vBOIDs: Taming Chaos via Coarse-grained Scheduling Abstraction for Containers

Kaesi Manakkal[*], Nathan Daughety[†], Yu Sun[‡], Marcus Pendleton[†], Hui Lu[*]

[*]***The University of Texas at Arlington***, [†]*Air Force Research Laboratory (AFRL)*, [‡]*Binghamton University*

USENIX OSDI 2026



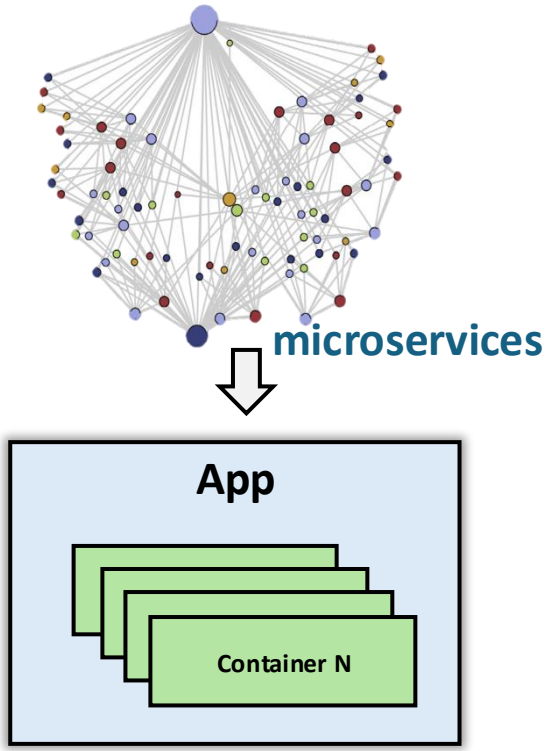




DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Background

- Many modern applications are built upon **microservices** architecture
  - A large app -> many smaller components that work together
  - Benefits:
    - Horizontal scaling, flexibility, fault tolerance, etc.
  - Deploying a microservice app → **containers**
    - Lightweight, easy to provision quickly

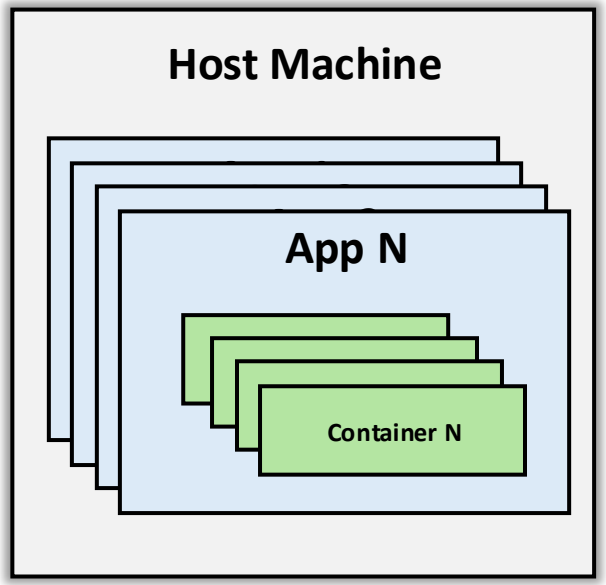


DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Background

- Efficiently serving microservices apps → Providers pack **thousands of containers** per machine
  - RunD[1]: **2,500**+ per node
  - Junction[2]: **3,000+** per node

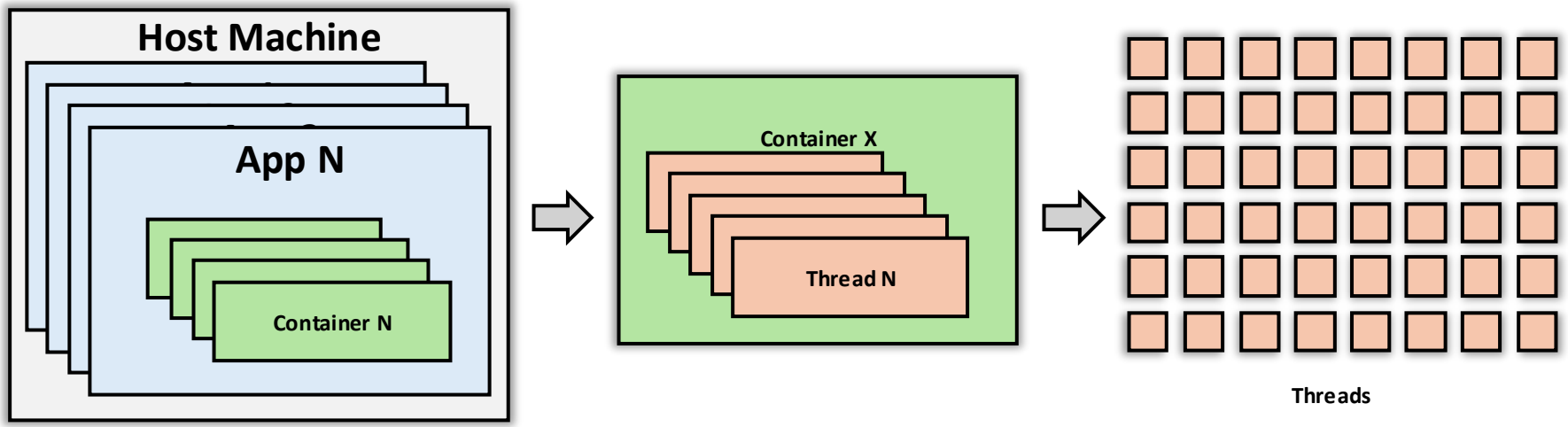
- A single microservice app could be made up of tens of containers could contain **hundreds of threads**
  - Hotel Reservation:
    - 24 services → 495 host threads



[1]Li, Zijun, et al. "{RunD}: A lightweight secure container runtime for high-density deployment and high-concurrency startup in serverless computing." *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. 2022.
[2]Fried, Joshua, et al. "Making kernel bypass practical for the cloud with junction." *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 2024.

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

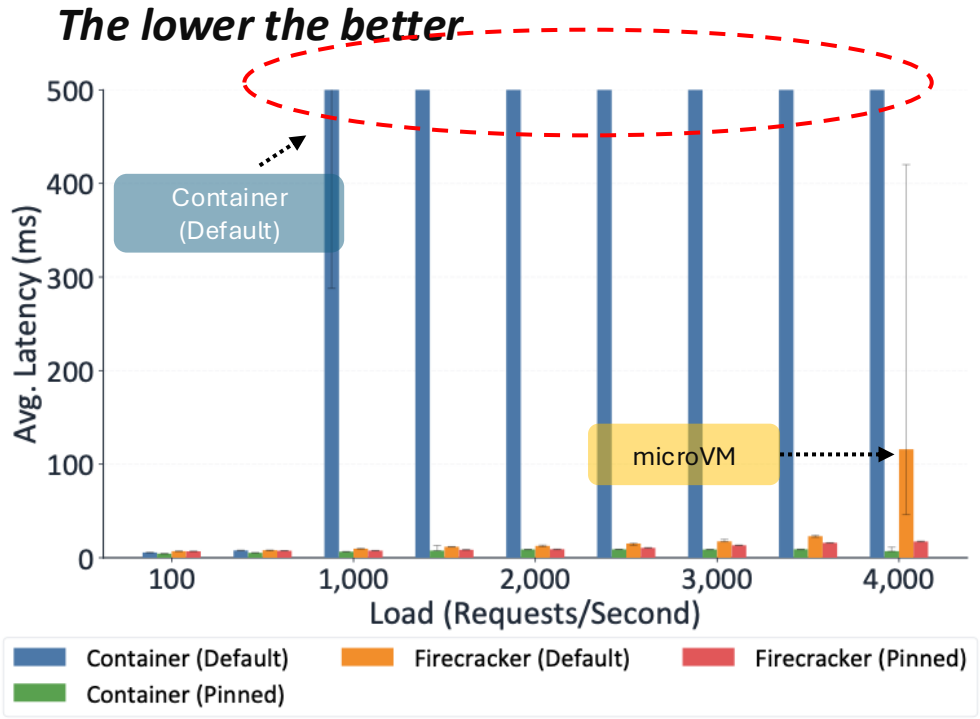
Motivation



- Modest rise in load can trigger a breakdown of hardware locality
- CPU scheduler moves threads faster than hardware state converges
  - TLB shootdowns (Inter-Processor Interrupts)
  - Cold L1/2 caches
  - Perturbed Last Level Cache (LLC)

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

“Scheduling Chaos” Example



Hotel Reservation[1]

- **24** microservices with **495** threads
- Default container case saturates before 1k RPS, violates 200 ms SLO
- Manual pinnning & micro-VMs scale smoothly
  - Because they limit inter-core migration

[1]https://github.com/delimitrou/DeathStarBench

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

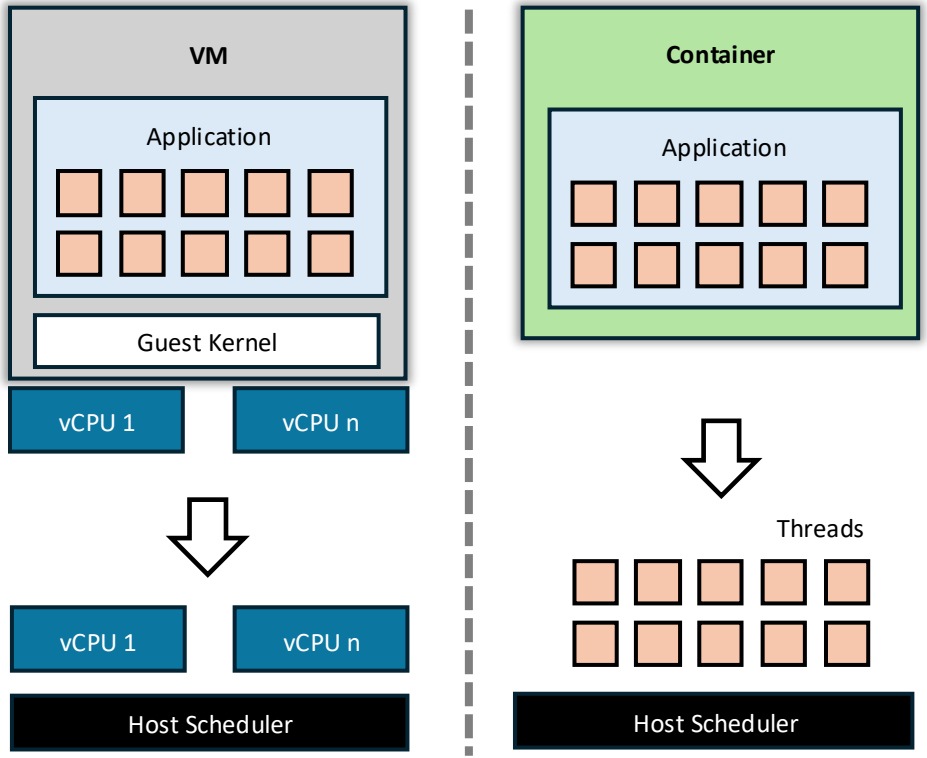
Architecture Comparison: VMs vs Containers

Virtual Machines (VMs)

- App threads hidden behind **small number of vCPUs**
- vCPUs are **long-lived**

Containers

- All app threads **directly visible** to host scheduler
- Threads typically **short-lived**



DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Root Cause Analysis

Intra-core switches are similar across configurations

- Workload-driven; preserve core-local state

Cross-core migrations differ by up to **3×**

- Disrupts caches, TLBs, and placement stability

Host-visible units at 1k RPS:

- Containers: **495 threads**
- Firecracker: **52 units**

The lower the better

Solution	Intra-core switches (#/s/core)	Cross-Core Migration (#/s/core)
Container (Default)	2677	316
Container (Pinned)	2699	117
Firecracker (Default)	3377	81
Firecracker (Pinned)	3559	2

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Root Cause Analysis

Intra-core switches are similar across configurations

- Workload-driven; preserve core-local state

The lower the better

Solution	Intra-core switches	Cross-Core
Firecracker (Default)	3377	81
Firecracker (Pinned)	3559	2

Key Takeaway

Performance loss comes from unstable cross-core placement, not necessary thread-level time-sharing.

Host-visible units at 1k RPS:

- Containers: **495 threads**
- Firecracker: **52 units**

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Insights

Static placement – ***pinning***

- Low, consistent latency 🙂
- Strands resources, requires manual tuning 😟

Dynamic balancing – ***unpinned (default CFS)***

- High resource utilization 🙂
- Per-thread cross-core migration results in large scheduling overhead 😟

MicroVMs – ***vCPUs***

- Less migration units → lower scheduling overhead 🙂
- Virtualization overhead vs containers 😟

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Research Question

Static placement – ***pinning***

- Low, consistent latency
- Strands resources, requires manual tuning
- Virtualization overhead vs containers

Dynamic balancing – ***default CFS***

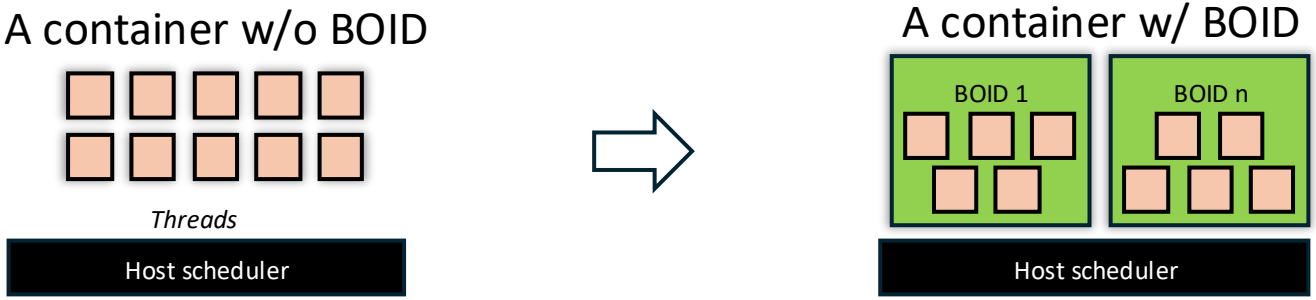
- High resource utilization
- Per-thread cross-core migration results in large scheduling overhead

Can containers expose ***fewer migration units***
- by decoupling thread execution from host migration, while avoiding the cost of full virtualization?

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Key Idea: BOID Abstraction

- **BOID** = **B**ound **O**bject **I**ntegrated **D**ispatch — a container-level virtual CPU
- Bundle a container's threads into a small set of BOIDs the host scheduler treats as migration units

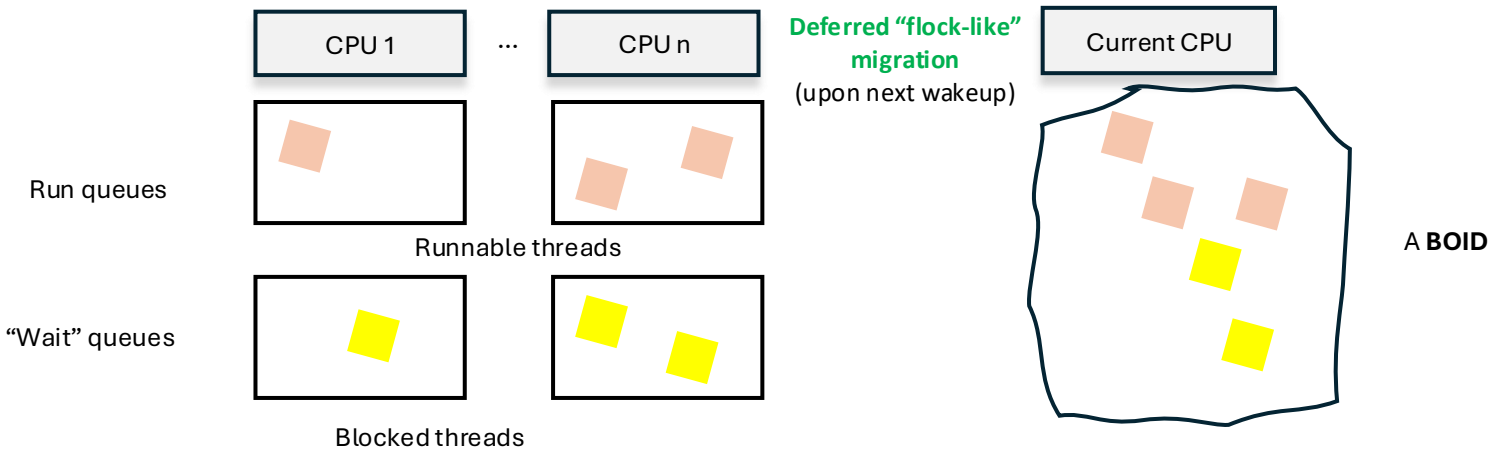


- Threads are still scheduled normally within a core
- Only host-level placement and migration move to BOID granularity

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Key Mechanism: Flock-like BOID Migration

- Moving a BOID must be cheap → **must not grab multiple runqueue locks**
- **Approach**: a *two-phase, lazy, state-based* protocol
  - *Logical* move: Single atomic CAS on **boid->new_cpu**
  - *Physical* move: On next wakeup, each thread observes new cpu and migrates itself (hook in **finish_task_switch**)

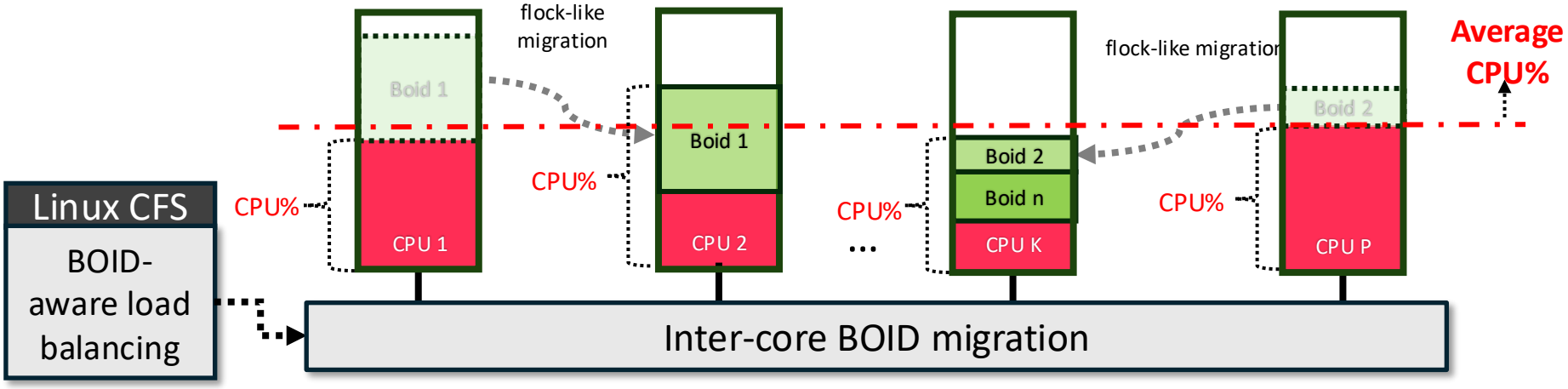


DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

BOID Scheduling

Inter-core BOID Balancer - *evens out load between CPUs*

- Moves BOIDs between CPUs on Host Machine
- Operates on coarse units - BOIDs
- Skips cache-hot BOIDs - locality-aware
- Avoids unstable moves - elephant BOIDs → delegate

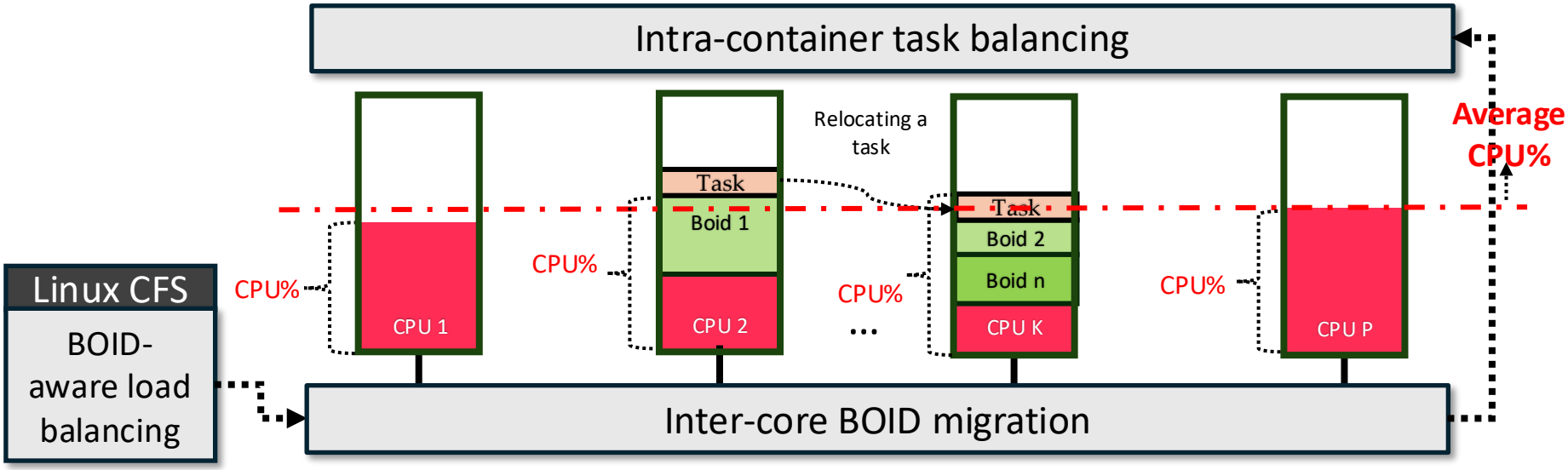


DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

BOID Scheduling

Intra-container Task Balancer - *evens out load between BOIDs*

- Moves threads between BOIDs in the same container
- Activated if internal imbalance appears
- Never exposes per-thread global migration



DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

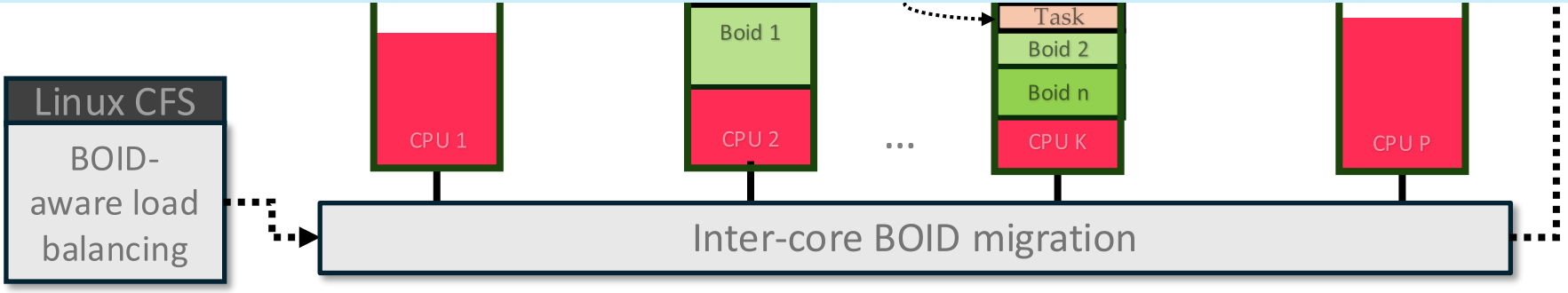
BOID Scheduling

Intra-container Task Balancer: Evens out load between BOIDs

- Moves threads between BOIDs in the same container
- Activated if internal imbalance appears
- Never crosses per-thread global migration

Net effect: Load-balancing search shrinks from O(# Threads) to O(# BOIDs)

BOIDs << # Threads → Less scheduling overhead



DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Implementation

- **~2,000 lines of code** — a patch to **Linux 6.15**
  - Primarily touches fair.c, pelt.c, cgroup.c
- Exposed via a new cgroup file, **cpu.boids**
  - Write N to reallocate the BOID array; **resizable at runtime** (no restart)

1. **Lazy, state-based migration**
   - Atomic CAS on boid->cpu + finish_task_switch hook
2. **O(1) incremental PELT aggregation**
   - a thread's load delta is atomically added to its BOID
3. **Lockless affinity (RCU):** p->boid via RCU
   - Placement (select_task_rq_fair) stays wait-free
4. **Tunable hysteresis**
   - Per-container migrate / balance cooldowns

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Evaluation Setup

Testbed

- 2× Intel Xeon Gold 6430 — 64 physical cores, 2 NUMA nodes × 32, SMT disabled
- 64 GB DDR5, NVMe SSD; Ubuntu 22.04, modified Linux 6.15

Workloads

- **Hotel Reservation (24 svc)**, Social Network (RPC-heavy, 27 svc), Media Services (CPU-bound, 33 svc)
- **Online Boutique on Kubernetes (11 svc), driven by Locust**

Baselines

- Container (Default / Pinned), Firecracker micro-VM (Default / Pinned)
- vBOIDs provisioned with the same # of BOIDs as cores / vCPUs

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Evaluation: Motivation Example

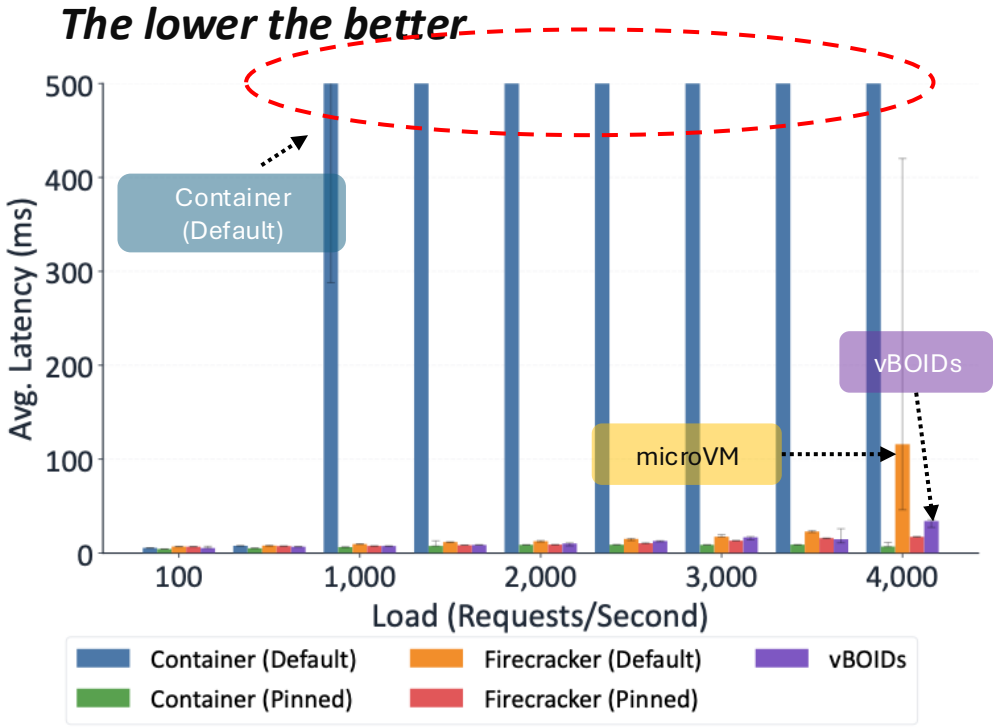
Default container: ~1k RPS

- vBOIDs reaches 4k RPS: **4× higher**

Why?

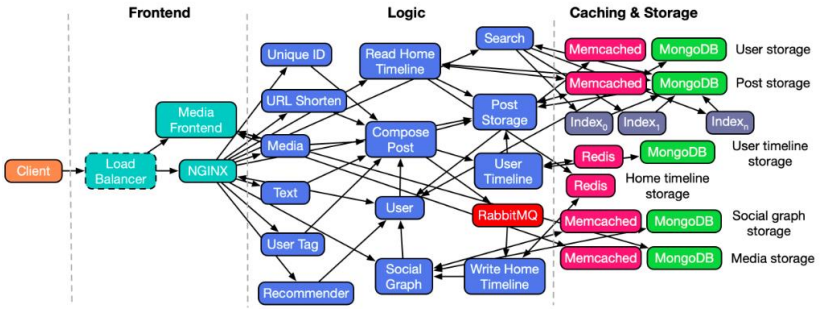
- 24 Microservices
  - **495** scheduler-visible units → **52 BOIDs**
- 1k RPS: **316** → **17** cross-core migrations

Reduced scheduler-visible units → reduced migration → higher performance



DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Evaluation: Social Network



More complex workload

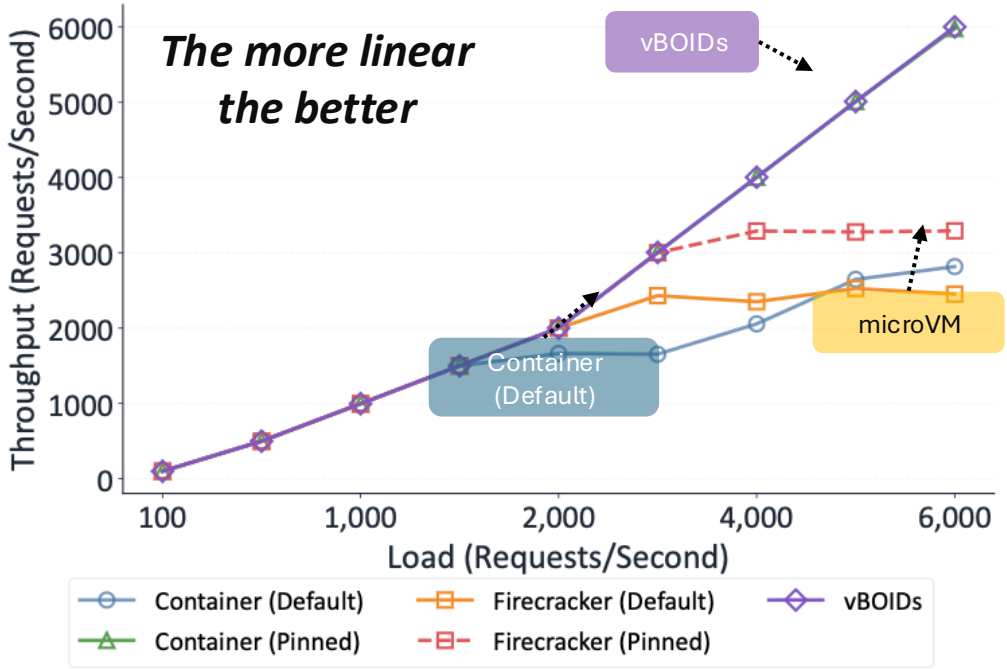
- Deeper service graph, more runnable threads

Default container saturates 2k RPS

- Firecracker reaches almost 3k
- vBOIDs sustains 5k RPS: **2.5× higher**

Why?

- **950** scheduler-visible units → **61 BOIDs**
- Far fewer cross-core migrations
  - 233 → 34 at 1K RPS



DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

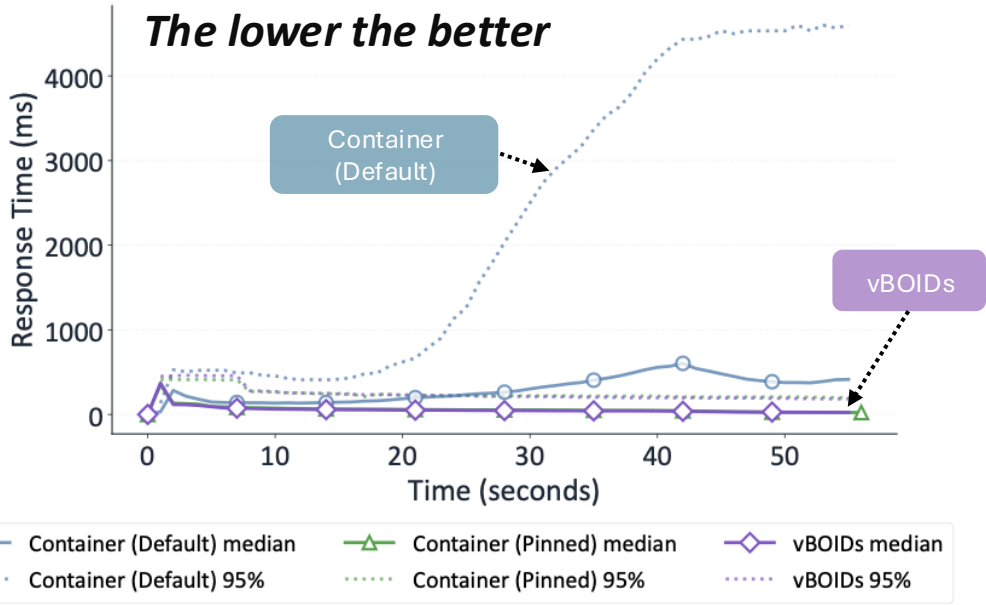
Evaluation: Online Boutique

Online Boutique on Kubernetes

- Default containers: **p95 > 4s**
- vBOIDs: **p95 < 500 ms,** matching pinned

Why it remains stable

- **359** scheduler-visible units → **31** BOIDs
- Fine-grained rebalancing stays within container



DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Summary

- **Problem:** per-thread host scheduling causes migration storms and destroys locality

- **Key idea:**
  - Group threads into BOIDs: container-level virtual cores for placement
  - Two-level scheduling: BOID placement + intra-container rebalancing

- **Result:** up to **3× higher throughput**, pinned-like stability

- **Deployment:** in-kernel → transparent to applications

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.

Thank you!

Kaesi Manakkal, Nathan Daughety[†], Yu Sun[‡], Marcus Pendleton[†], Hui Lu

kxm7920@mavs.uta.edu

github.com/kmanakk1/vboids

Supported by AFRL (FA8750-25-C-B038) and NSF (CCF-2415473, CNS-2415774)







DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited. Case Number AFRL-2026-0584.