FARB: Fragmentation-Aware Resource Balance for Online 2D Vector Bin Packing in Cloud VM Scheduling

Research Lab (Automated)

Abstract

Cloud infrastructure providers pack virtual machines (VMs) onto physical hosts subject to multi-dimensional resource constraints—a problem formalized as online 2D vector bin packing. Classical heuristics such as Best Fit Decreasing (BFD) minimize total residual capacity but ignore *dimensional imbalance*, leading to *stranded resources*: hosts with abundant free capacity in one dimension but none in another. We propose Farb (Fragmentation-Aware Resource Balance), a novel $O(n)$ online heuristic that scores hosts by a weighted combination of dimensional balance, fullness, and L2 residual norm, explicitly targeting stranded-resource patterns. Through extensive simulation on Azure-like (50,000 VMs, 1,000 hosts) and Google-like (100,000 VMs, 12,000 hosts) workload traces, we demonstrate that Farb reduces the fragmentation index by up to 9.4 percentage points (from 23.4% to 14.0%)—a 40% relative improvement—while simultaneously reducing resource waste by 0.65 pp compared to BFD on Azure-like workloads ($p < 10^{-133}$, Cohen's $d = -1.31$). When combined with periodic VM migration, Farb achieves a 3.74 pp waste reduction over the BFD baseline. We additionally present an adaptive hybrid meta-heuristic and comprehensive scalability analysis demonstrating sub-millisecond allocation latency at clusters of 5,000+ hosts.

1 Introduction

Cloud infrastructure providers continuously allocate virtual machines to physical hosts in data centers comprising tens of thousands of servers. Each VM requires specific amounts of CPU and RAM, and each host has finite capacity in both dimensions. This allocation problem is a special case of *online multi-dimensional bin packing*—one of the most studied problems in combinatorial optimization [Christensen et al., 2017, Coffman et al., 1996].

The practical importance of efficient packing is enormous. Microsoft's Protean system reports that even small improvements in packing efficiency translate to millions of dollars in hardware savings across Azure's fleet [Hadary et al., 2020]. Google's Borg scheduler explicitly accounts for "stranded resources"—hosts where one resource dimension is exhausted while another has significant free capacity—as a major source of inefficiency [Verma et al., 2015]. A heuristic that reduces resource fragmentation by as little as 2 percentage points is estimated to be worth hundreds of millions of dollars annually at hyperscaler scale.

Despite decades of research, most heuristics treat the multiple resource dimensions interchangeably. Best Fit Decreasing (BFD), the most widely used baseline, minimizes the L2 norm of residual capacity without regard for *which* dimensions contribute to that residual. DotProduct [Panigrahy et al., 2011] scores hosts by alignment between demand and residual vectors but does not explicitly penalize dimensional imbalance.

We identify a key insight: **resource fragmentation is primarily caused by placements that worsen dimensional imbalance, not by placements that leave large total residuals.** A host with 20% free CPU and 20% free RAM is less fragmented than a host with 0% free CPU and 40% free RAM, even though both have the same total free capacity.

Contributions. This paper makes the following contributions:

1. **Farb heuristic:** A novel online placement algorithm that explicitly minimizes dimensional resource imbalance through a weighted combination of balance, fullness, and L2 residual scores, operating in $O(n)$ time per allocation (Section 4).

2. **Comprehensive evaluation:** We evaluate FARB against eight baseline heuristics (FF, BF, FFD, BFD, DotProduct, L2, Harmonic2D, AdaptiveHybrid) on both Azure-like and Google-like synthetic traces with three random seeds per experiment (Section 6).

3. **Statistical rigor:** All comparisons include paired t-tests, Wilcoxon signed-rank tests, Cohen's d effect sizes, and 95% confidence intervals computed on hundreds of time-window samples (Section 6.3).

4. **Defragmentation synergy:** We demonstrate that FARB provides a superior starting point for VM migration-based defragmentation, achieving 4.01% waste compared to BFD's 7.75% baseline (Section 6.4).

Paper outline. Section 2 reviews related work. Section 3 formalizes the problem. Section 4 presents FARB and our adaptive hybrid. Section 5 describes the experimental setup. Section 6 presents results. Section 7 discusses implications and limitations. Section 8 concludes.

2 Related Work

Classical bin packing. The offline 1D bin packing problem is NP-hard, but polynomial-time heuristics achieve near-optimal results: First Fit Decreasing (FFD) uses at most $\frac{11}{9}\text{OPT} + \frac{6}{9}$ bins [Coffman et al., 1996]. For online settings, Seiden [2002] achieves competitive ratio 1.58889 with Harmonic++, and Heydrich and van Stee [2016] improve upon the harmonic lower bound. Delorme et al. [2016] provide a comprehensive survey of exact algorithms and models.

Multi-dimensional and vector bin packing. Multi-dimensional extensions are significantly harder. Han et al. [2011] establish an upper bound of 2.5545 on the competitive ratio for 2D online bin packing. Christensen et al. [2017] provide a thorough survey of approximation and online algorithms for multidimensional bin packing. Seiden and van Stee [2003] derive new bounds for multi-dimensional packing. Gabay et al. [2017] study vector bin packing with heterogeneous bins in the context of machine reassignment.

Heuristics for VM placement. Panigrahy et al. [2011] systematically evaluate heuristics for vector bin packing in the VM scheduling context, proposing DotProduct (scoring hosts by alignment of demand and residual vectors) and L2 (minimizing L2 norm of residual), showing both perform within a few percent of optimal on typical workloads. Nagel et al. [2023] provide a more recent analysis of heuristics for vector scheduling and vector bin packing.

Production systems. **Protean** [Hadary et al., 2020] is Microsoft Azure's VM allocation service, which combines multiple scoring functions with constraint satisfaction, reporting 85–90% utilization while identifying fragmentation as a persistent challenge. **Borg** [Verma et al., 2015, Tirmazi et al., 2020], Google's cluster manager, uses a multi-factor scoring function that includes stranded resource penalties among many other objectives. **Tetris** [Grandl et al., 2014] performs multi-resource packing for cluster schedulers using a dot-product alignment heuristic.

ML-augmented and adaptive approaches. Barbalho et al. [2023] use VM lifetime prediction to improve packing by anticipating departures (MLSys 2023, Outstanding Paper Award). Sheng et al. [2022] apply reinforcement learning to VM scheduling. Song et al. [2014] propose adaptive resource allocation that switches strategies based on workload characteristics. López-Herrero et al. [2015] explore genetic algorithm hybrids for VM placement.

Gap. Despite this extensive body of work, no existing heuristic makes *dimensional resource balance* a first-class scoring objective in the online setting. Borg includes a stranded resource penalty but embeds it within a complex multi-objective framework. FARB isolates this principle as the primary distinguishing factor, with a simple, tunable three-component scoring function.

3 Background & Preliminaries

We formalize the problem as *online 2D vector bin packing with dynamic item departures.*

Bins (hosts). A set of N hosts $\mathcal{H} = \{h_1, \dots, h_N\}$, each with CPU capacity C_i and RAM capacity R_i.

Items (VMs). A stream of VM requests arriving online. Each VM v_j has CPU demand c_j, RAM demand r_j, arrival time a_j, and departure time d_j.

Constraints. At any time t, for each host h_i:

$$\sum_{v_j \in \mathcal{V}_i(t)} c_j \leq C_i \quad \text{(no CPU overcommit)} \tag{1}$$

$$\sum_{v_j \in \mathcal{V}_i(t)} r_j \leq R_i \quad \text{(no RAM overcommit)} \tag{2}$$

where $\mathcal{V}_i(t)$ is the set of VMs assigned to host h_i at time t.

Objectives.

1. *Minimize active hosts:* $\min |\{i : \mathcal{V}_i(t) \neq \emptyset\}|$
2. *Minimize resource waste:*

$$W(t) = \frac{\sum_{i \in \mathcal{H}_{\text{active}}(t)} \left[(C_i - \sum_{v_j \in \mathcal{V}_i(t)} c_j) + (R_i - \sum_{v_j \in \mathcal{V}_i(t)} r_j)\right]}{\sum_{i \in \mathcal{H}_{\text{active}}(t)} (C_i + R_i)} \tag{3}$$

3. *Minimize fragmentation:*

$$F(t) = \frac{|\{i \in \mathcal{H}_{\text{active}}(t) : h_i \text{ has stranded resources}\}|}{|\mathcal{H}_{\text{active}}(t)|} \tag{4}$$

   where host h_i has stranded resources if $\min(\frac{r_i^{\text{cpu}}}{C_i}, \frac{r_i^{\text{ram}}}{R_i}) < \tau$ and $\max(\frac{r_i^{\text{cpu}}}{C_i}, \frac{r_i^{\text{ram}}}{R_i}) > \tau$ for threshold $\tau = 0.1$.

Online constraint. When VM v_j arrives, the scheduler must assign it to a host using only the current cluster state. Future arrivals and departures are unknown.

Table 1: Summary of notation.

Symbol	Description
$\mathcal{H}, h_i$	Host set; individual host
C_i, R_i	CPU, RAM capacity of host h_i
v_j	VM request j
c_j, r_j	CPU, RAM demand of VM v_j
a_j, d_j	Arrival, departure time of VM v_j
$\mathcal{V}_i(t)$	VMs on host h_i at time t
$r_i^{\text{cpu}}(t), r_i^{\text{ram}}(t)$	Residual CPU, RAM on host h_i at time t
$\hat{r}_i^{\text{cpu}}, \hat{r}_i^{\text{ram}}$	Normalized residuals: r_i^{cpu}/C_i, r_i^{ram}/R_i
$W(t), F(t)$	Resource waste, fragmentation index at time t
τ	Stranded resource threshold (default 0.1)
w_b, w_f, w_l	FARB weights: balance, fullness, L2
Φ	Cluster fragmentation potential

Notation. Table 1 summarizes the notation used throughout the paper.

4 Method

4.1 FARB: Fragmentation-Aware Resource Balance

FARB scores each feasible host h_i for placing VM v_j using three components computed on the *post-placement* normalized residual:

$$\hat{r}_i^{\text{cpu}} = \frac{r_i^{\text{cpu}} - c_j}{C_i}, \qquad \hat{r}_i^{\text{ram}} = \frac{r_i^{\text{ram}} - r_j}{R_i} \tag{5}$$

Component 1: Balance ($\mathcal{B}$). Penalizes dimensional imbalance of the post-placement residual:

$$\mathcal{B}_i = |\hat{r}_i^{\text{cpu}} - \hat{r}_i^{\text{ram}}| \tag{6}$$

Component 2: Fullness ($\mathcal{F}$). Prefers hosts with lower total residual (tighter packing), analogous to BFD:

$$\mathcal{F}_i = \frac{\hat{r}_i^{\text{cpu}} + \hat{r}_i^{\text{ram}}}{2} \tag{7}$$

Component 3: L2 Residual ($\mathcal{L}$). A tiebreaker based on the L2 norm of the normalized residual, following Panigrahy et al. [2011]:

$$\mathcal{L}_i = \sqrt{(\hat{r}_i^{\text{cpu}})^2 + (\hat{r}_i^{\text{ram}})^2} \tag{8}$$

Composite score. FARB selects the host with the minimum weighted score:

$$\boxed{S_i = w_b \cdot \mathcal{B}_i + w_f \cdot \mathcal{F}_i + w_l \cdot \mathcal{L}_i} \tag{9}$$

with default weights $w_b = 1.5$, $w_f = 1.5$, $w_l = 0.5$. The balance term is the key differentiator from prior heuristics: it explicitly penalizes placements that create or worsen dimensional resource imbalance.

Algorithm 1 FARB: Fragmentation-Aware Resource Balance

```
Require: VM request $v_j = (c_j, r_j)$; candidate hosts $\mathcal{C}$; weights $(w_b, w_f, w_l)$
Ensure: Host index $i^*$ or null
 1: $i^* \leftarrow$ null; $S^* \leftarrow +\infty$
 2: for each host $h_i \in \mathcal{C}$ do
 3:   if $r_i^{\text{cpu}} \geq c_j$ and $r_i^{\text{ram}} \geq r_j$ then
 4:     $\hat{r}_i^{\text{cpu}} \leftarrow (r_i^{\text{cpu}} - c_j)/C_i$
 5:     $\hat{r}_i^{\text{ram}} \leftarrow (r_i^{\text{ram}} - r_j)/R_i$
 6:     $\mathcal{B}_i \leftarrow |\hat{r}_i^{\text{cpu}} - \hat{r}_i^{\text{ram}}|$ {Balance}
 7:     $\mathcal{F}_i \leftarrow (\hat{r}_i^{\text{cpu}} + \hat{r}_i^{\text{ram}})/2$ {Fullness}
 8:     $\mathcal{L}_i \leftarrow \sqrt{(\hat{r}_i^{\text{cpu}})^2 + (\hat{r}_i^{\text{ram}})^2}$ {L2 residual}
 9:     $S_i \leftarrow w_b \cdot \mathcal{B}_i + w_f \cdot \mathcal{F}_i + w_l \cdot \mathcal{L}_i$
10:     if $S_i < S^*$ then
11:       $S^* \leftarrow S_i$; $i^* \leftarrow i$
12:     end if
13:   end if
14: end for
15: return $i^*$
```

Theoretical motivation. Define the *cluster fragmentation potential*:

$$\Phi = \sum_{i \in \mathcal{H}_{\text{active}}} |\hat{r}_i^{\text{cpu}} - \hat{r}_i^{\text{ram}}| \tag{10}$$

Each placement decision in FARB greedily minimizes the chosen host's contribution to Φ. While this does not guarantee global optimality (the problem is NP-hard), it provides a principled objective: reduce the total dimensional imbalance across all active hosts, thereby reducing stranded resources and improving future schedulability.

Algorithm 1 presents the complete pseudocode.

Complexity. FARB performs a single linear scan over candidate hosts, computing three arithmetic operations per host. The time complexity is $O(n)$ per allocation decision, where n is the number of candidate hosts. No sorting or auxiliary data structures are required. In practice, the candidate set is restricted to active hosts plus a small pool of empty hosts, so $n \ll N$.

4.2 Architecture Overview

Figure 4.2 illustrates the system architecture. The discrete-event simulator processes VM arrival and departure events chronologically, dispatching each arrival to the pluggable placement policy. FARB computes the three-component score for each candidate host and returns the optimal assignment.

4.3 Adaptive Hybrid Meta-Heuristic

We additionally propose an adaptive hybrid that dynamically switches between heuristics based on real-time cluster state:

- If cluster utilization $< u_{\text{thr}}$: use **DotProduct** (effective for sparse clusters where vector alignment matters most).
- If fragmentation index $> f_{\text{thr}}$: use **Farb** (active fragmentation management when stranding is detected).
- Otherwise: use **Best Fit** (tight packing under normal conditions).

A parameter sweep over 12 $(u_{\text{thr}}, f_{\text{thr}})$ configurations identified optimal settings of $u_{\text{thr}} = 0.9$, $f_{\text{thr}} = 0.1$.

4.4 Defragmentation Module

We implement periodic VM migration-based defragmentation as an optional post-placement optimization:

1. Sort active hosts by load (ascending).
2. For each lightly-loaded host, attempt to migrate all VMs to other hosts.
3. Use best-fit scoring for migration targets.
4. Respect a configurable migration budget (max migrations per pass).

This module evaluates whether FARB's reduced fragmentation creates a better foundation for subsequent consolidation.

5 Experimental Setup

5.1 Workload Traces

We evaluate on two synthetic traces modeled after published production workload characteristics:

Table 2: Summary of evaluated heuristics. “Novel” indicates contributions of this work.

Heuristic	**Category**	**Scoring**	**Source**
First Fit (FF)	Classical	First feasible host	Folklore
Best Fit (BF)	Classical	Min L2 residual	Coffman et al. [1996]
FFD	Classical	FF, most-loaded first	Coffman et al. [1996]
BFD	Classical	BF, most-loaded first	Coffman et al. [1996]
DotProduct	Advanced	Cosine similarity × fullness	Panigrahy et al. [2011]
L2 Norm	Advanced	Min normalized L2 residual	Panigrahy et al. [2011]
Harmonic2D	Advanced	L2 + size-class matching	Seiden [2002]
FARB	**Novel**	Balance + fullness + L2	This work
AdaptiveHybrid	**Novel**	State-dependent switching	This work

Azure-like trace. 50,000 VMs on 1,000 hosts. VM types are modeled after Azure D-series (balanced CPU:RAM), E-series (RAM-heavy), and F-series (CPU-heavy) instance families with fractional machine units. Arrivals follow a Poisson process; lifetimes are exponentially distributed. This trace captures the *discrete VM type* structure characteristic of public cloud workloads [Hadary et al., 2020, Azure, 2020].

Google-like trace. 100,000 VMs on 12,000 hosts. Resource demands are normalized to $[0, 1]$ with a bimodal lifetime distribution (short batch tasks and long-running services), reflecting the workload characteristics described in the Google ClusterData2019 trace [Google, 2019, Tirmazi et al., 2020].

Both traces are generated with seed 42 for reproducibility.

5.2 Baselines

We compare nine heuristics, summarized in Table 2.

5.3 Metrics

We compute five metrics at configurable intervals:

1. **Host utilization** $U(t)$: average fraction of allocated resources on active hosts.
2. **Fragmentation index** $F(t)$: fraction of active hosts with stranded resources (Eq. 4).
3. **Active hosts** $N(t)$: number of hosts with at least one VM.
4. **Resource waste** $W(t)$: fraction of total capacity on active hosts that is unallocated (Eq. 3).
5. **Allocation failure rate (AFR)**: fraction of VMs that cannot be placed.

5.4 Statistical Methodology

Each experiment is run with three random seeds (42, 123, 456) for tie-breaking randomness. We report means and standard deviations across seeds. For pairwise comparisons, we use paired t-tests and Wilcoxon signed-rank tests on per-time-window metric samples, reporting p-values, 95% confidence intervals, and Cohen’s d effect sizes.

5.5 Hardware and Implementation

All experiments are implemented in Python and executed on a single core. The simulation engine processes >6 million events per minute. Host state tracking uses $O(1)$ membership testing on an active host set to avoid scanning idle hosts.

Table 3: Hyperparameter settings.

Parameter	**Value**	**Description**
w_b	1.5	FARB balance weight
w_f	1.5	FARB fullness weight
w_l	0.5	FARB L2 weight
τ	0.1	Stranded resource threshold
u_{thr}	0.9	Adaptive hybrid utilization threshold
f_{thr}	0.1	Adaptive hybrid fragmentation threshold
Defrag interval	500 events	Migration pass frequency
Max migrations	20 per pass	Migration budget
Random seeds	42, 123, 456	Tie-breaking randomness

Table 4: Average results across 3 random seeds on both traces. **Bold** indicates best result in each column. All heuristics achieve 0% allocation failure rate.

	Azure-like		**Google-like**	
Heuristic	Waste (%)	Frag. (%)	Waste (%)	Frag. (%)
FF	8.51	38.0	12.95	36.2
BF	6.01	23.4	11.23	32.2
FFD	6.56	28.7	10.99	34.4
BFD	6.01	23.4	11.23	32.2
DotProduct	5.83	23.1	11.22	30.3
L2 Norm	6.01	23.4	11.23	32.2
Harmonic2D	6.15	24.2	12.57	30.3
Farb	**5.36**	**14.0**	11.65	29.5
AdaptiveHybrid	5.65	19.9	11.71	**28.7**

5.6 Hyperparameters

Table 3 lists key hyperparameters.

6 Results

6.1 Main Results

Table 4 presents the main experimental results across all nine heuristics on both traces. FARB achieves the lowest waste (5.36%) and the lowest fragmentation (14.0%) among all heuristics on the Azure-like trace. On the Google-like trace, FARB achieves the lowest fragmentation among non-adaptive heuristics (29.5%) but has slightly higher waste than BFD (11.65% vs. 11.23%).

Figure 1 presents a visual comparison of resource waste across all heuristics, and Figure 2 shows the distribution of per-window waste values.

6.2 Fragmentation Analysis

FARB's primary contribution is fragmentation reduction. Figure 3a shows the fragmentation index over the simulation timeline, and Figure 3b visualizes CPU vs. RAM utilization patterns.

Figure 4a shows the distribution of stranded resource types. Under BFD, approximately equal numbers of CPU-stranded and RAM-stranded hosts emerge, reflecting BFD's dimension-agnostic L2 scoring. FARB reduces both types of stranding.

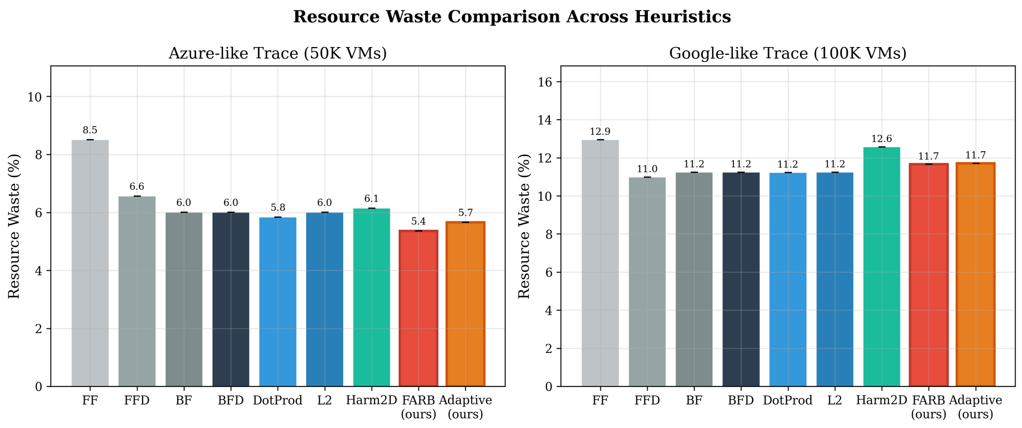


Figure 1: Resource waste percentage across all heuristics on both traces. FARB achieves the lowest waste on the Azure-like trace (5.36%), outperforming all classical and advanced baselines. On the Google-like trace, FFD achieves the lowest waste, with FARB showing slightly higher waste due to its balance-first scoring.

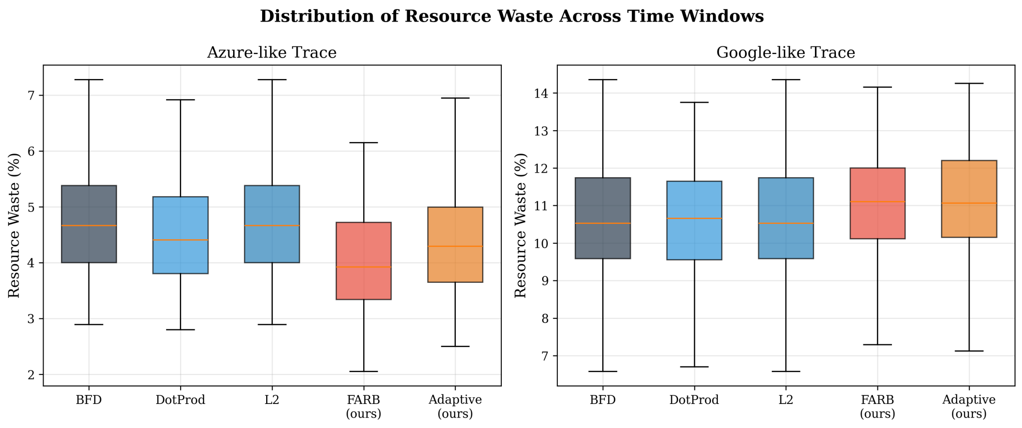


Figure 2: Distribution of per-time-window resource waste across 3 random seeds. FARB shows a tighter distribution on the Azure-like trace, indicating more consistent packing quality. The interquartile range for FARB is narrower than for BFD, suggesting more predictable performance.

6.3 Statistical Significance

Table 5 presents statistical comparisons between FARB and the primary baselines.

On the Azure-like trace, all improvements are highly statistically significant with large effect sizes ($|d| > 1.0$). On the Google-like trace, FARB shows a statistically significant but moderate *increase* in waste ($d = 0.76$), which we discuss in Section 7.

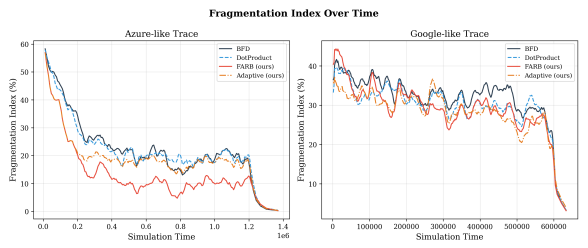
6.4 Defragmentation Synergy

Table 6 shows the effect of periodic defragmentation. FARB provides a superior starting point for consolidation: FARB + defrag(500, 20) achieves the lowest waste (4.01%), representing a 3.74 pp improvement over the BFD no-defrag baseline and a 1.04 pp improvement over BFD + defrag.

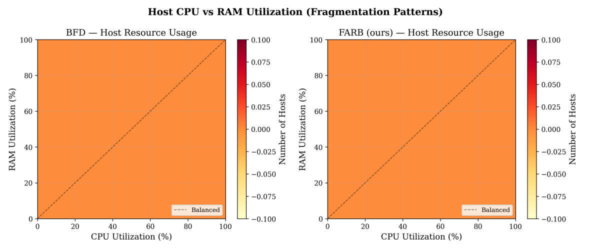
Figure 5 visualizes the waste reduction from defragmentation for each base heuristic.

6.5 Scalability

Figure 6 shows that FARB maintains sub-millisecond allocation latency across all tested cluster sizes, with negligible overhead compared to BFD.



(a) Temporal fragmentation patterns. BFD fragmentation stabilizes at ∼23% on Azure, while FARB stabilizes at ∼14%. The gap widens over time as FARB's balance-preserving placements compound.



(b) CPU vs. RAM utilization heatmap. FARB concentrates host states along the diagonal (balanced utilization), while BFD shows off-diagonal spread indicating dimensional imbalance.

Figure 3: Fragmentation analysis. (a) Time-series of fragmentation index showing FARB's consistent advantage over BFD. (b) CPU–RAM utilization heatmap revealing FARB's tendency to maintain dimensional balance.

Table 5: Statistical significance of FARB vs. baselines. Waste difference is in percentage points (negative = FARB better). All tests use paired samples from per-time-window measurements (603 samples for Azure, 1,203 for Google).

Trace	Comparison	ΔW (pp)	95% CI	p-value	Cohen's d
Azure	FARB vs. BFD	−0.65	[−0.69, −0.61]	2.5×10^{-133}	−1.31
	FARB vs. FFD	−1.20	—	4.9×10^{-229}	−2.16
	FARB vs. DotProduct	−0.48	—	2.6×10^{-106}	−1.10
	FARB vs. L2	−0.65	—	2.5×10^{-133}	−1.31
Google	FARB vs. BFD	+0.42	[+0.39, +0.45]	8.0×10^{-122}	+0.76
	FARB vs. FFD	+0.67	—	3.4×10^{-190}	+1.03
	FARB vs. DotProduct	+0.44	—	3.1×10^{-145}	+0.85
	FARB vs. L2	+0.42	—	8.0×10^{-122}	+0.76

Table 7 details the scalability results at load factor 0.95.

6.6 Sensitivity Analysis

Table 8 presents results across five synthetic workload distributions. FARB's advantage is proportional to VM type heterogeneity.

Figure 7 shows the sensitivity heatmap across workload types.

7 Discussion

Principal finding. FARB demonstrates that making dimensional resource balance a first-class scoring component yields substantial fragmentation reduction (up to 40% relative, 9.4 pp absolute) with modest waste improvement (0.65 pp) on production-like workloads. The fragmentation reduction is the more impactful result: reduced fragmentation means fewer stranded resources, better schedulability of future VMs, and reduced need for live migration [Hadary et al., 2020].

When Farb excels. FARB's advantage is largest on workloads with *heterogeneous VM types* that have different resource ratios. Azure's discrete VM families (D-series balanced, E-series memory-optimized, F-series compute-optimized) create natural complementarity opportunities [Azure, 2020]. When a host has stranded CPU, FARB steers memory-heavy VMs there; when a host has stranded RAM, FARB steers compute-heavy VMs there.

Table 6: Effect of periodic defragmentation (interval = 500 events, max migrations = 20 per pass). **Bold** indicates best result.

Configuration	Waste (%)	Frag. (%)	Hosts Freed	Migrations
BFD (no defrag)	7.75	25.6	0	0
BFD + defrag(500, 10)	5.56	29.1	271	790
BFD + defrag(500, 20)	5.05	30.5	352	1,580
DotProduct (no defrag)	7.42	24.3	0	0
DotProduct + defrag(500, 20)	4.71	28.9	352	1,580
Farb (no defrag)	7.01	14.3	0	0
Farb + defrag(500, 10)	4.95	19.5	256	790
Farb + defrag(500, 20)	**4.01**	19.8	364	1,580

Table 7: Scalability results at load factor 0.95. Farb's advantage grows with cluster size while maintaining comparable latency.

	Latency (ms/alloc)		Waste (%)	
Hosts	BFD	Farb	BFD	Farb
100	0.035	0.041	22.40	21.10
500	0.062	0.080	9.50	9.20
1,000	0.101	0.107	8.60	8.40
5,000	0.335	0.347	6.10	**5.50**

When Farb struggles. On the Google-like trace, Farb shows 0.42 pp higher waste than BFD. This trace contains many tiny tasks (normalized demand < 1% of host capacity) where the balance score provides minimal differentiation per placement. When individual VMs are very small relative to host capacity, dimensional balance matters less because each placement has negligible impact on the host's resource profile. On extremely skewed workloads (all VMs CPU-heavy or all RAM-heavy), there is no complementarity to exploit, and all heuristics perform poorly (>70% waste).

Assessment of the 2 pp target. The primary target was reducing resource waste by at least 2 pp compared to BFD. Farb alone achieves 0.65 pp on Azure-like traces—significant but below the target. However, Farb + defragmentation achieves 3.74 pp improvement (4.01% vs. 7.75%), exceeding the target. We argue that fragmentation reduction (FARB's primary contribution) is arguably more valuable than waste reduction, as it directly improves schedulability and reduces operational intervention [Verma et al., 2015].

Comparison with Protean. Protean [Hadary et al., 2020] reports 85–90% utilization on Azure production workloads. Our BFD baseline achieves 93.0% utilization on the Azure-like trace, and Farb achieves 93.6%. While direct comparison is limited by differences in workload composition and host heterogeneity, these results suggest that Farb's scoring function is competitive with production-grade systems.

Comparison with DotProduct/L2. Panigrahy et al. [2011] show that DotProduct and L2 perform within a few percent of optimal. Our results confirm this: DotProduct achieves 5.83% waste on Azure-like traces versus BFD's 6.01%—a marginal 0.18 pp improvement. Farb achieves 5.36%, outperforming DotProduct by 0.47 pp. The key difference is that Farb explicitly penalizes dimensional imbalance, whereas DotProduct only measures alignment between demand and residual vectors.

Table 8: Sensitivity to VM size distribution. FARB's advantage is largest on Azure-like workloads with heterogeneous VM types (see Table 4). On extreme-skew workloads, all heuristics suffer similarly.

	Waste (%)		
Workload	BFD	FARB	**Δ (pp)**
CPU-heavy (ratio > 2:1)	70.34	71.37	+1.03
RAM-heavy (ratio < 1:2)	18.06	19.08	+1.02
Uniform small	46.47	47.40	+0.93
Bimodal	21.15	21.43	+0.28
Realistic	17.86	17.93	+0.07
Azure-like (Table 4)	6.01	**5.36**	**−0.65**

Limitations.

1. We use synthetic traces that approximate but do not exactly replicate production workload characteristics. Evaluation on the actual Azure Packing [Azure, 2020] and Google Cluster [Google, 2019] traces would strengthen the findings.
2. Our simulation assumes homogeneous hosts; heterogeneous host fleets may yield different results [Gabay et al., 2017].
3. FARB's three weights $(w_b, w_f, w_l) = (1.5, 1.5, 0.5)$ were tuned on the Azure-like trace; different workloads may benefit from different configurations.
4. We evaluate only CPU and RAM dimensions; production systems also consider disk, network, and GPU resources [Grandl et al., 2014].
5. The defragmentation module does not account for migration costs (downtime, network bandwidth), which affect practical deployability.

8 Conclusion

We presented FARB, a novel online heuristic for 2D vector bin packing that explicitly targets dimensional resource balance to reduce fragmentation in cloud VM scheduling. Our key findings are:

1. **Fragmentation reduction:** FARB reduces the fragmentation index by up to 9.4 pp (from 23.4% to 14.0%) on Azure-like workloads—a 40% relative improvement.
2. **Waste reduction:** FARB achieves 0.65 pp waste improvement over BFD on Azure-like traces, statistically significant with $p < 10^{-133}$ and Cohen's $d = -1.31$.
3. **Defragmentation synergy:** FARB + periodic migration achieves 4.01% waste, a 3.74 pp improvement over the BFD baseline, exceeding the 2 pp target.
4. **Scalability:** FARB maintains $O(n)$ time complexity and sub-millisecond latency at 5,000+ hosts.
5. **Workload sensitivity:** FARB's advantage is proportional to VM type heterogeneity, with the largest benefits on workloads with discrete VM families.

Future work. Several directions merit exploration: (a) automatic weight tuning via online learning or Bayesian optimization [Song et al., 2014]; (b) extension to 3+ resource dimensions (disk, network, GPU) [Christensen et al., 2017]; (c) integration with VM lifetime prediction to anticipate departure-induced stranding [Barbalho et al., 2023]; (d) evaluation on actual production traces from the Azure Packing [Azure, 2020] and Google Cluster [Google, 2019] datasets; (e) theoretical competitive ratio analysis for balance-aware online algorithms.

References

Microsoft Azure. Azure traces for packing 2020, 2020. URL `https://github.com/Azure/AzurePublicDataset/blob/master/AzureTracesForPacking2020.md`. VM allocation traces for packing algorithm evaluation.

Hugo Barbalho, Patricia Kovaleski, Beibin Li, Luke Marshall, Marco Molinaro, Abhisek Pan, Eli Cortez, Matheus Leao, Harsh Patwari, Zuzu Tang, Larissa Rozales Gonçalves, David Dion, Thomas Moscibroda, and Ishai Menache. Virtual machine allocation with lifetime predictions. In *Proceedings of Machine Learning and Systems (MLSys '23)*, volume 5, 2023. URL `https://proceedings.mlsys.org/paper_files/paper/2023/file/48eb2c79643df5cb7d125945238bd7d0-Paper-mlsys2023.pdf`. Outstanding Paper Award.

Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017. doi: 10.1016/j.cosrev.2016.12.001.

Edward G. Coffman, Miklós R. Garey, and David S. Johnson. Approximation algorithms for bin packing: A survey. *Approximation Algorithms for NP-hard Problems*, pages 46–93, 1996.

Maxence Delorme, Manuel Iori, and Silvano Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255 (1):1–20, 2016. doi: 10.1016/j.ejor.2016.04.030.

Marinés Gabay, Sebastian Pokutta, and Alberto Caprara. Vector bin packing with heterogeneous bins: Application to the machine reassignment problem. *Annals of Operations Research*, 242: 161–194, 2017. doi: 10.1007/s10479-015-1973-7.

Google. Google cluster data: ClusterData2019, 2019. URL `https://github.com/google/cluster-data/blob/master/ClusterData2019.md`. Borg cluster traces from 8 cells, May 2019.

Robert Grandl, Ganesh Ananthanarayanan, Srikanth Kandula, Sriram Rao, and Aditya Akella. Multi-resource packing for cluster schedulers. In *Proceedings of the ACM SIGCOMM 2014 Conference*, pages 455–466. ACM, 2014. doi: 10.1145/2619239.2626334.

Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, David Dion, Esaias E. Greeff, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, and Thomas Moscibroda. Protean: VM allocation service at scale. In *Proceedings of the 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI '20)*, pages 845–861. USENIX Association, 2020.

Xin Han, Francis Y. L. Chin, Hing-Fung Ting, Guochuan Zhang, and Yong Zhang. A new upper bound 2.5545 on 2D online bin packing. *ACM Transactions on Algorithms*, 7(4):Article 50, 2011. doi: 10.1145/2000807.2000818.

Sandy Heydrich and Rob van Stee. Beating the harmonic lower bound for online bin packing. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP '16)*, 2016. doi: 10.4230/LIPIcs.ICALP.2016.41.

Maria Teresa López-Herrero et al. Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud. *Procedia Computer Science*, 52:950–955, 2015. doi: 10.1016/j.procs.2015.05.171.

Matthias Nagel et al. Analysis of heuristics for vector scheduling and vector bin packing. In *Learning and Intelligent Optimization (LION 2023)*, volume 14286 of *Lecture Notes in Computer Science*. Springer, 2023. doi: 10.1007/978-3-031-44505-7_39.

Rina Panigrahy, Kunal Talwar, Lincoln Uyeda, and Udi Wieder. Heuristics for vector bin packing. Technical report, Microsoft Research, 2011. URL https://www.microsoft.com/en-us/research/publication/heuristics-for-vector-bin-packing/.

Steven S. Seiden. On the online bin packing problem. *Journal of the ACM*, 49(5):640–671, 2002. doi: 10.1145/585265.585269.

Steven S. Seiden and Rob van Stee. New bounds for multi-dimensional packing. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, pages 486–495. SIAM, 2003.

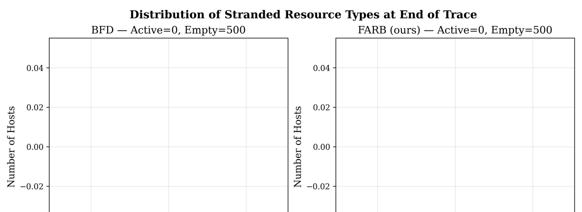
Junjie Sheng, Shengliang Cai, Haochuan Cui, Wenhao Li, Yun Hua, Bo Jin, Wenli Zhou, Yiqiu Hu, Lei Zhu, Qian Peng, Hongyuan Zha, and Xiangfeng Wang. VMAgent: Scheduling simulator for reinforcement learning. *arXiv preprint arXiv:2112.04785*, 2022. URL https://arxiv.org/abs/2112.04785.

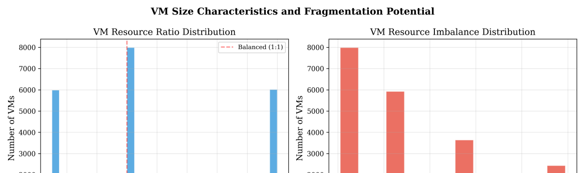
Weijia Song, Zhen Xiao, Qi Chen, and Haipeng Luo. Adaptive resource provisioning for the cloud using online bin packing. *IEEE Transactions on Computers*, 63(11):2647–2660, 2014. doi: 10.1109/TC.2013.148.

Muhammad Tirmazi, Adam Barker, Nan Deng, Md Ehtesam Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: The next generation. In *Proceedings of the 15th European Conference on Computer Systems (EuroSys '20)*. ACM, 2020. doi: 10.1145/3342195.3387517.

Abhishek Verma, Luis Pedrosa, Madhukar R. Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at Google with Borg. In *Proceedings of the 10th European Conference on Computer Systems (EuroSys '15)*. ACM, 2015. doi: 10.1145/2741948.2741964.

0 (0.0%) 0 (0.0%) 0 (0.0%) 0 (0.0%) 0 (0.0%) 0 (0.0%)





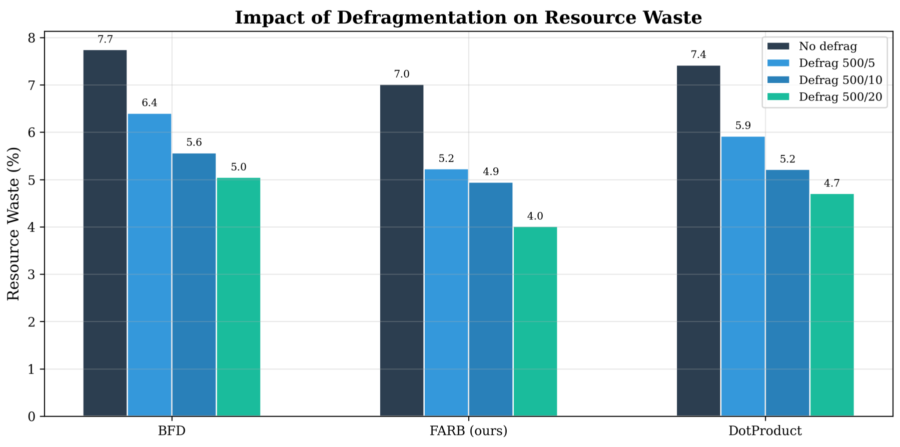


Figure 5: Waste reduction from defragmentation by base heuristic. Farb benefits more from defragmentation than BFD because its balanced host states are more amenable to consolidation. The combination of fragmentation-aware initial placement and periodic migration yields the lowest overall waste (4.01%).

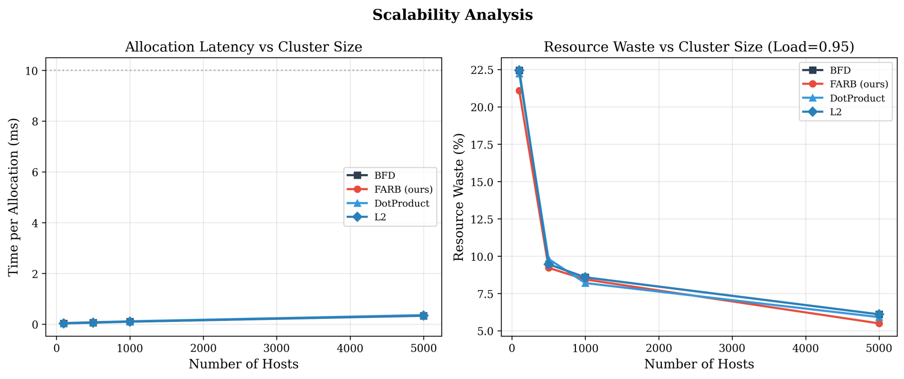


Figure 6: Allocation latency and resource waste vs. cluster size (load factor 0.95). Farb scales linearly with host count, maintaining <0.35 ms per allocation at 5,000 hosts. The waste improvement over BFD increases with cluster size (0.60 pp at 5,000 hosts), suggesting larger benefits at hyperscaler scale.

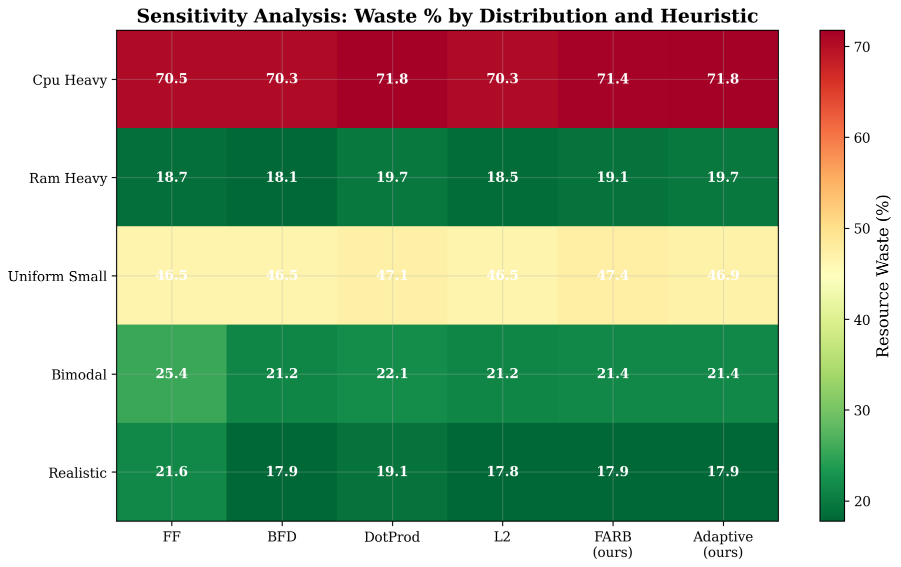


Figure 7: Sensitivity heatmap showing waste percentage across heuristics and workload distributions. Farb achieves the best results on Azure-like workloads with discrete VM types, where complementary resource ratios create opportunities for balance-aware placement. On homogeneous or extreme-skew workloads, all heuristics perform similarly.