Scalable OpenCHAMI Node Infrastructure Collection (SONIC)

Detim Zhao
Arizona State University
Tempe, AZ, USA
detimzhao@gmail.com
0009-0004-1919-5386

Isabella Staley
Wayne State University
Detroit, MI, USA
isabellastaley62@gmail.com
0009-0006-5508-1487

Brighton Sagrans
University of Delaware
Newark, DE, USA
blsgrit@icloud.com
0009-0003-7076-5439

Deb Rezanka
Los Alamos National Laboratory
Los Alamos, NM, USA
drezanka@lanl.gov

Aiden Phillips
Los Alamos National Laboratory
Los Alamos, NM, USA
mphillips@lanl.gov

John Dermer
Los Alamos National Laboratory
Los Alamos, NM, USA
jwdermer@lanl.gov

Matthew Trujillo
Los Alamos National Laboratory
Los Alamos, NM, USA
mvt@lanl.gov

***Abstract*—Reliable hardware telemetry is foundational to HPC cluster operations, enabling proactive thermal management, power optimization, and failure prediction. HPC monitoring systems often struggle with data loss, service interruptions, and scalability limitations. This work presents a BMC telemetry system deployed on Los Alamos National Laboratory's SI research cluster that achieves high availability through event-driven architecture combining intelligent polling, triple-redundant data paths, and fault-isolated services. Intelligent polling reduces network overhead by 81% while maintaining sub-second latency. A centralized consumer with batched writes achieves 82% reduction in database CPU utilization. Load testing collected over 77,000 samples across multiple load levels, demonstrating counterintuitive inverse scaling where consumer performance improves as cluster load increases. The poster will present the hybrid architecture pipeline, performance metrics under realistic workloads, and Grafana visualization dashboards demonstrating operational deployment on a nine-node production testbed.**

***Index Terms*—HPC, BMC, Redfish, telemetry, Kafka, InfluxDB, monitoring, OpenCHAMI, systemd, Grafana**

I. Introduction

A. Problem Context

HPC facilities transitioning from proprietary frameworks like Cray System Management (CSM) to open-source solutions such as OpenCHAMI require real-time hardware monitoring. While BMC Redfish APIs provide vendor-neutral telemetry access, continuous push-based monitoring generates 17,280 HTTP requests daily per BMC, saturating management networks. Single points of failure risk system-wide monitoring loss, while data persistence is required for compliance and disaster recovery.

B. Research Objectives

This work validates architectural solutions on Los Alamos National Laboratory's SI research cluster, a nine-node testbed. Objectives include: (1) reducing network overhead while maintaining real-time capabilities, (2) eliminating single points of failure, (3) achieving production-grade reliability with fault isolation, (4) validating scalability under realistic workloads, and (5) demonstrating operational maturity.

II. Methods

A. System Architecture

The system monitors nine Intel OpenBMC nodes with 144 sensors collecting power, thermal, PSU, and CPU metrics (Fig. 1).

Hybrid Polling. Telegraf agents with Redfish plugins poll at 15-30 second intervals. Independent systemd services per BMC (telegraf-bmc@101-109) provide fault isolation. Compared to continuous push (288-576 requests/min, avg 432), polling reduces HTTP requests by 81% to 81/min.

Triple-Output Streaming. Data flows simultaneously to Apache Kafka (live streaming, KRaft mode), InfluxDB (time-series analysis, 7-day retention), and JSONL files (disaster recovery). All outputs achieved 100% write success.

Centralized Consumer. A Python consumer performs batched InfluxDB writes from Kafka, replacing nine independent Telegraf writers. This achieves 82% CPU reduction (1.5% to 0.27%) with 27.3 MB memory and 0.32% CPU footprint.

Resources. Nine Telegraf services use 225 MB RAM and 7% CPU. Kafka and InfluxDB containers require 1.2 GB RAM and 15% CPU. Storage grows 45 MB daily, totaling 320 MB under 7-day retention.

B. Load Testing

VPIC [6], a particle-in-cell physics simulation, via SLURM across four load levels (Table I): idle (0 cores), light (72 cores, 33%), medium (144 cores, 67%), and heavy (216 cores, 100%). Testing spanned 5.5 hours of allocated monitoring windows with 3 hours of active computational load, collecting 77,000+ consumer lag samples. Allocated durations include baseline and stabilization periods between load transitions.

[1]Durations represent allocated monitoring windows including baseline collection and stabilization periods. Active VPIC computational load totaled approximately 3 hours.

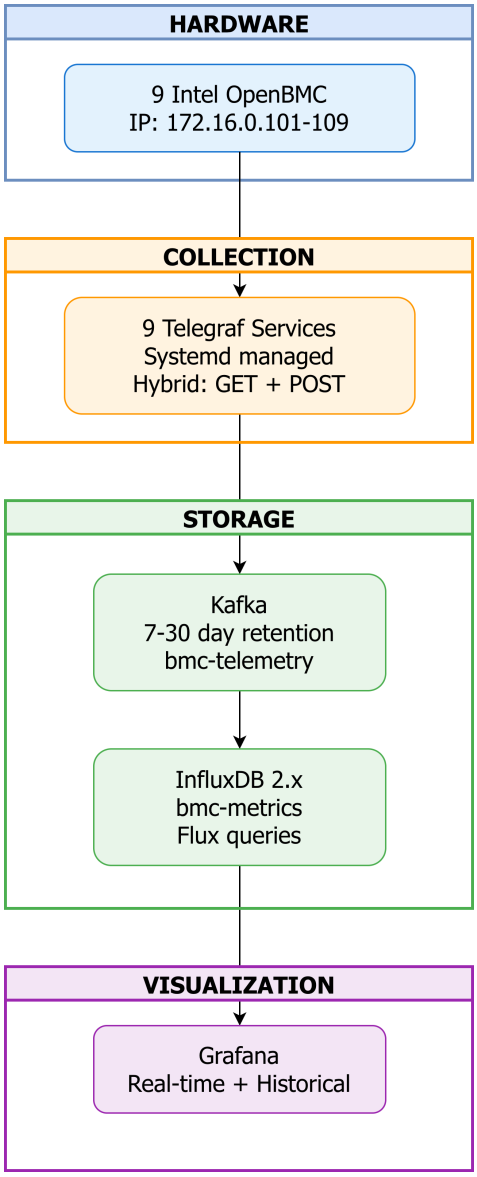


Fig. 1. SONIC hybrid architecture pipeline: Stage 1 (Hardware) - BMC sensors expose Redfish API; Stage 2 (Collection) - Telegraf polls endpoints with GET requests and event subscriptions; Stage 3 (Storage) - Parallel data streams to Kafka, InfluxDB, and JSONL; Stage 4 (Visualization) - Grafana queries InfluxDB using InfluxQL.

TABLE I
LOAD TESTING CONFIGURATION[1]

Load Level	Cores Used	Utilization	Duration
Idle	0	0%	1 hour
Light	72	33%	90 min
Medium	144	67%	120 min
Heavy	216	100%	120 min

III. ALTERNATE WORKFLOW: SYSTEMD TIMERS

An alternative implementation uses ten bash scripts with curl to collect BMC data, orchestrated via SLURM-scheduled jobs. Data flows through temporary JSON files, jq processing into InfluxDB line protocol, and HTTP POST to separate InfluxDB buckets. Systemd timers automate collection at 30-second to 15-minute intervals. This simpler approach provides broader endpoint access through GET-only requests, trading architectural sophistication for flexibility.

IV. RESULTS

A. Network Efficiency and Inverse Scaling

Intelligent polling reduces HTTP requests by 81% (432 to 81/min aggregate) while maintaining sub-second latency. Kafka's hash-based partition routing enables parallel processing.

Consumer lag demonstrated inverse scaling (Table II): lag decreased from 1.76 messages at light load to 1.54 at maximum load due to InfluxDB batching efficiency at higher message rates.

TABLE II
CONSUMER PERFORMANCE UNDER LOAD

Load	Util.	Avg Lag	Max Lag	CPU %
Idle	0%	1.17	9	0.31%
Light	33%	1.76	15	0.22%
Medium	67%	1.62	13	0.22%
Heavy	100%	1.54	14	0.32%

The system achieved 82% InfluxDB CPU reduction and 64-fold capacity headroom, with 100% job success across 54 SLURM jobs.

B. Operational Reliability

The system maintained 100% uptime over two weeks with zero data loss. Authentication failures were handled through per-BMC isolation with automatic retry within one polling interval. Independent service failures enabled continued monitoring of unaffected nodes.

Grafana dashboards (Fig. 2, 3) provide real-time visualization with InfluxQL queries for historical analysis.

V. DISCUSSION

A. Key Contributions

Centralized Consumer. Batched writes consolidate nine write streams, achieving 82% CPU reduction while improving fault tolerance and enabling independent component scaling.

Inverse Scaling. Performance improvement under load validates batching efficiency, suggesting batching architectures perform better at higher loads.

Fault Isolation. Per-BMC systemd services provide resilience unavailable in monolithic architectures, validated through authentication failure isolation with minimal impact.

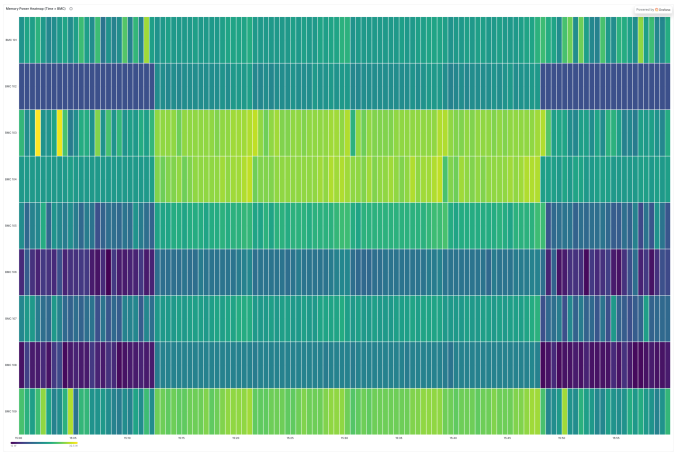


Fig. 2. Memory Power Heatmap (2026-07-15 9:00-10:00 AM). Rows represent BMCs, columns represent time intervals. Viridis colormap with cool colors (purple) indicating low power and hot colors (yellow) indicating high power, enabling pattern and anomaly detection.

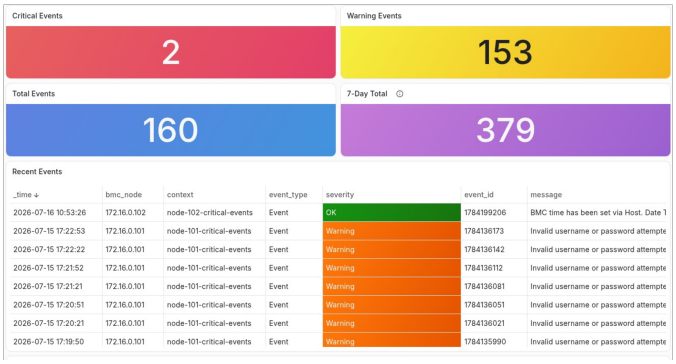


Fig. 3. BMC Events Dashboard (2026-07-13 to 2026-07-17) showing critical events count, warning events count, total events count, 7-day total, and recent events table with columns for time, BMC node, severity, event type, event ID, message, and context queried from InfluxDB bmc_metrics bucket.

B. Scalability and Future Work

The system demonstrates 64-fold capacity headroom. Projected 10× growth (90 BMCs) requires no infrastructure changes; 50× growth (450 BMCs) benefits from multi-consumer architecture.

Future work includes: **Agentic Grafana** for 1000+ node monitoring, **kernel-level integration** through BIOS/UEFI hooks for pre-boot validation, **full automation** with zero-touch deployment, **multi-source integration** correlating LDMS, SLURM, and OpenCHAMI data, and **vendor expansion** supporting Dell iDRAC and HPE iLO platforms. Current limitations include single-consumer bottleneck beyond 500 BMCs, 7-day retention constraints, and Intel OpenBMC specificity.

VI. Conclusion

This work demonstrates production-grade BMC telemetry using open-source components. The event-driven architecture combines intelligent polling (81% network reduction), triple-redundant data paths (100% write success), centralized consumer (82% CPU reduction), and per-node isolation (100% uptime).

Key contributions include: (1) centralized consumer architecture achieving 82% CPU reduction, (2) validation of inverse scaling through 5.5-hour load testing with 77,000+ samples, (3) 100% uptime with zero data loss at LANL SI cluster, (4) comprehensive automation demonstrating operational maturity, and (5) 64-fold capacity headroom.

The nine-node deployment validates design principles for larger production environments. The open-source implementation provides a replicable model for institutions requiring reliable hardware telemetry. Kafka-based architecture enables future extensibility for alerting, machine learning, and workload correlation without modifying core infrastructure.

Acknowledgment

This research was conducted at Los Alamos National Laboratory's SI research cluster. We thank our mentors for their guidance and support.

AI tools (Claude Sonnet 4.5) assisted with manuscript preparation, including verification of numeric consistency, grammar improvements, and formatting. All research methodology, experimental design, data analysis, and scientific conclusions are the original work of the authors, who verified all content for accuracy.

References

[1] Distributed Management Task Force, "Redfish Specification," version 1.0, 2015. [Online]. Available: https://www.dmtf.org/standards/redfish

[2] OpenCHAMI Project, "Open Composable Heterogeneous Adaptable Management Infrastructure," 2024. [Online]. Available: https://github.com/OpenCHAMI

[3] Apache Kafka Documentation, "Apache Kafka: A Distributed Streaming Platform," Apache Software Foundation, 2024. [Online]. Available: https://kafka.apache.org/documentation/

[4] InfluxData, "InfluxDB: Time Series Database," 2024. [Online]. Available: https://www.influxdata.com/

[5] Telegraf Documentation, "Telegraf Plugin-Driven Server Agent," InfluxData, 2024. [Online]. Available: https://www.influxdata.com/time-series-platform/telegraf/

[6] R. Bird, N. Tan, S. V. Luedtke, S. L. Harrell, M. Taufer, and B. Albright, "VPIC 2.0: Next generation particle-in-cell simulations," IEEE Trans. Parallel Distrib. Syst., vol. 33, no. 4, pp. 952–963, Apr. 2022.

[7] Grafana Labs, "Grafana: The Open Observability Platform," 2024. [Online]. Available: https://grafana.com/

[8] Intel Corporation, "OpenBMC: Open Source BMC Firmware," 2024. [Online]. Available: https://github.com/openbmc/openbmc