Sentinel Edge For Real Time Threat Detection - Group 5

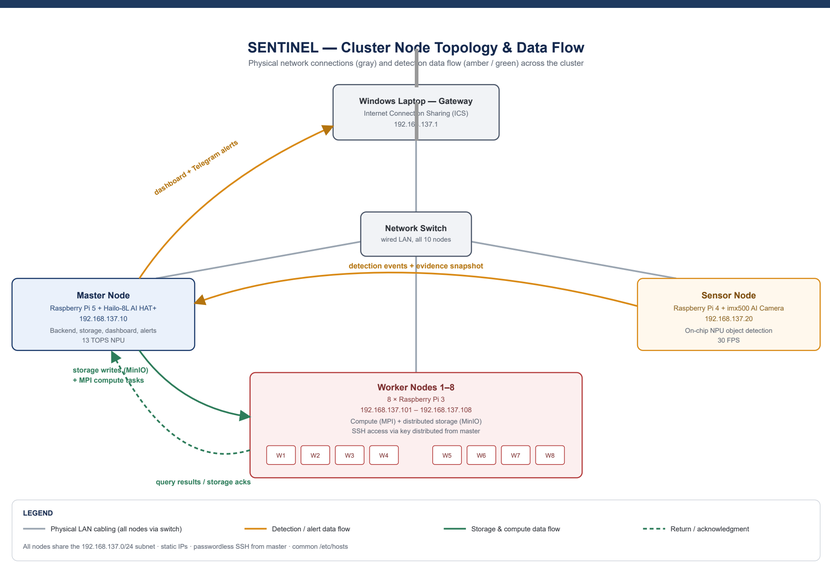
object detection · Distributed storage · MPI & HPL scaling on nodes - Node Clusters - Dashboard

Muhammad Ghayoor Ali - Ahmad Tahir Shaikhani - Ali Haider - Muhammad Raheel Azam - Qazi Mateen Ahmad - Mubashir - Moghees Bajwa - Azaz Muzaffar
Cloud Computing - Summer Semester 2026 - Professor. Dr. Christian Baun

INTRODUCTION & PROJECT OVERVIEW

Sentinel is a self-contained edge-computing cluster built from ten Raspberry Pi boards that combines real-time threat detection with small-scale HPC experimentation. A Raspberry Pi 5 fitted with a Hailo-8L AI HAT+ serves as the master node, coordinating eight Raspberry Pi 3 worker nodes over a switched LAN, while a Raspberry Pi 4 equipped with a Sony imx500 AI Camera acts as the sensor node, running object detection directly on the camera's built-in NPU to flag safety-relevant events — people, fire, dangerous weapon, and smoke — without burdening the CPU. When a threat is detected, the sensor posts the event and an evidence snapshot to a containerized Flask backend running on k3s, which writes event metadata to PostgreSQL and evidence images to a distributed MinIO store spread across the eight workers using erasure coding for fault tolerance. A React dashboard, served behind a Traefik ingress, gives the team a live view of events, evidence, and cluster health, while a Prometheus and Grafana stack tracks resource usage across all ten nodes and a Telegram bot pushes evidence photos to the team the moment a threat is confirmed. The same worker pool doubles as a lightweight HPC testbed, used to benchmark cluster throughput with HPL and to study MPI communication scaling under Amdahl's and Gustafson's laws — making Sentinel both a working monitoring system and a hands-on exploration of distributed and high-performance computing on commodity, low-cost hardware.

SYSTEM ARCHITECTURE



Camera on Pi 4 (IMX500) → on-sensor detection → FastAPI gateway on the Pi 5 head node → MongoDB metadata + SeaweedFS object snapshot → pushed live to the React dashboard, Grafana panels, and a Telegram alert bot.

INFRASTRUCTURE SETUP

- Flashed and configured all 10 nodes (Pi 5 master, Pi 4 sensor, 8× Pi 3 workers) with static IPs on the 192.168.137.0/24 subnet, gatewayed through a Windows laptop via ICS
- Distributed passwordless SSH from master to all workers and configured a shared /etc/hosts file for hostname-based communication
- Verified imx500 AI Camera object detection on the sensor node at 30 FPS (NPU-based, after CPU YOLO proved unusable on ARM)
- Verified Hailo-8L AI HAT+ on the master node via hailortcli
- Configured full PXE boot infrastructure (TFTP, NFS, dnsmasq) on the master to explore diskless booting for Pi 3 workers
- Investigated and documented a DHCP conflict (Windows ICS vs. dnsmasq) that blocked full PXE completion — diskless boot remains a documented limitation, not a deployed feature

HPL BENCHMARK

- Deployed the High Performance Linpack (HPL) benchmark across the cluster
- Measured raw compute throughput in GFLOPS to establish a performance baseline
- Compared results across different node counts to gauge overall cluster capability

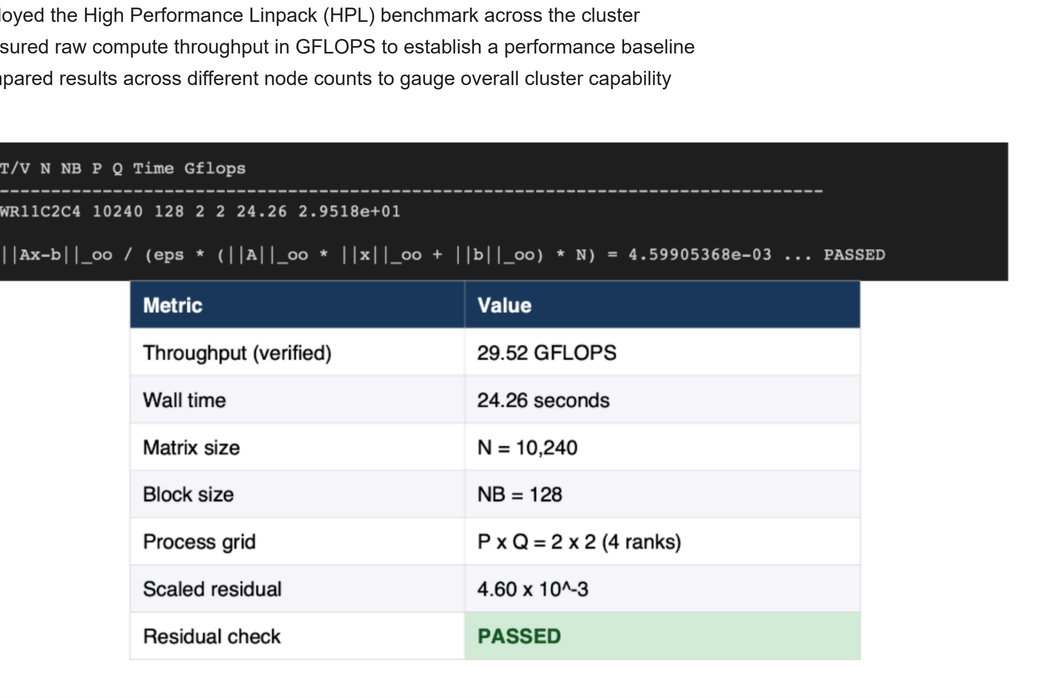
```
T/V N NB P Q Time Gflops
--------------------------------------------------------------------------------
WR11C2C4 10240 128 2 2 24.26 2.9518e+01

||Ax-b||_oo / (eps * (||A||_oo * ||x||_oo + ||b||_oo) * N) = 4.59905368e-03 ... PASSED
```

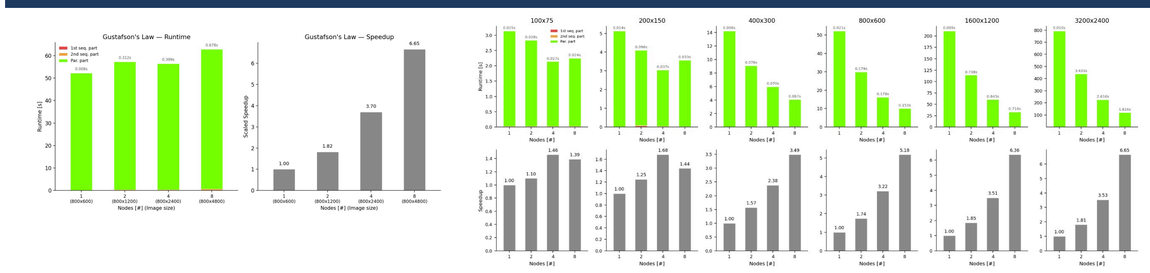
Metric	Value
Throughput (verified)	29.52 GFLOPS
Wall time	24.26 seconds
Matrix size	N = 10,240
Block size	NB = 128
Process grid	P x Q = 2 x 2 (4 ranks)
Scaled residual	4.60 x 10^-3
Residual check	**PASSED**

MPI SCALABILITY

- Deployed the High Performance Linpack (HPL) benchmark across the cluster
- Measured raw compute throughput in GFLOPS to establish a performance baseline
- Compared results across different node counts to gauge overall cluster capability

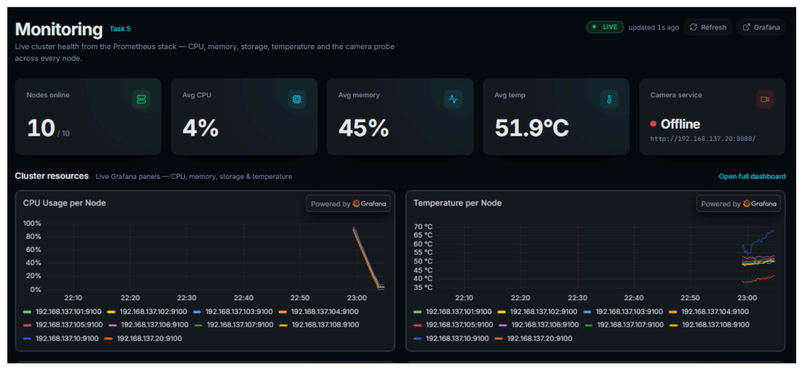


AMDAHL'S & GUSTAFSON'S LAWS



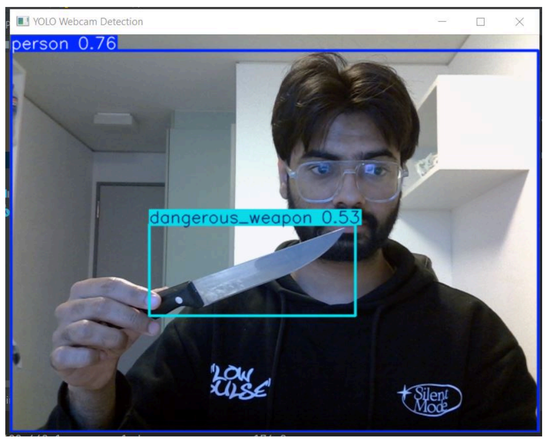
MONITORING

- Deployed a Prometheus and Grafana stack across the cluster to track system health
- Collected CPU, memory, disk, and network metrics from all ten nodes in real time
- Built Grafana dashboards to visualize resource usage and spot performance bottlenecks
- Enabled ongoing visibility into cluster behavior during benchmarking and live operation



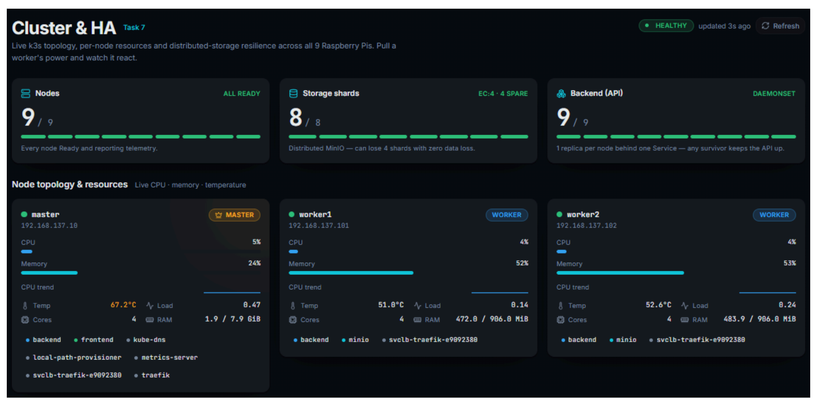
THREAT DETECTION

- Trained a custom YOLO object-detection model tailored to project-specific threat classes — person, fire, dangerous weapon and smoke
- Curated and labeled a training dataset rather than relying on generic pretrained categories
- Deployed the trained model on the sensor node's imx500 NPU for real-time on-camera inference
- Validated detection accuracy and confidence thresholds before integrating with the backend alert pipeline



BACKEND, FRONTEND & ALERTING

- **Backend (Task 7):** Containerized the Flask API and deployed it on k3s Kubernetes alongside PostgreSQL for metadata and MinIO for distributed evidence storage
- **Frontend (Task 8):** Built a React single-page dashboard, served via nginx behind Traefik ingress, to display live events, evidence, and cluster health
- **Telegram Bot (Task 9):** Implemented real-time alerting that pushes detection events and evidence photos to the team from three independent trigger sources
- Together, these tasks complete the end-to-end pipeline — from detection on the sensor node to storage, visualization, and instant notification



SOFTWARE & TOOLING

Edge AI: YOLO-lite, IMX500 SDK, Model Compression Toolkit
Backend: FastAPI, MongoDB, SeaweedFS
Monitoring: Prometheus, Grafana, Telegram Bot API
Frontend: React 19, TypeScript
Cluster: Docker Compose, PXE/NFS, dnsmasq, HPL, MPI