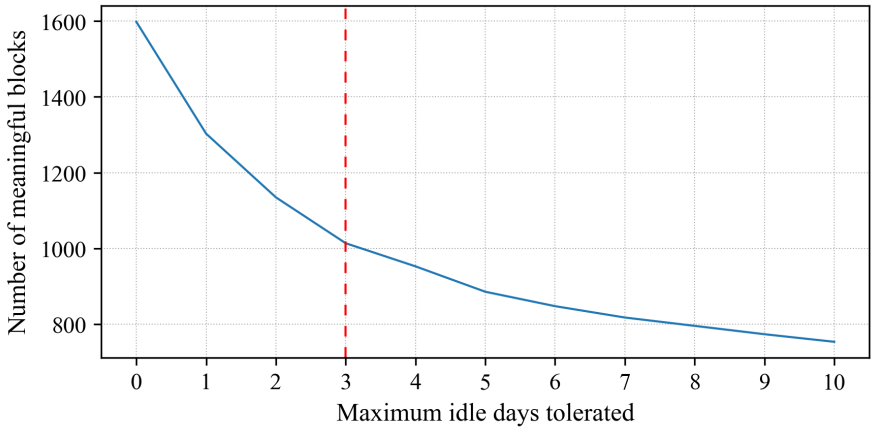


Fig. 12 Number of meaningful blocks as a function of the idle days tolerated within a block. The curve flattens gradually after about 3 days, which we adopt as the tolerance; the results change little when it is varied between 2 and 5 days, as shown in Section 4.3.1.

4.3.1 Measuring hoarding

Having defined activity and idleness through the thresholds above, we now quantify hoarding along the two dimensions identified at the start of Section 4.3: the tendency

to retain VMs during inactivity, and the tendency to provision them before they are needed.

The first dimension is captured by the ratio between total meaningful time and total idle time, summed over all VMs. Across the entire dataset, meaningful time amounts to only about 40% of idle time, meaning that VMs spend more than twice as long idle as they do in meaningful use. Figure 13 plots total meaningful against total idle days for each VM, with the dashed diagonal indicating equal proportions of the two; 74.2% of points lie above it, showing that idleness dominates VM lifetime for most VMs rather than just a small subset of outliers.

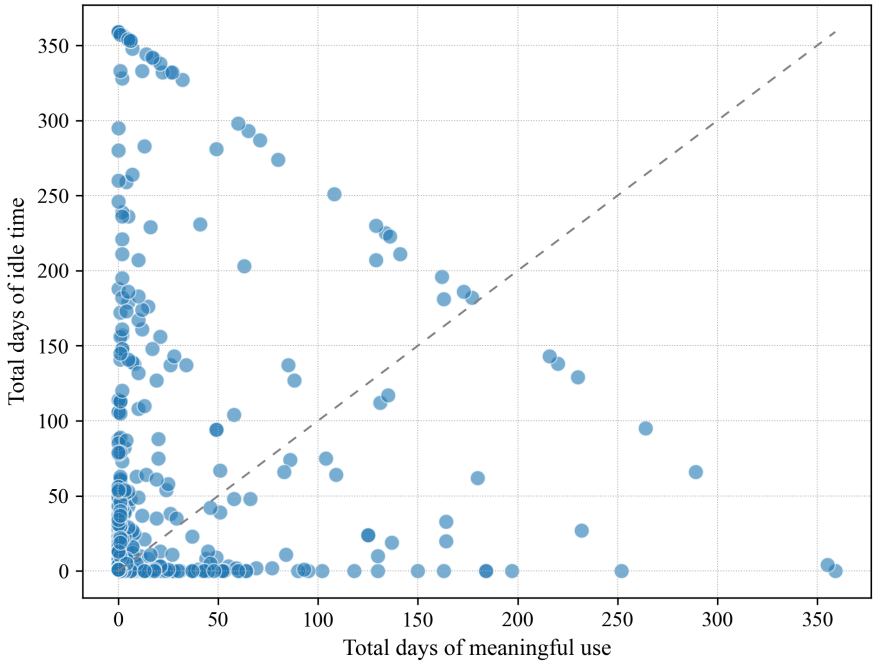


Fig. 13 Total days of meaningful use versus total days of idle time per VM. Points above the dashed diagonal represent VMs where idle time exceeds meaningful use, which is the case for 74.2% of VMs. Bridged gaps count as meaningful time.

The second dimension is the elapsed time between VM creation and the start of the first meaningful block. VMs already present at the beginning of the dataset cannot be assigned a creation time and are therefore excluded, reducing the pool by 118 VMs. Many VMs show a recurring pattern (Figure 14): a short meaningful block right after creation, followed by a long idle period, which suggests initial setup rather than genuine use.

We compare the distribution of leading-block durations with that of meaningful-block durations in general. For each candidate cutoff, we compute the distributional separation between the two, defined as the absolute difference between their empirical cumulative distribution functions (CDFs), that is, between the fractions of leading blocks and of all meaningful blocks lasting at most that many days (Figure 15). The maximum of this difference is the two-sample Kolmogorov–Smirnov statistic, and we take as cutoff the duration at which it is attained: 4 days. Leading meaningful blocks of 4 days or less are reclassified as idle whenever they are followed by an idle block, regardless of that block's length.

The reclassification is applied to the day labels before any measurement, so it affects both the time to first use and the ratio of meaningful to idle time. A leading block that is not followed by an idle block (for instance, because the VM was deleted or the observation ended) is left unchanged, as are VMs that begin with an idle block and the VMs already present at the beginning of the dataset. The rule moves at most 4 days per VM from meaningful to idle, and Figure 14 shows that for most reclassified VMs the idle block that follows is longer than the leading block. For VMs with no meaningful block, either because they were deleted or because the observation ended before their first meaningful use, we take their total idle time as the time to first use. This underestimates their true time to first use, so the reported mean is a lower bound. The cumulative distribution in Figure 16 shows that approximately 50% of VMs reach first meaningful use on the day of creation, driving a median of 0 days;

however, the mean of 5.94 days reflects a tail of VMs left idle for weeks or months before first use. This pattern is more pronounced among smaller VMs, though the reasons remain unclear and warrant further investigation.

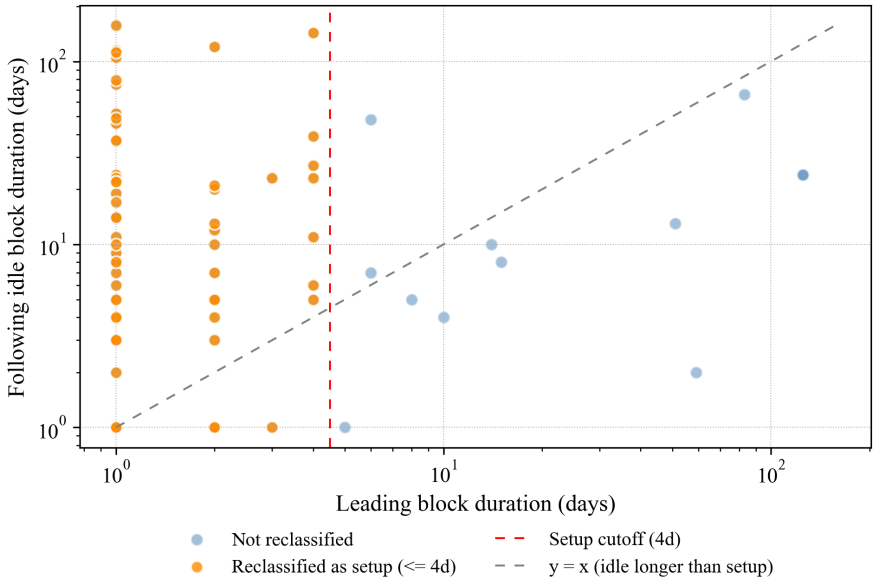


Fig. 14 Duration of the first meaningful block versus the duration of the subsequent idle block. Points above the dashed diagonal represent VMs where the following idle period exceeds the initial meaningful block.

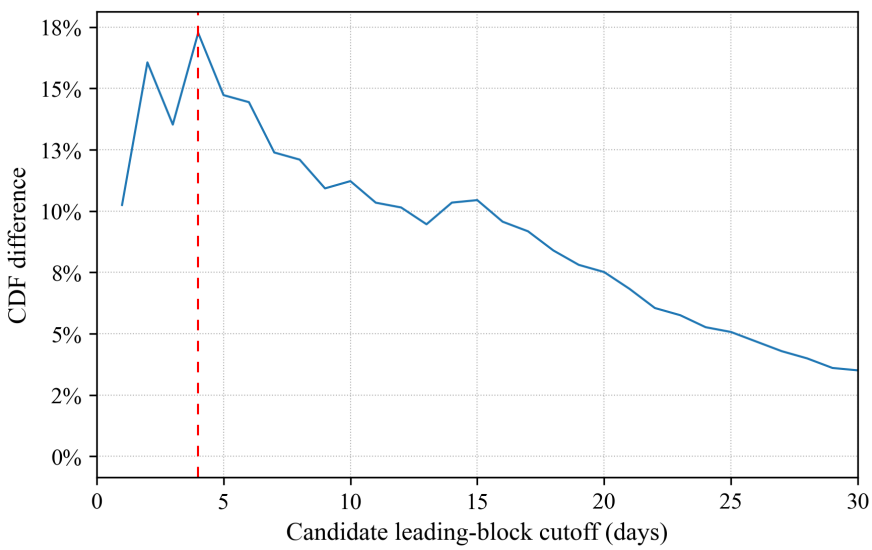


Fig. 15 Absolute difference between the empirical cumulative distribution functions (CDFs) of leading-block and general meaningful-block durations. The maximum, which is the two-sample Kolmogorov–Smirnov statistic, is attained at 4 days (dashed red line), the adopted reclassification threshold.

To check how much these results depend on the 3-day tolerance, we repeat the analysis with tolerances of 2 and 5 days. The ratio of meaningful to idle time changes from 39.71% to 38.04% and 42.43%, respectively. The share of VMs reaching first meaningful use on the day of creation remains at about 50%, and the mean time to first use changes from 5.94 days to 5.96 and 5.90 days, respectively. The peak of the leading-block separation (Figure 15) remains at 4 days under both alternative tolerances.

Taken together, these two measurements answer the question posed at the outset. If VMs were provisioned in response to actual demand, idle time would track meaningful use, and VMs would be put to work shortly after creation. Instead, idle time exceeds meaningful use for 74.2% of VMs (Figure 13), and a substantial tail of VMs sits unused for weeks after creation (Figure 16). The quota exhaustion of Section 4.2 is therefore

consistent with part of the allocation being held in reserve, and not only with genuine consumption. Static quotas capture how much a project could allocate, not how much it is putting to work, and the gap between the two is what our hoarding measures capture.

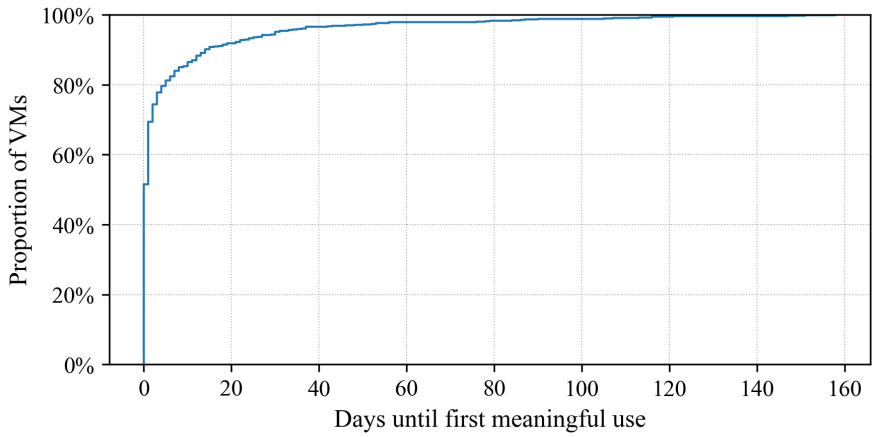


Fig. 16 Cumulative distribution of days elapsed between VM creation and first meaningful use. Approximately 50% of VMs reach first meaningful use on the day of creation, while the long tail indicates that a minority of VMs remain idle for extended periods before being put to use.

5 Discussion

The results of Section 4 describe quota exhaustion and resource hoarding separately. In this section, we first read them together and discuss the incentives that could explain the pattern they form, then consider what it implies for the design of quotas, and finally state the limitations of our study.

5.1 Quota exhaustion, hoarding, and incentives

Section 4.2 shows that quotas, more than physical capacity, limit provisioning: RAM is fully saturated on only 3.1% of days, yet about 27% of projects lack the quota to provision even the smallest VM when aggregate memory allocation is between 70%

and 80% of physical capacity. Section 4.3 shows that much of what is allocated is not used: VMs spend more than twice as long idle as in meaningful use, and a tail of VMs waits weeks before first meaningful use. Taken together, these results are consistent with part of the allocation being held in reserve, so quota exhaustion in this cloud cannot be read only as a sign of genuine demand.

The mechanism outlined in the introduction offers a plausible explanation. Because users are not charged, holding an idle VM costs its owner nothing, whereas releasing it carries a risk: with oversold quotas, a project that gives up capacity has no guarantee of obtaining it again later. Overselling is, in turn, safe for administrators most of the time, since about 38% of the provisioned RAM quota remains unallocated (Figure 8) and aggregate allocation rarely reaches physical capacity (Figure 4). Under these conditions, keeping VMs is a reasonable choice for each project, even though collectively it leaves capacity idle in some projects while others cannot provision. Our data are consistent with this explanation but cannot test it directly, since they record what users allocate and use, not why (Section 5.3).

5.2 Implications for quota design

If this explanation holds, raising quotas or adding capacity would likely treat the symptom rather than the cause, since new allocation rights would be subject to the same incentives. Mechanisms that change the incentives seem more promising. First, quotas that only cap allocation cannot distinguish capacity in use from capacity held in reserve; policies that also account for use, for example by reclaiming or shrinking allocation rights that stay idle for long periods, would target this gap directly, and systems such as CloudGC [8] show that idle VMs can be detected in production. Second, a reservation mechanism with explicit start and end times would give users the guarantee they currently obtain by holding VMs, at the cost of a time limit. Third, even without charging users, regularly reporting to each project how much of

its allocation is idle could make the cost of holding visible and weaken the incentive to hoard. Finally, idle capacity could be harvested rather than reclaimed, for example through backfill VMs [11] or oversubscription driven by usage patterns [3], although this raises utilization without changing the incentives behind the idleness. Our data cannot show which of these mechanisms would work best in practice; evaluating them is part of our future work (Section 6).

5.3 Limitations

Several limitations qualify these results. Our findings come from a single university private cloud observed for one year, so their generalization to other quota policies, user populations, and deployment sizes remains to be verified. Projects also joined the platform during the observation period and the aggregate quota grew accordingly, so the trends in Figures 3, 4, and 7 mix the effect of allocation pressure with that of a changing project population.

The two analyses also have limits of their own. Potential failure is inferred from quota arithmetic rather than from logged rejections, so it measures exposure to failed provisioning, not the number of requests actually denied or the demand that went unmet; it also considers only compute quotas, although projects can be blocked by other quotas, such as volumes or floating IP addresses, which we do not analyze. Idleness is defined on vCPU utilization alone (Section 4.3), so VMs that are genuinely used but rarely compute, such as in-memory caches, databases with low query rates, or jump hosts, are classified as idle, and the reported idle fractions should be read as upper bounds. Idleness is also not always waste: some VMs are idle by design, such as standby replicas or VMs that run periodic jobs, and our data do not reveal the purpose of a VM, so not all of the idle time we measure is hoarding. The link between quota exhaustion and hoarding rests on the consistency of the two analyses rather than on user-level evidence, since the hoarding metrics cover all VMs and not only

those in quota-bound projects, and the incentive explanation is not tested directly. Finally, idle and meaningful time are counted per VM without weighting by VM size, so the idle share of allocated capacity may differ from the idle share of VM time.

6 Conclusion and future work

Quota-oriented private clouds inherently decouple physical capacity from effective allocation rights. Although aggregate memory allocation frequently remains below physical limits, a non-negligible fraction of projects is simultaneously unable to provision additional VMs due to quota exhaustion. This behavior indicates that provisioning limits are defined primarily by static quotas rather than by instantaneous resource availability. As a result, memory may remain idle at the cluster level while projects operate under binding allocation constraints. At the same time, the resources that are allocated are poorly used: VMs spend more than twice as long idle as in meaningful use, and although about half of them reach their first meaningful use on the day of creation, some wait for weeks (Section 4.3). Based on these observations, we argue the following. (1) Static quotas can introduce systematic inefficiencies when not dynamically adjusted to sustained allocation patterns. (2) The absence of reservation or elasticity mechanisms amplifies heterogeneity across projects, causing some to experience persistent quota saturation while others retain unused allocation rights. (3) Without adaptive quota management, quotas may become a major driver of perceived resource scarcity, even under substantial physical headroom; the idle time we measure is consistent with allocation rights being held in advance and after use, as discussed in Section 5.

Building on these findings, our future work will focus on investigating alternative resource management strategies to overcome the limitations of fixed, static quotas. Specifically, we plan to explore dynamic, utilization-aware allocation mechanisms and

adaptive scheduling models that adjust to actual workload behaviors rather than relying on rigid administrative boundaries. To this end, we will evaluate approaches such as exploiting historical and temporal usage patterns to predict resource demands and enable smarter oversubscription, a strategy that has shown significant promise in large-scale cloud platforms [3]. Additionally, we intend to explore the implementation of utilization-prediction-aware VM consolidation techniques [10] and the introduction of backfill VMs [11] to actively harvest idle node capacity, which can substantially increase overall infrastructure utilization. We also plan to address the limitations discussed in Section 5.3 by defining meaningful use on several resources, testing the sensitivity of all thresholds, gathering user-level evidence on why VMs are kept idle, and repeating the analysis on other private clouds. By transitioning from static quota policies to predictive, elasticity-driven frameworks, we aim to mitigate quota-induced resource scarcity and significantly improve the operational efficiency of private cloud environments.

Acknowledgements. Not applicable.

Declarations

List of abbreviations.

API	Application Programming Interface
CDF	Cumulative Distribution Function
CPU	Central Processing Unit
IaaS	Infrastructure as a Service
IP	Internet Protocol
P95	95th percentile
RAM	Random Access Memory
UFCG	Federal University of Campina Grande
UUID	Universally Unique Identifier
vCPU	Virtual CPU
VM	Virtual Machine

Funding. Thiago Emmanuel Pereira is a CNPq/Brazil researcher (grant 313068/2025-9).

Competing interests. The authors declare no competing interests.

Ethics approval and consent to participate. Not applicable.

Data availability. The dataset supporting the conclusions of this article is available in the Mendeley Data repository, https://data.mendeley.com/datasets/trvb5k4x5m/2, and is described in [12].

Materials availability. Not applicable.

Code availability. https://github.com/ufcg-lsd/private-cloud-analytics

Author contributions. Paola Marques: Writing – Original Draft, Investigation, Formal analysis, Visualization. Francesco Desimoni: Writing – Original Draft, Investigation, Formal analysis, Visualization. Francisco Brasileiro: Writing – Review & Editing, Conceptualization, Methodology. Massimo Canonico: Writing – Original Draft, Writing – Review & Editing, Conceptualization, Methodology. Thiago Emmanuel

Pereira: Writing – Original Draft, Writing – Review & Editing, Conceptualization, Methodology.

Consent for publication. All authors consent to the publication of this manuscript.

References

[1] Reiss, C., Tumanov, A., Ganger, G.R., Katz, R.H., Kozuch, M.A.: Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In: Proceedings of the Third ACM Symposium on Cloud Computing. SoCC '12, pp. 1–13. Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2391229.2391236

[2] Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., Bianchini, R.: Resource Central: Understanding and predicting workloads for improved resource management in large cloud platforms. In: Proceedings of the 26th Symposium on Operating Systems Principles. SOSP '17, pp. 153–167. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3132747.3132772

[3] Reidys, B., Zardoshti, P., Goiri, Í., Irvene, C., Berger, D.S., Ma, H., Arya, K., Cortez, E., Stark, T., Bak, E., Iyigun, M., Novakovic, S., Hsu, L., Trueba, K., Pan, A., Bansal, C., Rajmohan, S., Huang, J., Bianchini, R.: Coach: Exploiting temporal patterns for all-resource oversubscription in cloud platforms. In: Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1. ASPLOS '25, pp. 164–181. ACM, New York, NY, USA (2025). https://doi.org/10.1145/3669940.3707226

[4] Khan, A., Yan, X., Tao, S., Anerousis, N.: Workload characterization and prediction in the cloud: A multiple time series approach. In: 2012 IEEE Network Operations and Management Symposium, Maui, HI, USA, pp. 1287–1294 (2012). https://doi.org/10.1109/NOMS.2012.6212065

[5] Uhlig, A., Braun, I., Wählisch, M.: The SAP cloud infrastructure dataset: A reality check of scheduling and placement of VMs in cloud computing. In: Proceedings of the 2025 ACM Internet Measurement Conference. IMC '25, pp. 746–760. Association for Computing Machinery, New York, NY, USA (2025). https://doi.org/10.1145/3730567.3764480

[6] Guo, J., Chang, Z., Wang, S., Ding, H., Feng, Y., Mao, L., Bao, Y.: Who limits the resource efficiency of my datacenter: an analysis of Alibaba datacenter traces. In: Proceedings of the International Symposium on Quality of Service. IWQoS '19, pp. 1–10. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3326285.3329074

[7] Everman, B., Gao, M., Zong, Z.: Evaluating and reducing cloud waste and cost - A data-driven case study from Azure workloads. Sustainable Computing: Informatics and Systems **35**, 100708 (2022) https://doi.org/10.1016/j.suscom.2022.100708

[8] Zhang, B., Al-Dhuraibi, Y., Rouvoy, R., Paraiso, F., Seinturier, L.: CloudGC: Recycling idle virtual machines in the cloud. In: Proceedings of the 2017 IEEE International Conference on Cloud Engineering. IC2E '17, pp. 105–115. IEEE Computer Society, Los Alamitos, CA, USA (2017). https://doi.org/10.1109/IC2E.2017.26

[9] Beloglazov, A., Buyya, R.: 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) https://doi.org/10.1002/CPE.1867

[10] Hsieh, S.-Y., Liu, C.-S., Buyya, R., Zomaya, A.Y.: Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. Journal of Parallel and Distributed Computing **139**, 99–109 (2020) https://doi.org/10.1016/j.jpdc.2019.12.014

[11] Marshall, P., Keahey, K., Freeman, T.: Improving utilization of infrastructure clouds. In: Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. CCGrid '11, pp. 205–214. IEEE Computer Society, Los Alamitos, CA, USA (2011). https://doi.org/10.1109/CCGrid.2011.56

[12] Marques, P., Mendes, M., Pereira, T.E., Farias, G.: Dataset on resource allocation and usage for a private cloud. Data in Brief **65**, 112514 (2026) https://doi.org/10.1016/j.dib.2026.112514

[13] Arabat, A., Sayagh, M.: An empirical study on cross-component dependent changes: A case study on the components of OpenStack. Empirical Software Engineering **29**(5), 109 (2024) https://doi.org/10.1007/s10664-024-10488-y

[14] Aghayev, A., Weil, S.A., Kuchnik, M., Nelson, M., Ganger, G.R., Amvrosiadis, G.: File systems unfit as distributed storage backends: lessons from 10 years of Ceph evolution. In: Proceedings of the 27th ACM Symposium on Operating Systems Principles. SOSP '19, pp. 353–369. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3341301.3359656