

# HZ5: A Fail-Closed Descriptor-Owned Allocator Profile Family

Author: Tomoaki Kurata (TOMOAKI KURATA)

Contact: charmpic moecharm.dev@gmail.com

X (Twitter): <https://x.com/CharmNexusCore>

Draft: 2026-05-27 v0.1

## Abstract

HZ5 is a follow-up allocator line in the Hakozuna project. Unlike the earlier HZ3/HZ4 paper, which focuses on a dual-profile operating policy for local-heavy and remote-heavy small-object allocation, HZ5 studies a descriptor-owned, fail-closed allocator family optimized for low RSS and explicit ownership boundaries. In the Linux RUNS=10 unified sweep, HZ5 profile rows often occupy a different Pareto point from tcmalloc, mimalloc, HZ3, and HZ4: on selected main/mid/cross remote-pressure workloads, HZ5 matches or exceeds tcmalloc throughput while using substantially less RSS. For example, on `t=8 main r90, hz5-pagerun64-main` reached 60.07M ops/s with 30MB RSS, while tcmalloc reached 24.45M ops/s with 424MB RSS. On `t=8 mid_only r90, hz5-large128-source16` reached 75.15M ops/s with 8MB RSS, while tcmalloc reached 20.12M ops/s with 478MB RSS.

The result is not a universal allocator replacement. Local-only mixed rows, small-object guard rows, and some LargeFront rows remain stronger on tcmalloc, mimalloc, HZ3, or HZ4. The contribution of HZ5 is therefore narrower but important: it demonstrates that descriptor ownership, fail-closed routing, and profile-scoped source design can form a low-RSS allocator family without requiring every route to share one hot-path policy. This paper presents HZ5 as a standalone profile-family study and treats HZ6, a possible transfer-first successor, as future work.

## 1. Introduction

Modern general-purpose allocators are pulled between three goals: low latency, scalability under remote frees, and bounded memory growth. Hakozuna HZ3/HZ4 showed that a single allocator policy does not need to cover all workload shapes: HZ3 is effective for local-heavy/default usage, while HZ4 is stronger for remote-heavy/high-thread cases.

HZ5 begins from a different question:

`Can an allocator expose stronger ownership boundaries and lower RSS  
without pretending to be one universal default profile?`

The answer from the current evidence is yes, but only if HZ5 is treated as a profile family. HZ5 wins where its source and routing assumptions match the workload. It loses where the workload is better served by HZ3/HZ4 or by mature production allocators such as tcmalloc and mimalloc.

### 1.1 Reader Map

This paper is not a replacement for the HZ3/HZ4 paper. It is a second paper in the same design lineage.

HZ3/HZ4 paper:

Box Theory, PTAG32 routing, local-vs-remote dual profile operation.

HZ5 paper:

descriptor ownership, fail-closed route contracts, low-RSS profile families.

HZ6 future work:

transfer-first design with class transfer caches and RSS governance.

## 2. Design Goals

HZ5 is guided by four design goals.

### 2.1 Descriptor-Owned State

The allocator state should not rely solely on user data pages or intrusive metadata in inactive memory. Descriptor ownership makes allocator-visible state explicit and enables route validation, source accounting, and future memory return policies.

### 2.2 Fail-Closed Routing

If a pointer looks like it might belong to HZ5 but fails validation, it should not silently fall through to another allocator. This is the fail-closed rule:

foreign pointer:

not HZ5-owned; use the configured external path if allowed.

HZ5-owned valid pointer:

handle inside HZ5.

HZ5-looking invalid pointer:

fail closed; do not route to libc/tcmalloc/HZ3 fallback.

### 2.3 Profile Families Instead of a Single Default

HZ5 uses profile rows because one policy is not optimal for every object size, alignment, and remote-free shape. The paper-facing Linux rows are:

HZ5 row	Scope
hz5-pagerun64-main /	MidPage PageRun64 general and cross-size
hz5-pagerun64-cross128	profiles
hz5-large128-rss	low-RSS LargeFront profile
hz5-large128-source16	LargeFront 128K throughput comparison lane
hz5-large128-transfer128	transfer-cache diagnostic lane, not a default

## 2.4 Keep Diagnostics Out of Speed Runs

HZ5 development used many diagnostic lanes. The paper-facing speed runs disable attribution counters so atomic counter overhead does not pollute throughput.

## 3. Implementation Overview

HZ5 combines route contracts, source maps, and profile-specific backing policies. The exact implementation differs by platform, but the contract is shared:

```
route:
  owned / foreign / invalid

ownership:
  descriptor is the allocator source of truth

source:
  page/run or large-front profile owns backing memory

diagnostics:
  separated from speed lanes
```

On Linux, HZ5 currently has the paper-facing profile families evaluated in the RUNS=10 unified sweep. On Windows, HZ5 has additional Local2P and control-plane experiments, but those are not used as the main HZ5 paper backend because their route coverage and RSS behavior are not equivalent to the Linux paper rows.

## 4. Evaluation

### 4.1 Protocol

The paper-facing dataset is:

paper/results\_hz5\_linux\_20260526\_run10\_unified

The run uses:

- Ubuntu 22.04.5 LTS, Linux 6.8.0-90-generic
- AMD Ryzen 7 5825U
- GCC 11.4.0, glibc 2.35
- RUNS=10, median ops/s
- ru\_maxrss for RSS
- the same runner and same machine for HZ3, HZ4, mimalloc, tcmalloc, system, and HZ5 rows

### 4.2 Representative Positive Rows

Case	Best HZ5 row	HZ5 ops/s	HZ5 RSS	tcmalloc ops/s	tcmalloc RSS
t=8 cross128 r0	hz5- large128- rss	103.42M	9MB	47.43M	44MB
t=8 cross128 r50	hz5- pagerun64- main	23.47M	58MB	19.74M	103MB
t=8 cross128 r90	hz5- large128- rss	18.67M	84MB	11.20M	190MB
t=8 main r50	hz5- large128- source16	65.11M	19MB	23.13M	413MB
t=8 main r90	hz5- pagerun64- main	60.07M	30MB	24.45M	424MB
t=8 mid_only r50	hz5- large128- source16	76.63M	7MB	19.41M	516MB
t=8 mid_only r90	hz5- large128- source16	75.15M	8MB	20.12M	478MB
t=8 large128 r90	hz5- large128- source16	12.19M	156MB	12.71M	193MB

These rows show the HZ5 Pareto point: strong throughput on selected remote-pressure or mid/large profiles with much lower RSS than tcmalloc in many cases.

### 4.3 Negative Rows and Claim Boundaries

HZ5 should not be over-claimed. The unified result note records rows where HZ5 is not the winner:

Case	Winner	Winner ops/s	Best HZ5 ops/s	Read
t=8 guard r0	tcmalloc	250.99M	119.78M	Small-object local-only remains better served by tcmalloc.
t=8 guard r50	mimalloc	77.16M	20.61M	HZ5 guard remote rows are not competitive.

Case	Winner	Winner ops/s	Best HZ5 ops/s	Read
t=8 guard r90	mimalloc	66.56M	11.26M	HZ5 small-object remote path is not the claim. Local-only mixed rows remain tcmalloc/HZ3 territory. HZ5 wins remote-pressure mid rows, not local-only mid rows. LargeFront 128K remains profile-sensitive.
t=8 main r0	tcmalloc	237.13M	120.30M	
t=8 mid_only r0	tcmalloc	227.07M	121.58M	
t=8 large128 r50	tcmalloc	18.61M	15.25M	

The correct conclusion is not “HZ5 is faster than tcmalloc.” The correct conclusion is that HZ5 defines a separate low-RSS profile family with strong remote-pressure rows and explicit ownership safety.

## 5. Discussion

### 5.1 Why HZ5 Is Separate from HZ3/HZ4

HZ3/HZ4 optimize operating policies for small-object local and remote-heavy workloads. HZ5 optimizes a different axis: ownership explicitness and RSS under selected profile shapes. This makes HZ5 a sibling allocator line rather than a drop-in successor.

### 5.2 Windows Evidence

Windows Local2P experiments are valuable portability/control evidence but are not the Linux paper backend. Recent Windows observations show exact **64K/a8192** remote-throughput promise, while long phase-churn pressure is not a broad low-RSS win. Therefore, this paper keeps Windows HZ5 as supplemental evidence and uses Linux as the paper-facing backend.

### 5.3 Lessons from No-Go Lanes

HZ5 development produced several important no-go results:

- transfer-cache variants that helped narrow rows but did not become defaults
- Windows Local2P variants that improved remote throughput but were not broad RSS winners

- diagnostic/counter lanes that explained behavior but were excluded from speed tables

These negative results support the profile-family framing.

## 6. Limitations and Future Work

- HZ5 is not a universal allocator default.
- The strongest paper-facing results are Linux profile rows; Windows HZ5 remains supplemental unless a future backend reaches matching route coverage.
- Fragmentation is measured through RSS and retention observations rather than a formal fragmentation model.
- HZ6 should be evaluated as a separate transfer-first line, with class transfer caches and RSS governance designed as primary boxes rather than diagnostic add-ons.

## 7. Conclusion

HZ5 demonstrates that descriptor-owned, fail-closed allocator design can occupy a useful low-RSS Pareto point distinct from HZ3, HZ4, tcmalloc, and mimalloc. Its strongest evidence is not broad universal dominance, but profile-scoped strength: mid/main/cross remote-pressure rows with substantially reduced RSS and explicit ownership boundaries. HZ5 should therefore be published as a standalone profile-family paper and linked to, but not merged into, the HZ3/HZ4 paper lineage.

## References

The HZ5 standalone paper should cite the HZ3/HZ4 paper DOI as predecessor work, along with tcmalloc, mimalloc, jemalloc, and the Hakozen source repository.