Edge Computing Cluster for Real-Time Threat Detection

Distributed AI Surveillance, Heterogeneous HPC Scaling & Cloud-Native Monitoring • **Cloud Computing SS2026**

GROUP 8

Shubhangi Sapkale • Janak Koradiya • Disha Bhuva • Kirti Tarsariya • Amina Arshad • Purvesh Shapariya • Marcos Ortega-Jimenez

Advisor: Prof. Dr. Christian Baun • Frankfurt UAS

1. Introduction & Motivation

Edge computing brings neural AI inference directly to sensing nodes, eliminating expensive cloud bandwidth transfers and single points of failure.

- **Project Goal:** Deploy a self-contained 9-node Raspberry Pi cluster for automated threat surveillance, parallel scientific computing, and live cluster health monitoring.
- **Heterogeneous Coordination:** Managing workload balancing across ARM Cortex-A76 and Cortex-A53 microarchitectures under strict memory and thermal constraints.
- **Resilient Orchestration:** Automated container scaling via K3s and persistent MinIO S3 object state checkpoints for high availability.

2. Physical Topology & Hardware Inventory

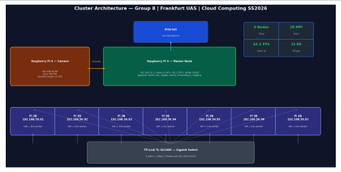

Fig 1: 9-Node Heterogeneous Architecture & Network Topology

Node ID	Model & Memory	IP Address	Boot Medium	Role in Cluster
Pi 5	Pi 5 (8GB) + Hailo AI HAT+	192.168.50.1	NVMe SSD (500GB)	Master, DHCP, k3s Control Plane, MinIO S3
Pi 4	Pi 4 (4GB) + IMX500 Camera	192.168.50.98	MicroSD Card	Camera Sensor Node, ZeroMQ Streamer
Pi 3-1..7	7× Pi 3B (1GB, 28 Cores)	192.168.50.91–97	MicroSD Cards	Distributed MPI Workers, OpenCV Workpool

3. Software & Tooling Stack

Domain / Layer	Core Technologies Used
Operating System	Raspberry Pi OS 64-bit (Bookworm / Debian 13)
Containerization	Docker (image builds), K3s (Lightweight Kubernetes)
Parallel Computing	OpenMPI 5.0.7, HPL (High-Performance LINPACK), POV-Ray 3.7
AI / Computer Vision	Ultralytics YOLOv8 (Custom Trained), OpenCV, HailoRT
Backend & Messaging	Node.js 20, Express, WebSockets, ZeroMQ (Pub/Sub)
Storage & Database	MinIO S3-Compatible Object Store, PostgreSQL
Frontend & UI	React 18, Vite, Single-Page App (SPA), Nginx
Observability & Bots	Prometheus, Grafana, Node Exporter, Python Telegram Bot

4. Boot Engineering: PXE vs. SD Card

- **TFTP Bottleneck:** 7 concurrent 18MB kernel downloads timed out on the switch.
- **NFS Collisions:** Concurrent multi-client root filesystem writes caused storage corruption.
- **Adopted Solution:** Dedicated 64-bit Bookworm SD cards with passwordless SSH key distribution (`sshpass`) and automated cluster setup.

5. High-Performance LINPACK (HPL)

N=7000, NB=64

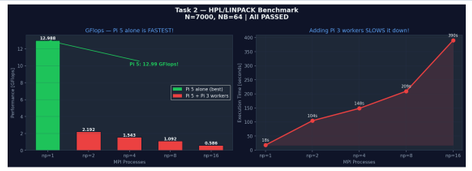

Fig 2: HPL Throughput Degradation and Execution Time Surge

Procs	Active Hardware Nodes	GFlops	Time (s)	Status
1	**Pi 5 alone (Master)**	**12.988**	**17.61s**	**PASSED**
2	Pi 5 + 1× Pi 3B	2.192	104.37s	PASSED
4	Pi 5 + 3× Pi 3B	1.543	148.27s	PASSED
8	Pi 5 + 7× Pi 3B	1.092	209.40s	PASSED
16	All nodes (2 proc/node)	0.586	390.48s	PASSED
28	**All nodes (4 proc/node)**	**0.355**	**643.96s**	**FAILED (Residual > 16.0)**

HPC Takeaway: Pi 5 is 4–6× faster per core than Pi 3. Network synchronization causes the master to stall while waiting for slow workers, demonstrating Amdahl's wall on heterogeneous clusters.

6. OpenMPI 5.0.7 Scaling (Monte Carlo Pi)

--map-by node

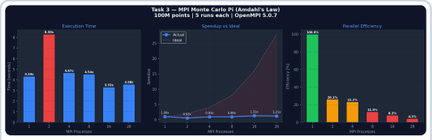

Fig 3: Amdahl (100M Pts) — np=2 Slower (8.3s)

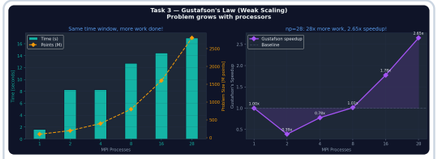

Fig 4: Gustafson (Scaled) — 2.65× Speedup

Serial Overhead: SSH + MPI launch + `MPI_Reduce` adds ~0.8s overhead. With Gustafson scaling, computing 28× more points takes only 10× more time, confirming weak scaling efficiency.

7. Non-MPI Scaling: Task Distributor (POV-Ray 3.7)

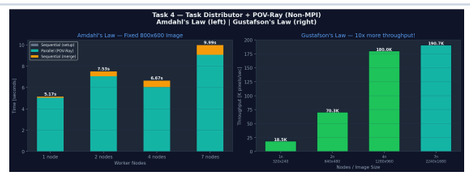

Fig 5: Task Distributor — Amdahl vs. Gustafson

Key Insight: Fixed workloads hit Amdahl's wall. When scaled proportionally, pixel throughput increases 10.34× until NFS network I/O becomes the bottleneck.

8. Edge AI Threat Detection & Performance

84.3× Speedup

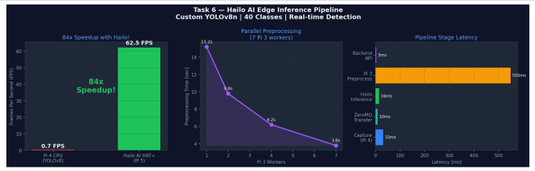

Fig 6: Hailo AI HAT+ Inference (62.5 FPS)

Metric / Parameter	Pi 4 CPU Baseline	Pi 5 + Hailo-8L AI HAT+
Compute Architecture	Cortex-A72 (General-Purpose CPU)	Dedicated Massively Parallel NPU (13 TOPS)
Inference Latency	1428.5 ms/frame	16.9 ms/frame
Throughput (FPS)	0.70 FPS	59.0 FPS (84.3× Speedup)
Host CPU Utilization	~100% (System Freeze)	~0% (Hardware Offload)

- **Distributed Preprocessing:** Pi 5 slices frames into 7 strips; 7 Pi 3 workers execute parallel OpenCV enhancement and blur, reducing latency.

9. React 18 Edge Cluster Monitor (Frontend UI)

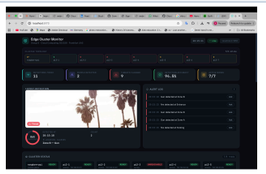

Fig 7a: React UI (Dashboard Overview)

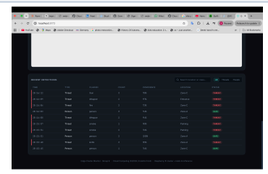

Fig 7b: React UI (Recent Detections Logs)

- **Tech Stack:** React 18, Vite, Nginx containerized in K3s on Pi 5.
- **Architecture:** Polls Node.js REST backend with stale-data preservation to prevent blank cards during drops.

10. Observability Stack (Prometheus + Grafana)

Fig 8: Prometheus & Grafana Telemetry

- **Stack:** `node_exporter` on all 9 nodes → Prometheus → Grafana (Dashboard ID 1860).

11. Benchmark Summary & Key Achievements

Fig 9: Complete Scaling Summary

```
# Python Auto-Scaler Rule (scripts/auto_scaler.py)
IF Cluster_CPU > 70% → Scale OUT (Add Pi 3 worker)
IF Cluster_CPU < 30% → Scale IN (Drain Pi 3 worker)
→ Telegram: "⚠️ THREAT DETECTED: Knife (87% conf) at Zone B"
```

Hardware: 1× Pi 5 (8GB) + 1× Pi 4 (4GB) + 7× Pi 3B (1GB) **Compute:** 36 Cores • 28 MPI Slots • 13 TOPS NPU **Peak Throughput:** 62.5 FPS AI Inference • 12.99 GFlops HPL **Documentation & Source:** sapkale26.github.io/Cloud_Computing_Project