International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 15(2023) pp. 1-28
https://doi.org/10.70917/ijcisim-2026-5923
Article

Certified Bounds on the Live Migrations Needed to Reclaim Servers Fragmented by Online Vector Bin Packing on a Production Cloud Trace

Luis Eduardo Muñoz Guerrero[1],*

[1]Universidad Tecnológica de Pereira, Facultad de Ingenierías, Pereira, Colombia

Abstract: Background. Online allocators place virtual machines (VMs) without knowing their life-times; departures leave servers fragmented, and reclaiming them takes costly live migrations. We bound how many servers a migration budget reclaims in a public production trace.
Methods. We prove the minimum-migration problem strongly NP-hard and, unless P = NP, inapproximable, and bracket the servers that a budget reclaims between a greedy drain and an elementary lower bound on migrations; column generation brackets the static optimum. Re-playing the Azure VM packing trace under five online policies, we test, under a plan frozen before opening the second week, whether a budget of 1% of the live VMs reclaims more than 1% of the servers over 167 hourly instants, with validity checks, 1,200 multiverse speci-fications, an external replication and exact baselines.
Results. Under Best Fit, a budget of 1% of the live VMs reclaimed for certain 5.62% of the servers in use on average (one-sided 95% lower bound 5.38%), valid under the plan, while no algorithm could reclaim more than 6.89%, the whole excess over the size-sum bound; at the 7 daily checkpoints the static optimum lay within 2 servers of the size-sum bound. The drain achieved on average 0.83 of the certified maximum, below the planned 0.9; 69% of the multiverse specifications exceeded the threshold, including every one with a budget of 2% or more; the replication was positive for all three machine shapes; and on small instances both bounds matched the exact minimum. Exploratory: with 0.1% of the VMs the bounds coin-cided at every instant; with 5% the drain stopped short of the excess before exhausting the budget.
Conclusions. Certified bounds make the migration cost of reclaiming fragmented servers measurable at production scale; on this trace under Best Fit, a budget of 1% of the live VMs reclaims 5.6% of the servers on average.
Keywords: Vector bin packing; live migration; virtual machine consolidation; column generation; NP-hardness; lower bounds; cloud computing

1. Introduction

An online allocator of virtual machines (VMs) places each request on a server without knowing how long the VM will live; as VMs depart, servers are left partially used, and the fleet keeps more servers busy than the live demand needs.

This paper measures, with certified lower and upper bounds, how many live migrations are needed to reclaim those servers in a public production cloud trace.

Placing VMs on servers with several resources is a vector bin packing problem [1, 2], and doing it as requests arrive and depart is dynamic bin packing, for which no online algorithm can guarantee a performance substantially better than that of First Fit [3].

Production allocators care about small gaps: the designers of Azure's allocator report that, at their scale, “even 1% in fragmentation reduction can lead to cost savings in the order of $100M per year” [4].

Copyright: © 2023 by the authors

During a capacity crunch, the same allocator identified fragmented machines and recommended migration targets, and a migration mechanism live-migrated some VMs to improve packing density [4].Live migration moves a running VM between hosts with a short service downtime [5], but it can degrade the service levels of the migrated applications and has costs that cannot be disregarded [6–8].

For an operator, the question is therefore one of cost: how many migrations buy back how many servers? Dynamic consolidation migrates VMs to reduce the number of active machines [9, 10]; its heuristics are validated by simulation on workload traces [10, 11], and integer programs can minimize the number of migrations [12] or limit it [11]. A survey of VM allocation finds the proposed approaches hardly comparable, because their problem models differ [13]. Theory bounds, in the worst case, the repacking that keeps a packing close to optimal when items arrive, or arrive and depart [14–19], which does not say what a given production sequence needs. Production traces have been released to the community [20], including one meant to design and test packing algorithms [4], which has been used to shrink offline instances of dynamic vector bin packing [21]. To our knowledge, no prior study reports certified lower and upper bounds on the live migrations needed to reclaim the machines left fragmented by online allocation of a public production VM trace.

This paper makes four contributions. First, it states the minimum-migration problem (MinMig), the least number of VMs that must change machine to vacate a given number of machines, and shows by a short reduction from bin packing that it is NP-hard in the strong sense and, unless P = NP, admits no approximation within any finite factor, which is why it is bracketed rather than computed at production scale. Second, it pairs an elementary certified lower bound on migrations with a greedy drain whose every prefix is a feasible placement, so the number of machines that a migration budget can reclaim is bracketed from both sides.

Third, it brackets the static optimum at daily checkpoints of the measured week by column generation with a Farley bound and an integer solution, at the scale of tens of thousands of machines.

Fourth, it measures these quantities hour by hour in the second week of the Azure packing trace, which stayed sealed until an analysis plan was frozen in the repository, and reports the pre-registered primary result with its validity checks, secondary hypotheses, a multiverse of specifications, an external replication, exact baselines and measured computational cost.

On the sealed week, under Best Fit, a budget of 1% of the live VMs reclaims for certain 5.6% of the machines in use on average, out of an excess of 6.9% that bounds what any repacking can reclaim. Section 2 describes the data, the formulation, the bounds, the algorithm and the analysis plan; Section 3 reports the results, following the order fixed by the plan; Section 4 discusses implications and threats to validity; Section 5 concludes.

2. Materials and Methods

Data. We use the Azure VM packing trace released with the Protean allocator [4] under a CC BY 4.0 license. It records 5,559,800 VM requests of one production zone over a window of 14 days: for each VM, its type, its priority and its start and end times (in days; a missing end means that the VM was still alive when the trace closed), and for each pair of VM type and machine type, the fraction of each machine resource that the VM needs. We recorded the SHA-256 digests of the archive and of its database at download, so that every later download is checked against them, documented the trace in a datasheet [22] and checked its schema and values before any analysis (no duplicate identifiers and no end time that precedes its start); Table 1 summarizes the trace, those checks and the partition below, and Figure 1 shows its daily arrivals, how long its VMs stay alive and how many of its requests each machine type can host. The trace does not contain the placement of the VMs, the number of machines of the zone or calendar dates. Following the trace, the rest of the paper calls the servers machines.

Sealed partition. The trace was split in time before looking at any outcome. Requests that start in the first seven days (3,315,629, with their end times truncated at day seven so that nothing leaks from the second week) formed the exploration set, used only to design the analysis; the second week (2,244,171 further requests) stayed sealed. The

code that loads it refuses to run unless the frozen-plan tag exists and the current commit descends from it, exposes only the fields that the plan declares and logs every access.

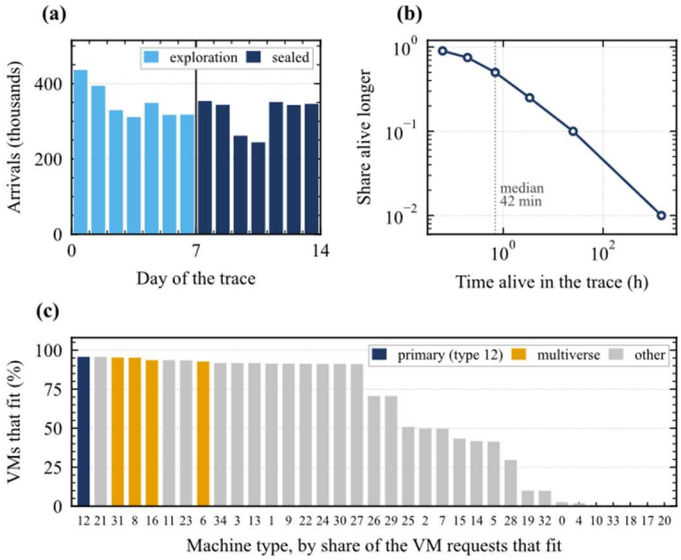


Figure 1. The trace. (a) VM requests that arrive each day, in the exploration week (light) and in the sealed week (dark). (b) Survival function of the time alive in the trace of the VMs that end, from its 10th to its 99th percentile, on logarithmic axes; the dotted line marks the median, 42 min. (c) Share of the 5,559,800 VM requests whose type can run on each of the 35 machine types, sorted; dark: the primary machine type, which hosts 95.7%; orange: the other machine types of the multiverse.

Problem. At an instant t the online policy leaves the live VMs V_t on U_t machines (Table 2 collects the notation). Machines are identical, with capacity one in every dimension, and the size-sum bound is

$$L_t = \lceil \max_{1 \le d \le D} \sum_{v \in V_t} s_{vd} \rceil \le OPT_t \le U_t \tag{1}$$

Any repacking uses at least OPT_t machines, so by Eq. (1) the excess $E_t = U_t - L_t$ bounds from above the number of machines that it can save; the question is how many VMs must move to save them. The excess splits as

$$E_t = (U_t - OPT_t) + (OPT_t - L_t), \tag{2}$$

where the first term of Eq. (2) is the number of machines that an optimal repacking saves and the second is the slack of the size-sum bound, which column generation brackets at the checkpoints (Lemma 1 below).

Problem 1 (minimum migration). Given the online placement x and an integer Δ, find a feasible placement on at most $U_t - \Delta$ machines that changes the machine of as few VMs as possible. Its value $MinMig_t(\Delta)$ is the optimum of the integer program

(3)

$$\min \sum_{v \in V_t} (1 - y_{v\,x(v)}) \; s.t. \quad \sum_{j} y_{vj}$$

$$= 1 \quad \forall v \sum_{v \in V_t} s_{vd}\, y_{vj} \le u_j \quad \forall j, d \sum_{j} u_j$$

$$\le U_t - \Delta \; y_{vj}, u_j \in \{0,1\}$$

The value of Eq. (3) is infinite when no feasible placement exists. With a budget of $m = \lfloor \beta n_t \rfloor$ migrations, the number of reclaimable machines is

$$\Delta_t^*(\beta) = max\{\Delta : MinMig_t(\Delta) \le m\} \tag{4}$$

By Theorem 1 below, no polynomial-time algorithm computes Eq. (4) unless P = NP: with $m \ge n_t$, $\Delta \le \Delta_t^*(\beta)$ holds exactly when MinMig has a feasible solution, since a placement changes the machine of at most n_t VMs. It is therefore not computed exactly at production scale; Theorem 2 brackets it from both sides.

Complexity. Theorem 1. *Deciding whether MinMig has a feasible solution is NP-complete in the strong sense, already with one dimension. Hence MinMig is strongly NP-hard and, unless P = NP, no polynomial-time algorithm approximates it within any finite factor.*

Proof. We reduce from Bin Packing in its decision version, which is NP-complete in the strong sense [23, 24]. Given item sizes $a_1, \ldots, a_n \in (0,1]$ and an integer k, build in linear time an instance with $D = 1$, one VM per item with $s_v = a_v$, the placement that puts every VM alone on its own machine (feasible because $a_v \le 1$; here $U_t = n$) and $\Delta = n - k$.

If the items fit into k bins, that packing is a feasible solution of Eq. (3); conversely, a feasible solution uses at most k machines, which are bins holding all the items. The sizes are copied unchanged, so strong NP-completeness carries over. An approximation algorithm with a finite ratio would return a finite value exactly on the feasible instances and would decide Bin Packing. ■

Table 1. Characteristics of the trace, checks run before any analysis, and the temporal partition.

Quantity	**Value**
VM requests	5,559,800
Tenants (anonymized)	1,899,639
VM types used / defined	265 / 488
Machine types	35
High / low priority	4,963,900 / 595,900
Alive when the trace starts	860,365
Without end time (alive at the end)	570,284
Duplicate identifiers; end not after start	0; 0
Requests that machine type 12 can host	5,321,338 (95.7%)
Quartiles of the time alive in the trace, VMs that end (hours)	0.19, 0.69, 3.39
Exploration week (start before day seven)	3,315,629

Sealed week (start from day seven)	2,244,171

Table 2. Notation of the formulation, the bounds and Algorithm 1 (Table 3 gives that of the statistical analysis).

Symbol	Meaning
$t; V_t, n_t$	instant (days since the start of the trace); VMs alive at t and their number
D; s_v	number of resource dimensions; demand of VM v as a fraction of one machine, in $(0,1]^D$
U_t; c_j; $c_{(i)}$	machines in use at t; number of VMs initially on machine j; i-th smallest c_j
L_t, OPT_t, E_t	size-sum bound, static optimum (vector bin packing) and excess
$x(v)$; y_{vj}, u_j	machine of v in the online placement; binary assignment and machine-use variables
Δ; $MinMig_t(\Delta)$	machines to vacate; minimum number of VMs that must change machine (Problem 1)
β; m; $\Delta_t^*(\beta)$	budget as a fraction of the live VMs; budget in migrations; reclaimable machines
l_j, r_j; O_v	load and remaining capacity of machine j; open machines where v fits
z_{RMP}, π; A	value and dual prices of the restricted master linear program; feasible machine patterns
K, b_k; z_{MP}	VM types and live VMs of type k; value of the full master linear program
v^-; U^*	upper bound on the pricing knapsack; machines of an integer solution
L_{mig}, U_{mig}	lower bound on migrations; cumulative migrations of Algorithm 1
w_v, w_j; m^w	memory of VM v and of the VMs initially on machine j; budget in migrated memory
$f_t(\beta), g_t(\beta)$	certified lower and upper bounds on $\Delta_t^*(\beta)$

The static problem behind OPT_t is vector bin packing, which has no asymptotic approximation scheme already in two dimensions unless P = NP [25]. Exact solutions are therefore out of reach at production scale, and the measurements below rest on bounds that are certified instead.

Certified bounds. Lemma 1 (static optimum). *Let z_{RMP} be the value of the restricted master linear program at an iteration of column generation over the feasible machine patterns A* [26, 27]*, π its dual prices, $v^- \geq max_{a \in A} \pi^\intercal a$ an upper bound on the pricing knapsack and U^* the number of machines of an integer solution. Then*

$$\frac{z_{RMP}}{max(1, v^-)} \leq OPT_t \leq U^* \tag{5}$$

Proof. The scaled prices $\pi / max(1, v^-)$ are feasible for the dual of the full pattern program, so their objective bounds its optimum from below, and that optimum bounds OPT_t from below; the integer solution gives the upper

side of Eq. (5). ■ A bound of this kind was proposed to terminate cutting stock linear programs before optimality [28]; here it also certifies the integer optimum when its rounded-up value meets an integer solution.

Lemma 2 (lower bound on migrations). *For every* $\Delta \le U_t$,

$$MinMig_t(\Delta) \ge L_{mig}(\Delta) = \sum_{i=1}^{\Delta} c_{(i)} \tag{6}$$

Proof. A placement on at most $U_t - \Delta$ machines leaves at least Δ of the current machines without VMs. Every VM on those machines must move, and any Δ of them hold at least the sum in Eq. (6). ■ With a weight $w_v \ge 0$ per migrated VM (for instance, its memory), the same argument bounds the minimum migrated weight by the sum of the Δ smallest machine weights.

Lemma 3 (invariants of Algorithm 1). *During Algorithm 1, (i) no machine exceeds its capacity in any dimension; (ii) each committed drain reduces the number of machines in use by exactly one; (iii) a failed drain leaves the state unchanged; and (iv) the accumulated cost equals the number of distinct VMs whose machine differs from their initial one.*

Proof. (i) A VM is placed only where it fits in every dimension. (ii) The drained machine empties and no machine is opened. (iii) Tentative placements are undone. (iv) A VM is counted when it first leaves its initial machine, and drained machines are never destinations. ■

Theorem 2 (certified interval). *Let*

$$\begin{aligned} f_t(\beta) &= max\{\Delta \le E_t : U_{mig}(\Delta) \le m\}\, g_t(\beta) \\ &= max\{\Delta \le E_t : L_{mig}(\Delta) \le m\} \end{aligned} \tag{7}$$

Then the bounds of Eq. (7) *satisfy* $f_t(\beta) \le \Delta_t^*(\beta) \le g_t(\beta)$.

Proof. Every prefix of Algorithm 1 is a feasible placement on $U_t - \Delta$ machines that moves $U_{mig}(\Delta)$ VMs (Lemma 3), so $MinMig_t(f_t(\beta)) \le m$ and $\Delta_t^*(\beta) \ge f_t(\beta)$. By Lemma 2, $L_{mig}(\Delta_t^*) \le MinMig_t(\Delta_t^*) \le m$, and $\Delta_t^* \le U_t - OPT_t \le E_t$ by Eq. (1); hence Δ_t^* belongs to the set that defines $g_t(\beta)$. ■ **Corollary.** If Algorithm 1 vacates E_t machines at an instant, then $OPT_t = L_t$ there: the drain certifies the static optimum.

Figure 2 illustrates Algorithm 1 and the interval on a small state built for the purpose, not taken from the trace: within a budget of three migrations the drain vacates two machines, the bound of Eq. (6) does not rule out a third, and the third cheapest machine holds a VM that fits on no other machine, so Δ_t^* equals the lower end of the interval.

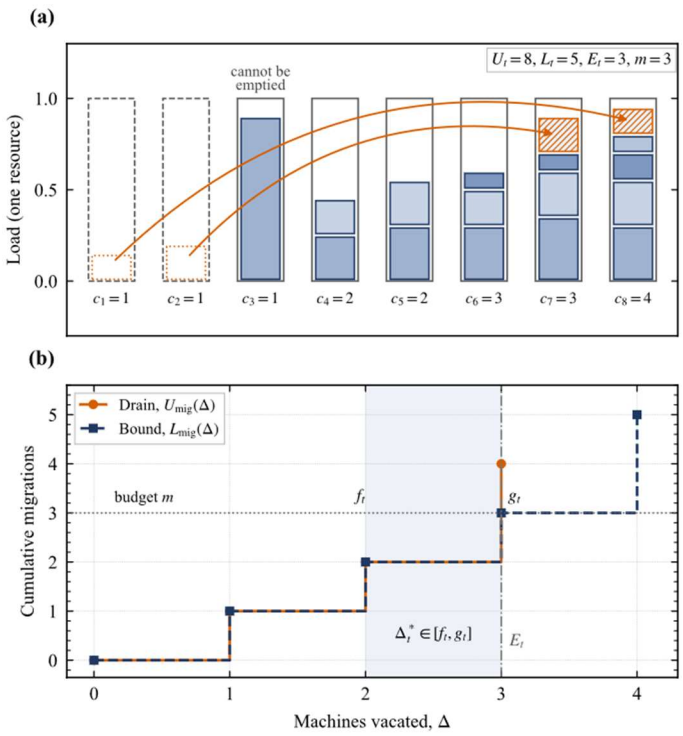


Figure 2. Algorithm 1 and the certified interval on a small illustrative state (built for the figure, not data of the study). (a) Eight machines of capacity one with their VMs stacked, labeled with their numbers of VMs c_j; within a budget of three migrations the drain empties the first two machines (dashed) into the machines with the largest load where each VM fits (hatched), and the third cannot be emptied. (b) Cumulative migrations of the drain, $U_{mig}(\Delta)$, and of the bound $L_{mig}(\Delta)$ against the machines vacated: f_t and g_t are the last points of each curve within the budget m, capped at the excess E_t, and Δ_t^* lies between them (shaded).

Algorithm 1. The upper bound comes from a greedy drain (Algorithm 1). It visits the machines in use from the one with the fewest VMs, tries to move every VM of the visited machine to another machine in use, and keeps the moves only if the whole machine empties; otherwise it undoes them. VMs leave a drained machine from the largest to the smallest (by their largest dimension), and each goes to the machine, among those not drained, with the largest load in the maximum norm where it fits. Ties are broken by index, so the result is a deterministic function of the state. VMs leave only drained machines, so the VMs initially on a machine have not moved when it is visited and draining it costs c_j migrations; the VMs it received from earlier drains move again at no extra cost, because each VM is counted once. With a weight per VM, c_j becomes the total weight of the VMs initially on j. Every committed drain is a feasible placement with one machine fewer (Lemma 3), which is what Theorem 2 needs; no optimality is claimed for the drain itself, and the exact baselines below measure how far it is from MinMig on small states.

Algorithm 1 Greedy drain with undo

Input: placement x on U machines, demands s_v, budget m, cap E

Output: cumulative migrations $U_{mig}(1), \ldots, U_{mig}(\Delta)$

sort the machines in use by their initial c_j, ascending (ties by index)

$\Delta \leftarrow 0$, $cost \leftarrow 0$

for each machine j in that order **do**

  if $\Delta = E$ or $cost + c_j > m$ **then stop**

  mark j; **for** each VM v now on j, by decreasing $max_d s_{vd}$ (ties by index) **do**

    $h \leftarrow$ unmarked machine in use where v fits, with the largest L_∞ load (ties by index)

    if no such h **then** undo the tentative moves, unmark j and go to the next j

    place v on h tentatively

  commit: $\Delta \leftarrow \Delta + 1$, $cost \leftarrow cost + c_j$, $U_{mig}(\Delta) \leftarrow cost$

return $U_{mig}(1..\Delta)$

Online replay. All machines are of type 12, the type that can host the most requests (95.7%; type 21 can host the same ones), with its four resources (cores, memory, solid-state disk and network); requests of VM types that it cannot host are excluded. We replay the requests in time order: a VM arrives at its start time and leaves at its end time, if the trace records one. At equal times departures are processed before arrivals, and ties are broken by VM index; the VMs that were alive when the trace starts arrive first, in the order of their (anonymized) start times. A policy places each arriving VM on an open machine where it fits in every dimension and opens a new machine only when none fits; it never migrates. With $l_j \in [0,1]^D$ the load of machine j, $r_j = 1 - l_j$ its remaining capacity and O_v the open machines where v fits ($s_v \le r_j$), the primary policy, Best Fit with the load measured in the maximum norm (BF), and the secondary ones, Best Fit with the sum of loads (BF-L1), Worst Fit (WF), First Fit (FF) and Dot Product (DP), choose

$$\begin{aligned} BF\colon j^* &= arg\,max_{j \in O_v} \| l_j \|_\infty \quad BF - L1\colon j^* \\ &= arg\,max_{j \in O_v} \| l_j \|_1 \quad WF\colon j^* \\ &= arg\,min_{j \in O_v} \| l_j \|_\infty \quad FF\colon j^* \\ &= min\, O_v \quad DP\colon j^* \\ &= arg\,max_{j \in O_v} \frac{s_v^{\mathrm{T}} r_j}{\| s_v \|_2 \| r_j \|_2} \end{aligned} \tag{8}$$

with ties broken by the lowest index. The DP rule of Eq. (8) adapts to online arrivals the dot-product heuristic for vector bin packing [1], which fills one bin at a time with the item of largest weighted dot product. The policies see

each end time only when the departure occurs, so the placement before an instant does not depend on later end times; an automated test perturbs those end times and checks that no placement changes.

Measurement. At each measurement instant we copy the online state and record the live VMs and the machines in use, the size-sum bound and the excess. Algorithm 1 then runs once, capped at the excess and with the largest budget of the plan (5% of the live VMs); since every prefix of its run is feasible, $f_t(\beta)$ for any smaller budget is read from its cumulative curve, and $g_t(\beta)$ follows from the sorted machine VM counts. After each drain the code checks that no machine exceeds its capacity and that every VM is still placed exactly once.

Static certification. At noon of each day of the sealed week (seven checkpoints) we certify the static optimum by column generation [26, 27]. VMs of the same type have identical demands, so the instance is a cutting stock problem with one demand per VM type: with b_k live VMs of type $k \in K$, of demand $s_k \in (0,1]^D$, the master linear program over the patterns $a \in A$ (machine contents that fit) and the pricing problem that bounds v^- in Lemma 1 are

$$z_{MP} = \min \sum_{a \in A} x_a \quad s.t. \sum_{a \in A} a_k x_a \geq b_k \; \forall k \in K,\; x \geq 0 \qquad (9)$$

$$v^- \geq \max_{a \in Z^K_{\geq 0}} \pi^\top a \quad s.t. \sum_{k \in K} a_k s_{kd} \leq 1 \; \forall d$$

The master of Eq. (9), restricted to the patterns of the earlier iterations, is solved with the HiGHS solver through SciPy [29]; the pricing knapsack is solved exactly by a branch and bound with a linear relaxation bound, which hands the hard cases to a mixed-integer solver in a separate process; when a solver stops before proving optimality, the smaller of its relaxation and branch-and-bound upper bounds is used as v^-, which keeps Lemma 1 valid. The bound of Lemma 1 is updated at every iteration, and the procedure stops when no column improves or when the rounded-up bound reaches the rounded-up value of the master. An integer solution comes from rounding down the master solution and packing the remaining demand with First Fit Decreasing; the gap between it and the rounded-up bound brackets the optimum.

Pre-registered analysis. The analysis plan, with its hypotheses, thresholds, decision rules, validity checks and seeds, was written from the exploration week and frozen as a tagged commit before the sealed week was opened; one amendment, also committed before the first access, corrected two implementation defects that property tests had found in the exploration code and fixed operational details that the plan had left open (Supplementary Section S1; Table 3 collects the notation of the analysis). The primary outcome is the fraction of machines in use that a budget of 1% of the live VMs (a small budget for one reclamation round) certainly reclaims, measured every hour of the sealed week (T = 167 instants). The tests concern the means of

$$Y_t = \frac{f_t(\beta)}{U_t}, \theta = \frac{1}{T}\sum_{t=1}^{T} Y_t, R_t = \frac{f_t(\beta)}{g_t(\beta)}, D_t^\pi = Y_t^\pi - Y_t^{BF}, \qquad (10)$$

where $R_t = 1$ when $g_t = 0$ and π is a secondary policy. Hypothesis H1 states that the mean θ of Y_t in Eq. (10) exceeds δ = 1% of the machines; the threshold comes from the estimate of the Protean designers quoted above [4].

The series is strongly autocorrelated, so uncertainty comes from the stationary bootstrap [30] with a mean block length λ of 17.3 h, chosen on the exploration series by the method of Politis and White [31, 32] and frozen in the plan: each resample joins blocks

$$Y_q^* = (Y_{I_g}, Y_{I_g+1}, \ldots, Y_{I_g+L_g-1}), I_q \sim U\{1, \ldots, T\}, \quad L_q \sim Geom(1/\lambda), \tag{11}$$

with the indices of Eq. (11) taken modulo T, until it has T values; 10,000 resamples with seed 7919. With the bootstrap means $\theta_1^*, \ldots, \theta_B^*$, the one-sided 95% lower bound and the reported p-value are

$$\theta_{inf} = Q_\alpha(\theta_1^*, \ldots, \theta_B^*) \quad p = \frac{1 + \#\{b: \theta_b^* \leq \delta\}}{B + 1} \tag{12}$$

where Q_α is the empirical α-quantile (inverse of the empirical distribution function) and α = 0.05. With the one-sided upper bound $\theta_{sup} = Q_{1-\alpha}$ of the same resamples, the test of Eq. (12) decides

$$H1: \quad \{positive, \theta_{inf} > \delta, informative\ null, \theta_{sup} < \delta, undecided, otherwise, \tag{13}$$

and a positive result is valid if it also passes the four validity checks below. In Eq. (13), the null result is informative in the sense of a one-sided equivalence test [33]: at the same level, the data rule out a mean at or above the threshold.

The plan's precision study on the exploration week expected a bootstrap standard error $\hat{\sigma}$ of 0.70 percentage points for the mean over T instants, so the smallest mean that the test detects with power $1 - \gamma$ is

$$\theta_{MDE} = \delta + (z_{1-\alpha} + z_{1-\gamma})\hat{\sigma}, \tag{14}$$

Table 3. Notation of the statistical analysis and of the cost model.

Symbol	Meaning
Y_t; δ; θ, θ_b^*; B	primary outcome $f_t(\beta)/U_t$; relevance threshold; mean of Y_t and its value in bootstrap resample b; number of resamples
T; R_t; D_t^π	number of instants; ratio f_t/g_t (H4); paired difference of policy π from Best Fit (H5)
λ; I_q, L_q	mean block length; start and length of block q of a bootstrap resample
α; Q_α	significance level; empirical α-quantile of the bootstrap means
θ_{inf}, θ_{sup}; P, S	one-sided bootstrap bounds of the mean; share of the specifications above δ and their set
$\varepsilon_k^{(r)}$, l, R; $\eta_t(\beta)$	block signs, block length and permutations of the check on H5; machines reclaimed per hundred migrations
$\hat{\sigma}$; γ; θ_{MDE}	expected standard error of the mean; type II error rate; minimum detectable mean
k_-, k_+	numbers of bootstrap means at or below zero and at or above zero (two-sided test)

N; U^{-}; M_t	number of trace events; mean number of machines in use; placements tried in one drain
T_{replay}, T_{drain}	running time of the online replay and of one drain

where z_q is the standard normal q-quantile. With the power of 0.8 that the plan used, Eq. (14) gives θ_{MDE} = 2.75% of the machines, close to the 2.70% that the exploration week averaged.

Validity checks. V1 is a negative control: on the compact integer placement of each checkpoint, which has no fragmentation left by the online policy, Algorithm 1 without the cap must reclaim almost nothing (a mean above 0.5% of the machines invalidates the result).

V2 checks the invariants of Theorem 2 at every instant and compares the integer outputs with an independent replica written from the plan without sharing code with the pipeline, at 12 instants drawn with seed 104729 and zero tolerance. V3 is the end-time perturbation test described above. V4 requires the same decision when the outcome is measured every half hour instead of every hour.

Secondary hypotheses. H2 states that the mean excess E_t/U_t exceeds 1%; H3, a deterministic check, that the certified Gilmore–Gomory bound exceeds L_t by less than 0.1% at every checkpoint; H4, that the mean ratio f_t/g_t is at least 0.9; and H5, that the outcome of each secondary policy differs from that of BF. H2 and H4 use the p-value of Eq. (12) with their thresholds; H5 bootstraps the mean of the paired differences per instant and uses the two-sided p-value

$$p = min\{1, \frac{2(1+min(k_-,k_+))}{B+1}\} \quad k_- = \#\{b: \theta_b^* \le 0\} \quad k_+ = \#\{b: \theta_b^* \ge 0\} \tag{15}$$

with a block sign-flip permutation test as a check on Eq. (15): the mean paired difference $D^{-\pi}$ is compared with the means obtained by flipping the sign of whole blocks of l = 17 consecutive hourly instants,

$$D^{-\pi,(r)} = \frac{1}{T}\sum_{t=1}^{T} \varepsilon_{\lceil t/l \rceil}^{(r)} D_t^{\pi}, \quad p_{perm} = \frac{1+\#\{r \le R: |D^{-\pi,(r)}| \ge |D^{-\pi}|\}}{R+1}, \tag{16}$$

with independent random signs $\varepsilon_k^{(r)} \in \{-1,1\}$ and R = 10,000 permutations in Eq. (16). The six p-values of H2, H4 and H5 are adjusted with the Holm procedure [34]; the plan reserves the Benjamini–Hochberg procedure [35] for exploratory comparisons, and the exploratory analyses reported here are descriptive, without tests.

Robustness. The block length is varied to half and twice its value. A multiverse analysis [36, 37] crosses the policy (5 options), the budget (6 values from 0.1% to 5%), the priorities (all, or high only), the dimensions (those of the machine type, or cores and memory), the machine type (5 types: 12 and the four others that can host the most requests of the exploration week, leaving aside type 21, which admits the same VM types as 12) and the migration cost (number of VMs, or migrated memory, with the budget taken as the same fraction of the live memory), 1,200 specifications measured every six hours of the sealed week. It reports their specification curve and the share of specifications whose one-sided lower bound exceeds δ,

$$P = \frac{1}{|S|}\sum_{s \in S} 1[\theta_{inf}^{(s)} > \delta], \tag{17}$$

where, in Eq. (17), S is the set of specifications and $\theta_{inf}^{(s)}$ is the lower bound of Eq. (12) for specification s. With the migrated memory as the cost, the drain counts the memory of the VMs initially on each machine that it vacates, and the bound of Eq. (6) and the budget become

$$L_{mig}^{w}(\Delta) = \sum_{i=1}^{\Delta} w_{(i)}, \qquad w_j = \sum_{v:\, x(v)=j} w_v, \; m^w = \beta \sum_{v \in V_t} w_v, \tag{18}$$

where w_v is the memory of VM v and $w_{(i)}$ the i-th smallest machine weight w_j. With unit weights, Eq. (18) gives back the bound of Eq. (6) and the budget m.

External replication. The same estimator is applied to the Azure public VM trace of 2019 [20], which records creation and deletion times and the core and memory buckets of each VM. Each VM takes the value of its bucket (twice the lower limit for the open bucket), and three machine shapes are tried, from 48 cores and 384 GB to 96 cores and 768 GB; the replay uses BF on these two resources, and the outcome is measured at 167 hourly instants from day 7 to day 14 after the first creation of the trace, as on the sealed week. This is a secondary analysis of a different trace, not a validity check.

Exact baselines. Two integer programs, solved with the COIN-OR Branch and Cut solver (CBC) through PuLP, measure what the bounds leave out on small instances. The first packs random subsets of the live VMs of each checkpoint (from 25 to 800 VMs, 5 replicates, a limit of 600 s) with the assignment constraints of Eq. (3), the objective $\sum_j u_j$, no limit on the machines and the symmetry-breaking constraints $u_j \geq u_{j+1}$, and compares its optimum and time with the certification. The second solves Eq. (3) itself on blocks of consecutive machines of the real state (from 4 to 32 machines) for every reclaimable number of machines, and compares its optimum with Algorithm 1 and with L_{mig}.

Benchmark. Following the rules of Hoefler and Belli [38], each stage of the pipeline is timed in a fresh single-threaded process after a discarded warm-up run (the first call compiles the numerical kernels), with the median and interquartile range of repeated runs, the processor time and the peak resident memory. Both the online replay and the drain scan the machines in use for every placement, so their costs are

$$T_{replay} = O(N\, U^{-} D), \qquad T_{drain} = O(M_t\, U_t D) \tag{19}$$

where N is the number of events, U^{-} the mean number of machines in use and M_t the number of tentative placements of a drain, which the budget bounds together with the failed drains. Since the machines in use grow in proportion to the live VMs, both costs are quadratic in the size of the trace by Eq. (19); the measured exponent is the slope of a log-log fit over random subsets of the VMs of the exploration week, from one thirty-second of them to all of them, doubling each time.

Software and verification. The pipeline (Figure 3) is written in Python 3.10.9 with NumPy [39], SciPy and numba [40]; the exact baselines use CBC through PuLP 2.7.0. Every stage writes its results with their provenance (commit, digest of the data, plan tag, seed and library versions). Property-based tests written with Hypothesis [41] check Lemmas 2 and 3 and Theorem 2 against an exhaustive MinMig on random small instances; these tests found the two defects that the amendment corrected. Automated checks stop the build when a number in the manuscript does not come from a results file or when a cited sentence lacks a verified supporting passage, and the manuscript, its tables and its figures are generated from the results by code.

3. Results

Replay of the sealed week. Machine type 12 hosts 5,321,338 of the 5,559,800 requests; the other 238,462 (4.3%) are of VM types that it cannot host and were excluded, as the plan states. Over the 167 hourly instants of the sealed week, Best Fit kept between 90,310 and 92,096 machines in use for 777,655 to 820,096 live VMs (Figure 4a). Besides the analyses of the plan, this section reports descriptive counts and summaries that were added after the results were known, without tests; Supplementary Section S1 lists them.

Confirmatory result. With a budget of 1% of the live VMs, Algorithm 1 reclaimed on average 5.62% of the machines in use ($\hat{\theta}$; one-sided 95% lower bound 5.38%, $p = 0.00010$ by Eq. (12), the smallest value attainable with 10,000 resamples). The lower bound exceeds the threshold of 1%, so the result is positive, as it would be with any threshold below 5.38%, and it passes the four validity checks: by the rule of the plan it is valid.

Static optimum and negative control. At the 7 checkpoints, column generation brackets the static optimum within 1 machine (Table 4): the certified lower bound of Lemma 1 never exceeds the size-sum bound by more than 0.0012% of it and equals it at 2 checkpoints, so H3, which requires less than 0.1%, holds. Since $L_t \leq OPT_t \leq U^*$, at the checkpoints the excess E_t exceeds the excess over the static optimum, $U_t - OPT_t$, by at most 2 machines. On the compact integer placements, Algorithm 1 without the cap reclaimed no machine at any checkpoint (V1 invalidates the result above 0.5%).

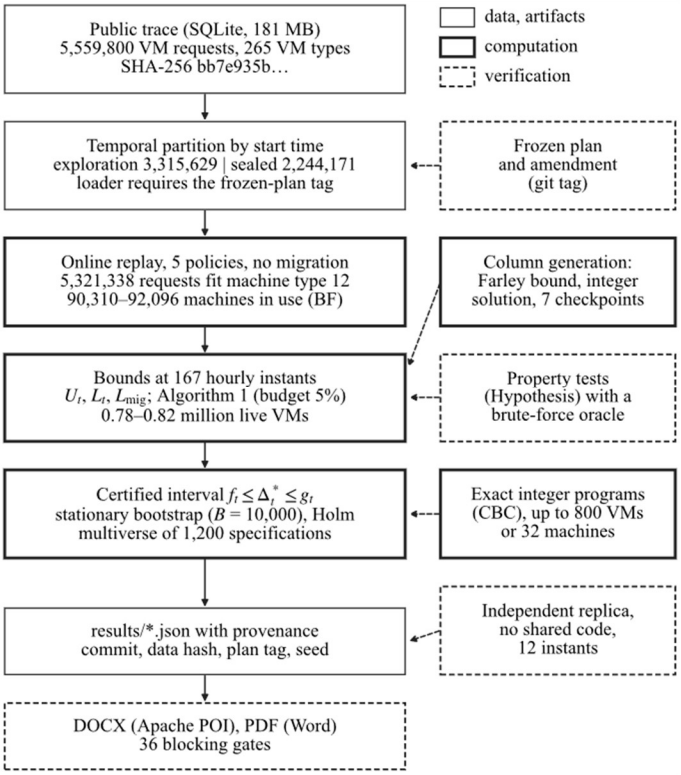


Figure 3. Pipeline of the study, from the public trace to the manuscript (solid arrows), with the size of the data at each stage. The boxes on the right feed or check the stage they point at: the frozen plan gates the sealed week, column generation brackets the static optimum, property tests with a brute-force MinMig oracle check Algorithm 1 and the bounds, exact integer programs measure the bounds on small instances, and an independent replica recomputes the primary outcome. Thin border: data and artifacts; thick border: computation; dashed border: verification.

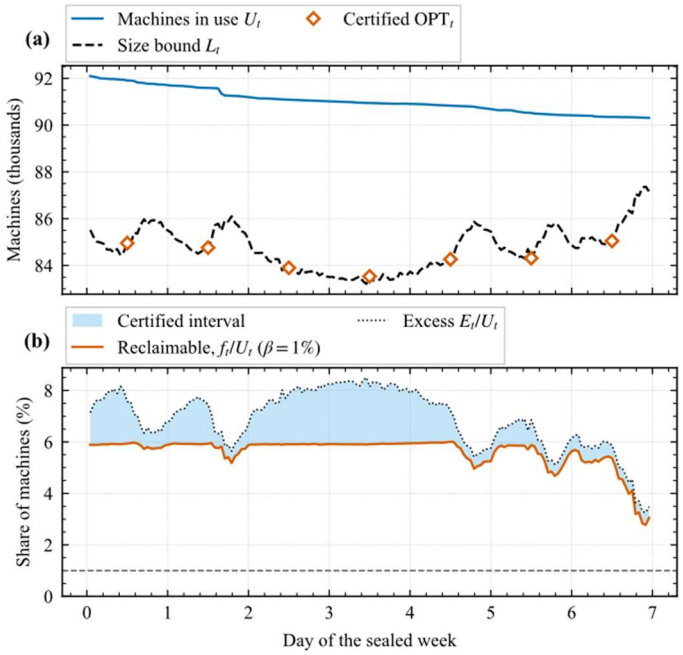


Figure 4. The sealed week under Best Fit. (a) Machines in use U_t, size-sum bound L_t and, at the daily checkpoints, the static optimum certified to within 1 machine (Table 4). (b) Share of the machines that a budget of 1% of the live VMs certainly reclaims (f_t/U_t) and its certified upper bound (g_t/U_t, top of the band), which at this budget equals the excess E_t/U_t at every instant; the dashed line is the threshold δ.

Table 4. Certification of the static optimum at the checkpoints of the sealed week: size-sum bound L_t, certified lower bound of Lemma 1 rounded up, machines U^* of the integer solution, column-generation iterations and time of the certification, and machines that Algorithm 1 reclaims from the compact placement (check V1). Day: day of the sealed week.

Day	L_t	Bound	U^*	Iter.	Time (s)	V1
0.5	84,955	84,956	84,957	370	56	0
1.5	84,760	84,761	84,762	354	35	0
2.5	83,896	83,897	83,898	416	46	0
3.5	83,531	83,532	83,533	395	189	0
4.5	84,260	84,261	84,262	351	67	0
5.5	84,310	84,310	84,311	355	77	0
6.5	85,045	85,045	85,046	323	32	0

Validity checks. No machine exceeded its capacity after the drain, $f_t \leq g_t \leq E_t$ held and both bounds grew with the budget at all 1,170 measured instants of the hourly series of the five policies and the half-hourly series of BF, and the independent replica reproduced the live VMs, the machines in use, the size-sum bound, the excess and the reclaimable machines exactly at 12 of its 12 instants (V2; Supplementary Table S3). Redrawing the end times that fall after the tested instant left the placement of every VM that had arrived by then unchanged at all 3 tested instants, by which 4,224,225 VMs had arrived on average (V3). Measured every half hour, the outcome has mean 5.60% and one-sided lower bound 5.35%, the same decision as the hourly series (V4).

Secondary hypotheses. Figure 5 gathers the estimates and intervals of every pre-registered test. The mean excess of the online placement over the size-sum bound is 6.89% of the machines in use (one-sided lower bound 6.35%, Holm p = 0.00060), so H2 is supported. The greedy drain achieves on average a fraction 0.83 of what the certified upper bound allows (one-sided lower bound 0.79, threshold 0.9, Holm p = 1.000), so H4 is not supported. Against Best Fit, the mean difference in the reclaimable share is: FF −0.94 pp (95% interval −1.19 to −0.66, Holm p = 0.0010); BF-L1 −0.53 pp (95% interval −0.74 to −0.29, Holm p = 0.0010); WF −1.42 pp (95% interval −1.54 to −1.25, Holm p = 0.0010); DP −1.39 pp (95% interval −1.54 to −1.19, Holm p = 0.0010). After the Holm adjustment, all four policies differ from Best Fit, and all reclaim less than it (H5). The block sign-flip permutation test agrees with the bootstrap in every comparison. Table 5 gathers the criterion and the decision of every pre-registered test and check. Table 6 compares the five policies and Figure 6 follows them over the week. The size-sum bound depends only on the live VMs, which are the same under every policy, so the excess follows the machines in use, from 5.38% of them under FF to 15.09% under WF: a policy that packs more tightly leaves a smaller excess to reclaim. The excess is not what a budget reclaims: WF leaves the largest excess, yet 1% of the live VMs reclaims on average 4.19% of its machines, against 5.62% under BF. Under every policy the machines that host a single VM outnumber the budget at least 2.8 times at every instant, and the drain visits them first; yet it reclaims fewer machines than the budget allows, because at least 78% of those machines could not be emptied: their VM fit on no other machine in use.

Table 5. Pre-registered hypotheses and validity checks: criterion of the plan, result (estimate and, in parentheses, its one-sided lower bound) and decision. H2, H4 and H5 use Holm-adjusted p-values.

Test	Criterion	Result	Decision
H1	lower bound of θ above 1%	5.62% (5.38%)	positive, valid
V1	reclaims at most 0.5% of a compact placement	0 machines	passed
V2	invariants hold; replica equal at 12 instants	1,170 instants; 12 of 12	passed
V3	no placement changes when later end times change	3 of 3 instants unchanged	passed
V4	same decision every half hour	5.60% (5.35%)	passed
H2	lower bound of mean E_t/U_t above 1%	6.89% (6.35%)	supported
H3	Lemma 1 bound within 0.1% of L_t	at most 0.0012%	holds
H4	mean R_t at least 0.9	0.83 (0.79)	not supported
H5	mean D_t^{π} differs from zero	−1.42 to −0.53 pp	all four below BF

Table 6. Online policies on the sealed week: mean machines in use U_t, mean excess E_t/U_t, mean share that a budget of 1% of the live VMs certainly reclaims f_t/U_t, mean share of machines that host a single VM, and instants (of 167) at which the policy reclaims less than BF.

Policy	U_t (k)	E_t/U_t	f_t/U_t	One VM	Below BF
BF	91.0	6.89%	5.62%	28.1%	–
BF-L1	90.0	5.86%	5.09%	28.4%	124
FF	89.6	5.38%	4.67%	28.5%	167
WF	99.8	15.09%	4.19%	23.9%	162
DP	99.2	14.58%	4.23%	23.7%	162

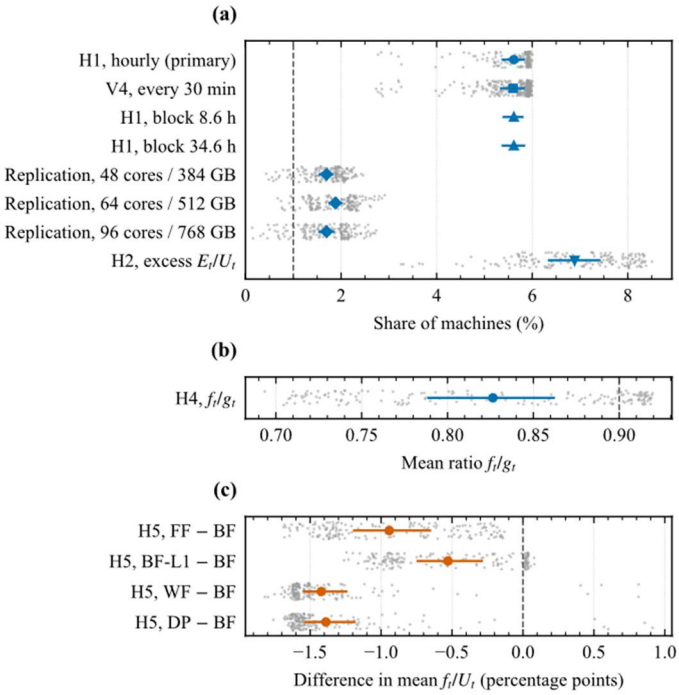


Figure 5. Pre-registered tests. (a) Mean reclaimable share with a budget of 1% (H1, the half-hourly check V4, the block sensitivity and the external replication by machine shape) and mean excess (H2); the bars span the one-sided lower and upper bounds (5th and 95th percentiles of 10,000 stationary-bootstrap means), and the dashed line is δ. (b) Mean ratio f_t/g_t (H4) against its threshold. (c) Mean paired difference of each policy from BF (H5) with its two-sided 95% interval. Grey dots: the value at each measured instant behind its mean (hourly; half-hourly for V4).

Robustness. With half and twice the block length, the one-sided lower bound of H1 is 5.40% with a block of 8.6 h and 5.38% with a block of 34.6 h (the Politis–White method gives 15.2 h on the sealed series, inside that range), which does not change the decision. The outcome itself exceeded the threshold at all 167 instants (minimum 2.77%), and the excess at all 167 (minimum 3.25%), so the decisions on H1 and H2 do not depend on the bootstrap, whose stationarity assumption the series does not meet (Supplementary Section S2); nor does the decision on H4, whose mean ratio, 0.83, is itself below 0.9. FF reclaimed less than BF at every instant, WF and DP at 162 of the 167, and BF-L1 at 124, so only for FF does the decision of H5 not rest on the inference; for the other three it rests on the bootstrap, with which the test agrees.

Across the multiverse, 69.2% of the 1,200 specifications have a lower bound above the threshold (Figure 7); Table 7 breaks the share down by factor and Figure 8 shows how pairs of factors interact. Over the 2,700 measured instants of the 100 simulations behind the multiverse, a budget of 1% reclaimed on average 54.9% of the excess and never all of it (Figure 9a); counted in migrated memory, the same budget reclaimed on average 52.3% of what it reclaimed counted in VMs, and a budget of 5%, 82.0% (Figure 9b). Every specification with a budget of 2% or more passes. By construction, 40 specifications, all with a budget of 0.1% counted in VMs, cannot pass: each reclaimed machine costs at least one migration, so the outcome is at most $\beta n_t/U_t$, which stays below δ at every instant of those specifications. Among the other 1,160, 71.6% have a lower bound above the threshold. Besides the budget, the migration cost moves the share of positive specifications the most, from 56% (cost in migrated memory) to 82% (cost in VMs); the policy, the priorities, the dimensions and the machine type move it by at most 22 percentage points.

Table 7. Multiverse by factor: number of specifications, share whose one-sided lower bound exceeds the threshold and median estimate, for each level of each factor. Every specification uses the stationary bootstrap with the frozen block length converted to six-hour instants (2.9 instants).

Factor	Level	Specs	Positive	Median
Policy	BF	240	74%	1.81%
	BF-L1	240	76%	1.80%
	DP	240	64%	1.82%
	FF	240	72%	1.91%
	WF	240	60%	1.54%
Budget	0.1%	200	16%	0.54%
	0.25%	200	50%	1.15%
	0.5%	200	66%	1.84%
	1%	200	84%	2.63%
	2%	200	100%	3.79%
	5%	200	100%	5.30%
Priorities	high only	600	58%	1.36%
	all	600	80%	3.26%
Dimensions	cores, memory	600	70%	1.86%
	declared	600	68%	1.81%
Machine type	6	240	65%	1.70%
	8	240	72%	2.16%
	12	240	63%	1.62%
	16	240	76%	2.17%
	31	240	70%	1.81%
Migration cost	migrated memory	600	56%	1.35%
	VM count	600	82%	2.28%

External replication. On the Azure trace of 2019 (2,695,548 VMs, of which the 144,461 whose deletion equals their creation are excluded, as the amendment fixed; every other VM fits every machine shape), the same estimator gives, for machines of 48 cores and 384 GB: 1.69% (lower bound 1.56%); 64 cores and 512 GB: 1.88% (lower bound 1.76%); 96 cores and 768 GB: 1.70% (lower bound 1.55%).

The result is positive for all three machine shapes.

With a budget of 5%, Algorithm 1 vacated the whole excess at 151, 164 and 165 of the 167 instants, respectively, which by the Corollary certifies $OPT_t = L_t$ there; on the sealed week of the packing trace it never did.

Figure 10 follows the three machine shapes over the week.

Exact baselines. The direct integer program reached a proven optimum in every replicate up to 200 VMs and in 29 of the 30 instances overall, always equal to the rounded-up certified bound of column generation where both are known; the only exception, with 400 VMs, reached the time limit, and column generation certified its optimum in 0.24 s (Figure 11a).

Of the 20 blocks of consecutive machines drawn from the real state (5 of each size from 4 to 32 machines), 4 had machines to vacate (one of 8 machines, two of 16 and one of 32); in the others the size-sum bound equals the number of machines.

In their 20 cases (vacating 1 to 12 machines), the greedy drain and the lower bound L_{mig} both equaled the exact MinMig each time, so the bracket is exact there (Figure 11b).

These substates are small, and on the full states the drain stops short of the whole excess, as the exploratory analyses below show, so they do not show the gaps that larger instances may have.

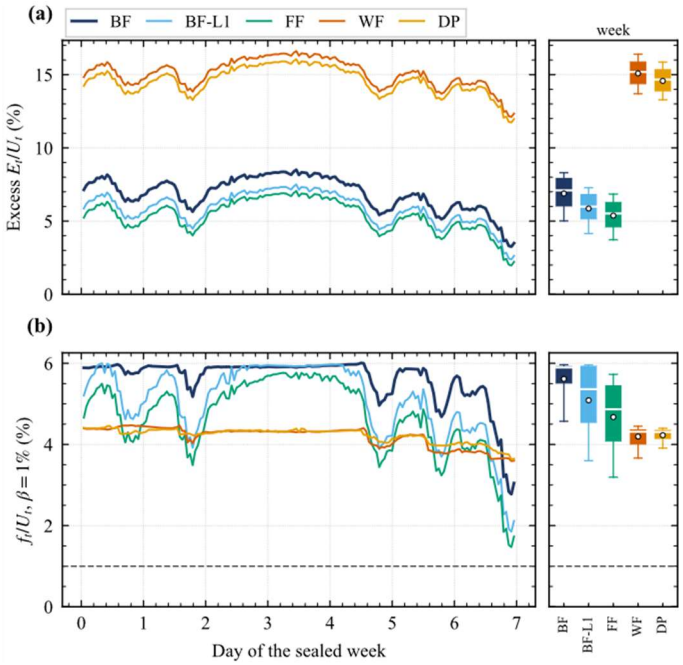


Figure 6. The five online policies over the sealed week. (a) Excess E_t/U_t. (b) Share of the machines that a budget of 1% of the live VMs certainly reclaims, f_t/U_t; the dashed line is δ.

At the right of each panel, the distribution of each policy over the week on the same scale: box from the first to the third quartile, white line at the median, whiskers at the 5th and 95th percentiles and the mean as a dot.

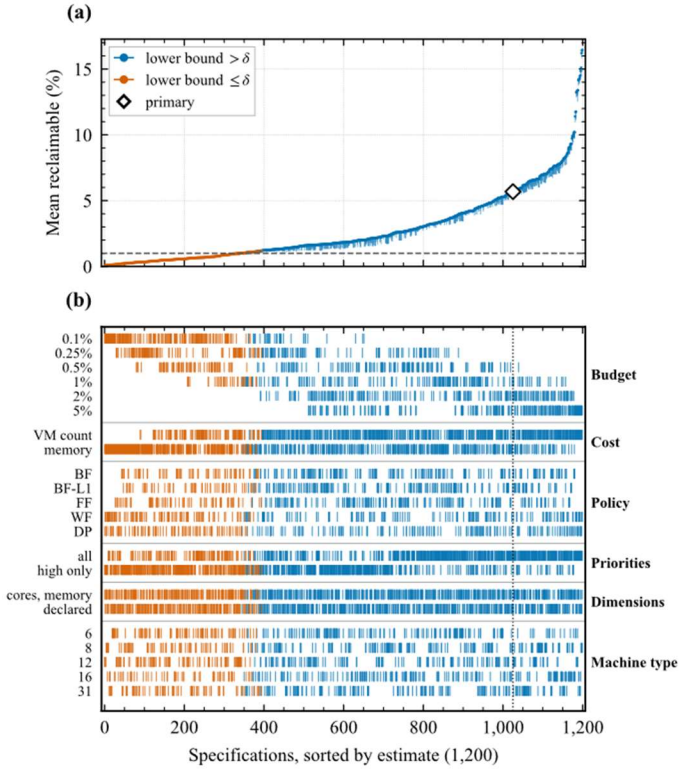


Figure 7. Specification curve of the multiverse (1,200 specifications) with its decision panel. (a) Estimate (dot) and one-sided lower bound (bottom of each bar) of the mean reclaimable share, sorted by the estimate, in blue when the lower bound exceeds δ (dashed) and in vermillion otherwise; the diamond marks the primary specification. (b) For each specification, a mark in the row of the level that it takes in each of the six factors, with the same colors: the columns of marks show which choices the low estimates share; the dotted line is the primary specification.

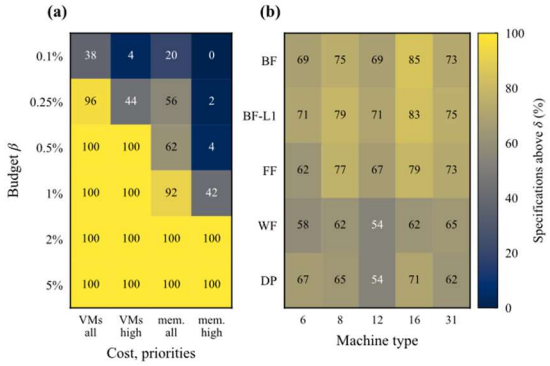


Figure 8. Interactions in the multiverse that the marginal shares of Table 7 do not show: percentage of the specifications whose one-sided lower bound exceeds δ (a) by budget and by migration cost (VM count or migrated memory) and priorities (all VMs or high priority only), 50 specifications per cell, and (b) by policy and machine type, 48 specifications per cell.

Computational cost. Table 8 lists the cost of each stage at one instant of the exploration week (day 5, after replaying 3,177,096 requests: 831,619 live VMs on 95,744 machines); the direct integer program runs on a random subset of 50 of its VMs.

Measuring an instant from its placement (the machines in use, the size-sum bound and the drain) took about 21 s on one thread; the replay up to the instant, which dominates the cost here, is needed only because this study reconstructs the placement from the trace, and an allocator already knows it. The fitted log-log slopes of the wall-clock time against the number of VM requests are 1.91 for the online replay and 1.83 for the drain (Figure 12), against the slope of two that Eq. (19) predicts when the machines in use grow with the trace.

Table 8. Cost per stage (single thread, fresh process, warm-up discarded): median and interquartile range (IQR) of the wall-clock time, median processor (CPU) time and peak resident memory of the process, which loading the trace dominates. Processor time counts only the measuring process: the mixed-integer solvers that column generation and the direct integer program call run in processes of their own. The direct integer program packs a random subset of 50 VMs of the instant.

Stage	Time (s)	IQR (s)	CPU (s)	Memory (MiB)
Load the trace	7.42	0.02	7.36	1,936
Build the events	0.49	0.01	0.48	1,935
Online replay to the instant	215.17	0.34	214.77	1,935
Measure U and L	0.02	0.00	0.02	1,936
Greedy drain (Algorithm 1)	21.18	0.08	21.17	1,936
Column generation	85.16	0.31	1.42	1,936
Direct integer program	0.06	0.00	0.02	1,935

Exploratory analyses. The following summaries were not hypotheses of the plan; they describe the confirmatory series with two-sided 95% bootstrap intervals and no tests (Figure 13; Figure 14 maps them instant by instant), and some of them were added after the results were known (Supplementary Section S1). The efficiency of a budget is the number of machines that Algorithm 1 vacates per hundred migrations of it,

$$\eta_t(\beta) = \frac{100\, f_t(\beta)}{max(1, m)}, \tag{20}$$

where $m = \lfloor \beta n_t \rfloor$ is the budget in migrations; Eq. (20) averaged 100.0 machines with a budget of 0.1% of the live VMs and 14.9 with 5%. The certified interval closed ($f_t = g_t$, so Δ_t^* is known exactly) at all 167 instants with 0.1% and at 159 instants with 0.25%. The drain runs out of machines that it can vacate before it runs out of budget: with 5% it tried every machine in use at all 167 instants without reaching the budget, so it reclaimed what it would reclaim with no budget limit, on average 94.0% of the excess (95% interval 93.2% to 94.7%) and never all of it; a budget of 2% already reclaimed as much at 133 instants. The migration bound does not rule out reclaiming the whole excess with this budget: it requires at least 0.79% of the live VMs on average ($L_{mig}(E_t)$).

At the checkpoints the static optimum lies within 2 machines of L_t, so a full repacking would reclaim all of the excess except at most 2 machines, at a number of migrations that this study does not bound from above. On average 28.1% of the machines in use host a single VM.

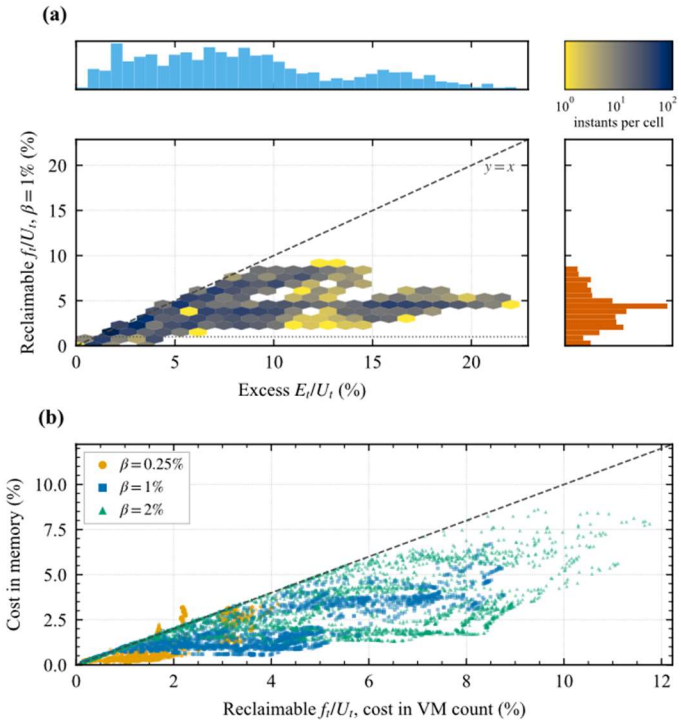


Figure 9. Anatomy of the multiverse over its 2,700 measured instants (100 simulations of six-hourly instants; descriptive). (a) Joint density, on a logarithmic color scale, of the excess and of the share of the machines that a budget of 1% of the live VMs certainly reclaims, with the cost in VM count; the histograms give each marginal distribution, the dashed line is $y = x$ (the whole excess) and the dotted line is δ. (b) The same share with the cost in migrated memory against the cost in VM count, on the same state, for three budgets; the dashed line is $y = x$.

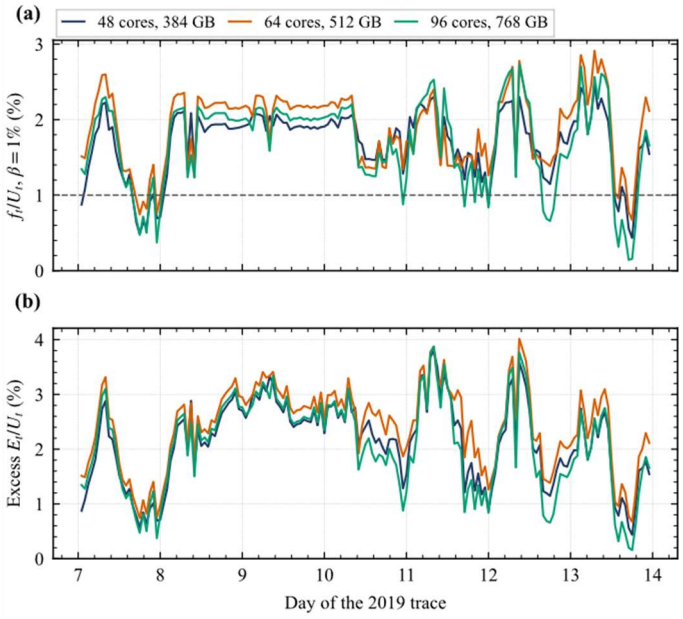


Figure 10. External replication on the Azure trace of 2019 (Best Fit on cores and memory) by machine shape, over its hourly instants from day 7 to day 14: (a) share of the machines that a budget of 1% of the live VMs certainly reclaims, f_t/U_t, with δ dashed; (b) excess E_t/U_t.

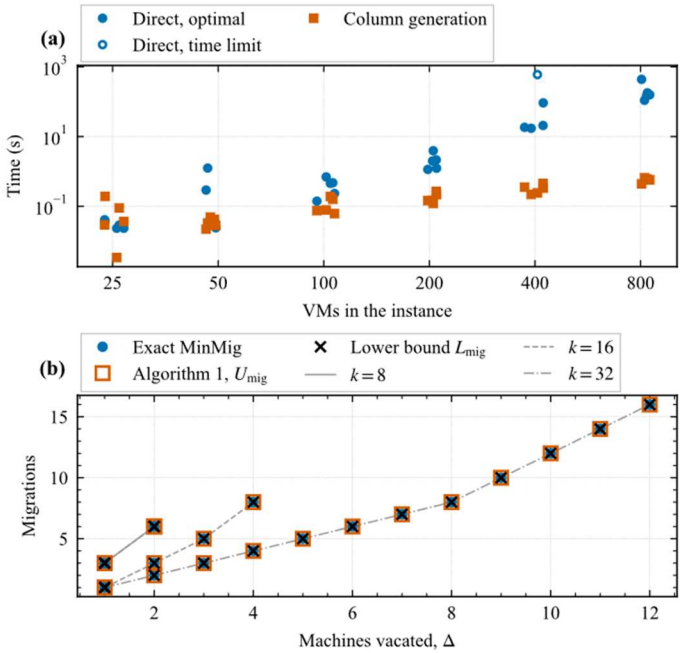


Figure 11. Exact baselines. (a) Time of the direct integer program (CBC; open markers reached the time limit) and of the column-generation certification against the number of VMs, one marker per instance. (b) Migrations needed to vacate Δ machines in blocks of k consecutive machines of the real state: exact MinMig (dots), Algorithm 1 (open squares) and the lower bound L_{mig} (crosses), one line per block.

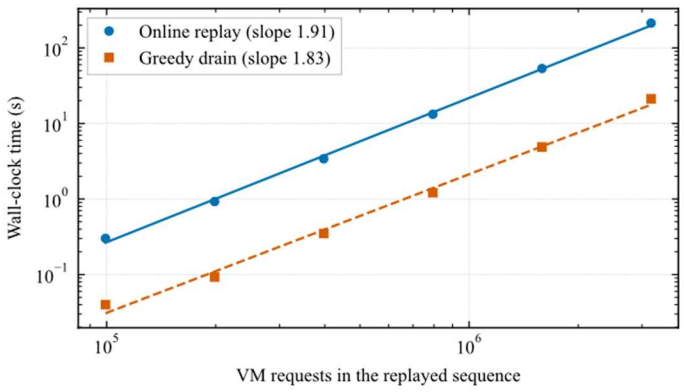


Figure 12. Wall-clock time of the online replay and of Algorithm 1 against the number of VM requests, on random subsets of the exploration week (log-log): markers give the median of repeated single-threaded runs, and lines the power law fitted to them, whose slope the legend gives.

4. Discussion

What the bounds measure. Under Best Fit, a one-shot budget of 1% of the live VMs, about 8,000 migrations at a given instant, reclaims for certain 5.6% of the machines in use on average, about 5,100 machines. The test was not a foregone conclusion: on the exploration week, on which the plan was designed, the same estimator gave a one-sided lower bound of 1.49%, close to δ, and the sealed week reclaimed on average 2.1 times as much. At this budget the upper bound is the excess itself: the migration bound of Eq. (6) does not rule out reclaiming every excess machine with 1% of the VMs, so $g_t = E_t$ at all 167 instants. The migration bound binds with smaller budgets, and with 0.1% it meets the drain at all 167 instants, where Δ_t^* is therefore known exactly.

Because the certified static optimum stays within 2 machines of the size-sum bound at the checkpoints, the second term of Eq. (2) is at most that there: the excess is fragmentation left by the online policy and not slack in the bound, and a budget of 1% recovers on average 83% of it (H4: mean ratio 0.83 of the drain to the upper bound). The plan expected a ratio of at least 0.9, which the data do not support: at this budget the drain is still limited by the budget at 162 of the 167 instants, and the upper bound is the excess itself, so the gap mixes migrations that the budget does not allow with machines that the bound cannot rule out.

Implications for operators. The reclaimable share sits well above the 1% that the designers of Azure's allocator associate with savings of the order of a hundred million dollars per year at their scale [4], although that estimate refers to their fleet and not to one replayed zone. The budget curve shows diminishing returns. With 0.1% of the VMs every migration vacated a machine, which only a machine hosting a single VM allows, and 28.1% of the machines host one. Beyond a budget of 2%, more budget buys almost nothing, because the drain stops before using it. The rest of the excess, 360 machines on average, is out of reach of Algorithm 1, which moves VMs only out of the machines that it empties and never rearranges the machines that stay in use. The online policy also moves the baseline: under First Fit the replayed zone kept on average 1.6% fewer machines than under Best Fit without any migration, comparable to what Best Fit recovers with a budget between 0.1% and 0.25% of the live VMs. Each migration also has a cost in bandwidth, time and service level [6, 8], so the curve, rather than any single point, is what an operator needs to choose a budget.

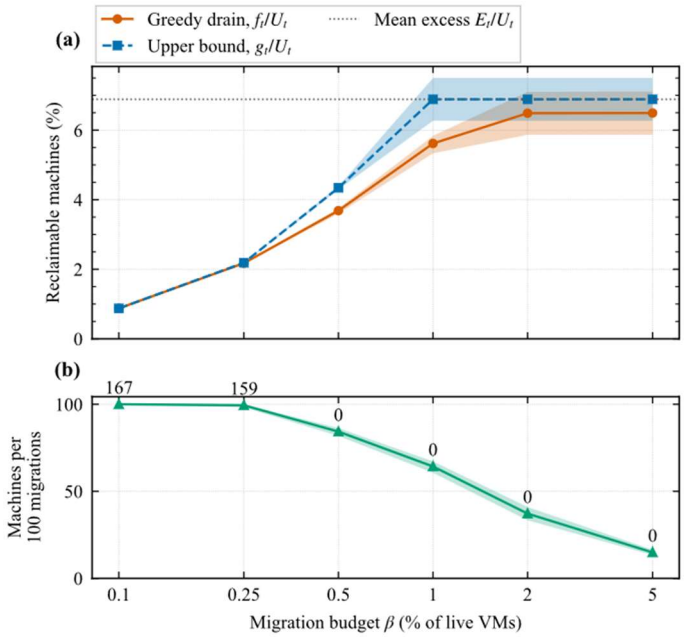


Figure 13. Exploratory budget curve. (a) Mean share of the machines reclaimed by Algorithm 1 (f_t/U_t) and its certified upper bound (g_t/U_t) against the budget, with two-sided 95% bootstrap bands, and the mean excess. (b) Machines reclaimed per 100 migrations of budget; the numbers give the instants, of 167, at which the certified interval closes ($f_t = g_t$).

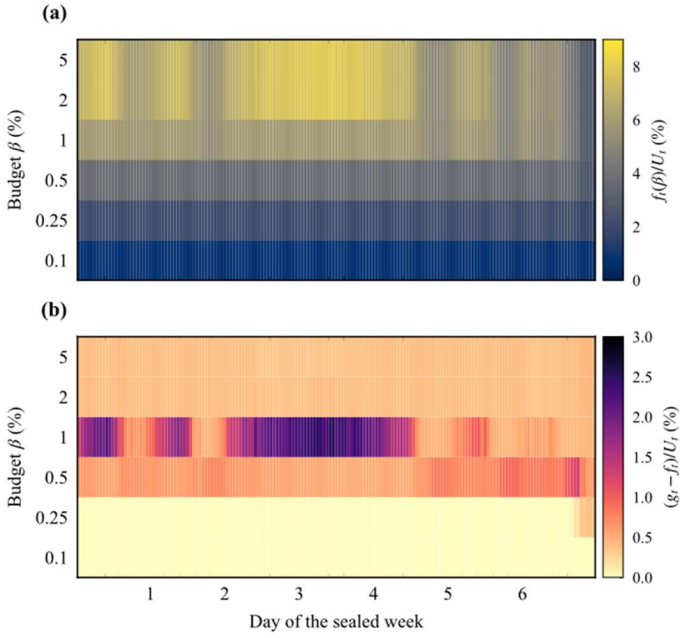


Figure 14. Budget curve instant by instant on the sealed week under Best Fit (exploratory): (a) share of the machines that each budget certainly reclaims, $f_t(\beta)/U_t$; (b) width of the certified interval, $(g_t(\beta) - f_t(\beta))/U_t$, which is zero where the interval closes.

Relation to prior work. Consolidation studies minimize or limit migrations on simulated data centers and workload traces [10–12], and the theory of dynamic bin packing bounds the repacking per arrival or departure in the worst case [14, 17, 19]. The measurement here sits between the two: it fixes a real sequence of requests and an online

policy, and brackets the minimum number of migrations needed to reclaim the fragmentation that this policy leaves, with a lower bound that holds for any repacking algorithm. Both sides of the bracket are computable at the scale of a production zone, which is what makes the measurement possible; the exact minimum is not.

Threats to validity. Table 9 lists the threats that the plan anticipated and how the study addresses each of them.

Limitations and future work. The cost model counts migrations, or the migrated memory, but not their duration, their order or the network they share, and the drain ignores the fault-domain and placement rules that production allocators also enforce [4]. Under such rules the lower bound on migrations still holds, because it bounds every repacking, but the reclamation that Algorithm 1 achieves may become infeasible. Each instant is measured as a one-shot repacking of the state that the online policy left; the study does not simulate a reclamation process that migrates while requests keep arriving, how long the migrations of a budget take to execute, nor how the policy would use the reclaimed machines afterwards. The replay uses one machine type for the whole zone, whereas production zones mix hardware generations [4]; with several machine types the static problem is vector bin packing with heterogeneous bins [2], and the bounds would have to weigh which machines are vacated.

The bootstrap intervals describe the hour-to-hour variability of one week of one zone, not the variability between zones or periods, and the two weeks of the trace differ: on the exploration week, on which the plan was designed, the excess averaged 3.56% of the machines and the same budget reclaimed 2.70% (Supplementary Table S1), against 6.89% and 5.62% on the sealed week.

The difference lies in the first days of the replay (Figure 15), which starts from an empty zone: on each of days 1 to 4 of the exploration week the outcome averaged between 0.65% and 1.97% of the machines, and on days 5 and 6, 5.63% and 5.88%, within the range of the days of the sealed week (4.65% to 5.92%).

The excess of the exploration week rose from 0.68% to 4.95% of the machines within one hour on day 4.8, while the live VMs fell from 854,072, the most at any measured instant of that week, to 833,186, and stayed above 4.5% afterwards, so how much a budget reclaims can depend on the recent history of the demand, which one week of one zone does not span. With budgets of 1% and more, the migration bound does not bind and the upper side of the interval is the excess itself; a stronger lower bound on MinMig would narrow it, and the linear relaxation of Eq. (3) strengthened by valid inequalities is a natural candidate. The independent replica shares only the plan with the pipeline, so it guards against errors of implementation, not against a misreading of the plan. Most VMs in such traces live minutes while some stay for weeks [4], and VM behavior is fairly consistent across the lifetimes of a customer's VMs [20]; allocators that use known or predicted lifetimes [42, 43] could leave less to reclaim, and the same bounds can measure by how much.

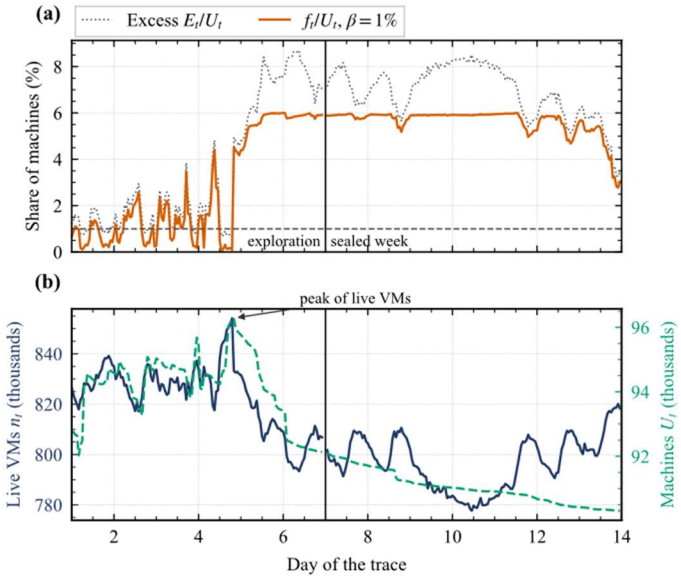


Figure 15. The two weeks of the trace under Best Fit, hour by hour: (a) share of the machines that a budget of 1% of the live VMs certainly reclaims (f_t/U_t) and excess (E_t/U_t), with δ dashed; (b) live VMs n_t and machines in use U_t (right axis). The vertical line separates the two weeks; the arrow marks the measured peak of live VMs in the exploration week.

Table 9. Threats to validity, by type (in italics), and how the study addresses them.

Threat	**Mitigation**
Construct. Counting VMs ignores the size and duration of each migration	Multiverse with the migrated memory as the cost and the budget as a share of the live memory
Construct. Reclaimed machines may be kept empty as operational reserves	Results read as a potential; the budgets studied are small
Internal. The placement comes from a replayed policy; the production placement was not published	Five online policies (H5) and their multiverse
Internal. Low-priority lifetimes depend on evictions by the production allocator	Multiverse restricted to high-priority VMs
Internal. Future end times could leak into the placement	Perturbation test V3 on the sealed data
Internal. Implementation errors in the simulator, the drain or the bounds	Property tests against brute force; independent replica (V2); exact baselines
External. One production zone, two weeks and one machine type	Replication on the 2019 trace with three machine shapes; five machine types in the multiverse
Conclusion. Strong autocorrelation and non-stationarity of the hourly series	Stationary bootstrap with a frozen block length, block sensitivity, half-hourly check V4, diagnostics; the outcome exceeds the threshold at every instant
Conclusion. Multiplicity and analytic flexibility	Frozen plan and sealed week, one amendment before access, Holm adjustment, exploratory analyses without tests

5. Conclusions

Reclaiming the machines that online allocation leaves fragmented is a strongly NP-hard problem that admits no finite approximation unless P = NP, but its value can be bracketed at production scale: a greedy drain gives what a budget certainly buys, and an elementary bound on migrations gives what no algorithm can exceed. On the sealed week of the Azure packing trace replayed under Best Fit, a budget of 1% of the live VMs reclaimed for certain 5.6% of the machines in use on average, under a plan frozen before the data were opened, with validity checks, a multiverse of specifications, an external replication and exact baselines. With 0.1% of the VMs the bracket closes at every instant, so the number of machines that the budget reclaims is known exactly; with any budget of the plan, the greedy drain stops short of the whole excess, which leaves the last part of the fragmentation to repacking. The bounds need only a placement and the VM sizes, so they apply to any allocator whose placements can be replayed or logged, including lifetime-aware ones, and the code that produces every number of this paper is part of the study.

6. Declarations

Funding. This research received no external funding.

Author contributions. The author is the sole contributor: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, visualization and writing (original draft, review and editing).

Conflict of interest. The author declares no conflict of interest.

Data availability. The Azure VM packing trace of 2020 and the Azure public VM trace of 2019 are public under the CC BY 4.0 license in the Azure Public Dataset repository (https://github.com/Azure/AzurePublicDataset). The frozen analysis plan and its amendment and the results files with their provenance are available from the author on request.

Ethics. The study uses only public, anonymized traces of computing systems; it involves no human participants or personal data and required no ethics approval.

REFERENCES

1. R. Panigrahy, K. Talwar, L. Uyeda and U. Wieder. "Heuristics for Vector Bin Packing". Microsoft Research, 2011. Available at: https://www.microsoft.com/en-us/research/publication/heuristics-for-vector-bin-packing/
2. M. Gabay and S. Zaourar. "Vector bin packing with heterogeneous bins: application to the machine reassignment problem", *Annals of Operations Research*, 242 (1), pp. 161–194, 2016. https://doi.org/10.1007/s10479-015-1973-7
3. E. G. Coffman Jr., M. R. Garey and D. S. Johnson. "Dynamic Bin Packing", *SIAM Journal on Computing*, 12 (2), pp. 227–258, 1983. https://doi.org/10.1137/0212014
4. O. Hadary, L. Marshall, I. Menache, A. Pan, E. E. Greeff, D. Dion et al. "Protean: VM Allocation Service at Scale". In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pp. 845–861, 2020. Available at: https://www.usenix.org/conference/osdi20/presentation/hadary
5. C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt and A. Warfield. "Live Migration of Virtual Machines". In *2nd Symposium on Networked Systems Design & Implementation (NSDI 05)*, 2005. Available at: https://www.usenix.org/conference/nsdi-05/live-migration-virtual-machines
6. W. Voorsluys, J. Broberg, S. Venugopal and R. Buyya. "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation", in *Cloud Computing* (Lecture Notes in Computer Science), pp. 254–265, Springer Berlin Heidelberg, 2009. https://doi.org/10.1007/978-3-642-10665-1_23
7. A. Strunk. "Costs of Virtual Machine Live Migration: A Survey". In *2012 IEEE Eighth World Congress on Services*, pp. 323–329, 2012. https://doi.org/10.1109/services.2012.23
8. M. E. Elsaid, H. M. Abbas and C. Meinel. "Virtual machines pre-copy live migration cost modeling and prediction: a survey", *Distributed and Parallel Databases*, 40 (2–3), pp. 441–474, 2022. https://doi.org/10.1007/s10619-021-07387-2
9. F. Hermenier, X. Lorca, J.-M. Menaud, G. Muller and J. Lawall. "Entropy: a consolidation manager for clusters". In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, pp. 41–50, 2009. https://doi.org/10.1145/1508293.1508300
10. A. Beloglazov and R. Buyya. "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers", *Concurrency and Computation: Practice and Experience*, 24 (13), pp. 1397–1420, 2012. https://doi.org/10.1002/cpe.1867
11. T. C. Ferreto, M. A. Netto, R. N. Calheiros and C. A. De Rose. "Server consolidation with migration control for virtualized data centers", *Future Generation Computer Systems*, 27 (8), pp. 1027–1034, 2011. https://doi.org/10.1016/j.future.2011.04.016

12. C. Ghribi, M. Hadji and D. Zeghlache. "Energy Efficient VM Scheduling for Cloud Data Centers: Exact Allocation and Migration Algorithms". In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pp. 671–678, 2013. https://doi.org/10.1109/ccgrid.2013.89
13. Z. Á. Mann. "Allocation of Virtual Machines in Cloud Data Centers—A Survey of Problem Models and Optimization Algorithms", *ACM Computing Surveys*, 48 (1), pp. 1–34, 2015. https://doi.org/10.1145/2797211
14. Z. Ivkovic and E. L. Lloyd. "Fully Dynamic Algorithms for Bin Packing: Being (Mostly) Myopic Helps", *SIAM Journal on Computing*, 28 (2), pp. 574–611, 1998. https://doi.org/10.1137/s0097539794276749
15. L. Epstein and A. Levin. "A robust APTAS for the classical bin packing problem", *Mathematical Programming*, 119 (1), pp. 33–49, 2009. https://doi.org/10.1007/s10107-007-0200-y
16. J. Balogh, J. Békési, G. Galambos and G. Reinelt. "Lower Bound for the Online Bin Packing Problem with Restricted Repacking", *SIAM Journal on Computing*, 38 (1), pp. 398–410, 2008. https://doi.org/10.1137/050647049
17. K. Jansen and K.-M. Klein. "A Robust AFPTAS for Online Bin Packing with Polynomial Migration", *SIAM Journal on Discrete Mathematics*, 33 (4), pp. 2062–2091, 2019. https://doi.org/10.1137/17m1122529
18. S. Berndt, K. Jansen and K.-M. Klein. "Fully dynamic bin packing revisited", *Mathematical Programming*, 179 (1–2), pp. 109–155, 2020. https://doi.org/10.1007/s10107-018-1325-x
19. B. Feldkord, M. Feldotto, A. Gupta, G. Guruganesh, A. Kumar, S. Riechers and D. Wajc. "Fully-Dynamic Bin Packing with Little Repacking". In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)* (LIPIcs 107), pp. 51:1–51:24, 2018. https://doi.org/10.4230/lipics.icalp.2018.51
20. 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 *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 153–167, 2017. https://doi.org/10.1145/3132747.3132772
21. R. van Bevern, A. Melnikov, P. Smirnov and O. Tsidulko. "On data reduction for dynamic vector bin packing". arXiv preprint arXiv:2205.08769v1, 2022. Available at: https://arxiv.org/abs/2205.08769v1
22. T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. Wallach, H. Daumé III and K. Crawford. "Datasheets for datasets", *Communications of the ACM*, 64 (12), pp. 86–92, 2021. https://doi.org/10.1145/3458723
23. F. Brandão and J. P. Pedroso. "Bin packing and related problems: General arc-flow formulation with graph compression", *Computers & Operations Research*, 69, pp. 56–67, 2016. https://doi.org/10.1016/j.cor.2015.11.009
24. M. X. Goemans and T. Rothvoss. "Polynomiality for Bin Packing with a Constant Number of Item Types", *Journal of the ACM*, 67 (6), pp. 1–21, 2020. https://doi.org/10.1145/3421750
25. A. Ray. "There is no APTAS for 2-dimensional vector bin packing: Revisited". arXiv preprint arXiv:2104.13362v1, 2021. Available at: https://arxiv.org/abs/2104.13362v1
26. P. C. Gilmore and R. E. Gomory. "A Linear Programming Approach to the Cutting-Stock Problem", *Operations Research*, 9 (6), pp. 849–859, 1961. https://doi.org/10.1287/opre.9.6.849
27. F. Vanderbeck. "Computational study of a column generation algorithm for bin packing and cutting stock problems", *Mathematical Programming*, 86 (3), pp. 565–594, 1999. https://doi.org/10.1007/s101070050105
28. A. A. Farley. "A Note on Bounding a Class of Linear Programming Problems, Including Cutting Stock Problems", *Operations Research*, 38 (5), pp. 922–923, 1990. https://doi.org/10.1287/opre.38.5.922
29. P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python", *Nature Methods*, 17 (3), pp. 261–272, 2020. https://doi.org/10.1038/s41592-019-0686-2
30. D. N. Politis and J. P. Romano. "The Stationary Bootstrap", *Journal of the American Statistical Association*, 89 (428), pp. 1303–1313, 1994. https://doi.org/10.1080/01621459.1994.10476870
31. D. N. Politis and H. White. "Automatic Block-Length Selection for the Dependent Bootstrap", *Econometric Reviews*, 23 (1), pp. 53–70, 2004. https://doi.org/10.1081/etc-120028836
32. A. Patton, D. N. Politis and H. White. "Correction to "Automatic Block-Length Selection for the Dependent Bootstrap" by D. Politis and H. White", *Econometric Reviews*, 28 (4), pp. 372–375, 2009. https://doi.org/10.1080/07474930802459016
33. D. J. Schuirmann. "A comparison of the Two One-Sided Tests Procedure and the Power Approach for assessing the equivalence of average bioavailability", *Journal of Pharmacokinetics and Biopharmaceutics*, 15 (6), pp. 657–680, 1987. https://doi.org/10.1007/bf01068419
34. F. Bretz, W. Maurer, W. Brannath and M. Posch. "A graphical approach to sequentially rejective multiple test procedures", *Statistics in Medicine*, 28 (4), pp. 586–604, 2009. https://doi.org/10.1002/sim.3495
35. Y. Benjamini and Y. Hochberg. "Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing", *Journal of the Royal Statistical Society: Series B (Methodological)*, 57 (1), pp. 289–300, 1995. https://doi.org/10.1111/j.2517-6161.1995.tb02031.x
36. S. Steegen, F. Tuerlinckx, A. Gelman and W. Vanpaemel. "Increasing Transparency Through a Multiverse Analysis", *Perspectives on Psychological Science*, 11 (5), pp. 702–712, 2016. https://doi.org/10.1177/1745691616658637
37. U. Simonsohn, J. P. Simmons and L. D. Nelson. "Specification curve analysis", *Nature Human Behaviour*, 4 (11), pp. 1208–1214, 2020. https://doi.org/10.1038/s41562-020-0912-z
38. T. Hoefler and R. Belli. "Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results". In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, 2015. https://doi.org/10.1145/2807591.2807644

39. C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau et al. "Array programming with NumPy", *Nature*, 585 (7825), pp. 357–362, 2020. https://doi.org/10.1038/s41586-020-2649-2
40. S. K. Lam, A. Pitrou and S. Seibert. "Numba: a LLVM-based Python JIT compiler". In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, pp. 1–6, 2015. https://doi.org/10.1145/2833157.2833162
41. D. R. MacIver, Z. Hatfield-Dodds and many other contributors. "Hypothesis: A new approach to property-based testing", *Journal of Open Source Software*, 4 (43), article 1891, 2019. https://doi.org/10.21105/joss.01891
42. Y. Azar and D. Vainstein. "Tight Bounds for Clairvoyant Dynamic Bin Packing", *ACM Transactions on Parallel Computing*, 6 (3), pp. 1–21, 2019. https://doi.org/10.1145/3364214
43. M. Liu and X. Tang. "Dynamic Bin Packing with Predictions", *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 6 (3), pp. 1–24, 2022. https://doi.org/10.1145/3570605