Karim Elezabawy

Machine Learning for Predictive Maintenance On HPC Nodes

Supervised by: Dr.Christian Boehme

The Cost of Unplanned Downtime

90% of enterprises estimate that

one hour of unplanned downtime costs

at least **$300,000**

ITIC, *2024-2025 Global Server Hardware and Server OS Reliability Survey*, 2024

The Growing HPC Energy Challenge

- Data centers consume **1.5% of global electricity**
- Growing rapidly due to AI/ML demands
- **Goal:** Can we predict failures *before* they happen?

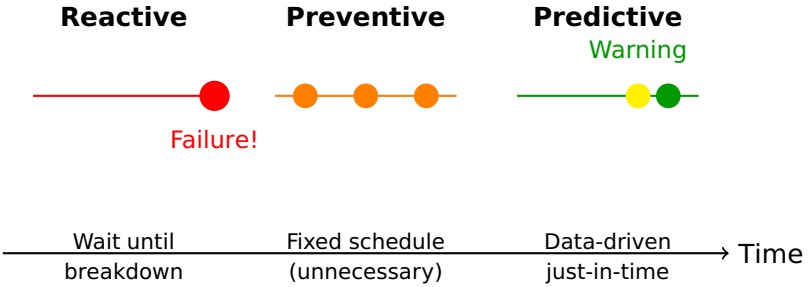
Image source: https://gwdg.de/hpc/

International Energy Agency, *AI is Set to Drive Surging Electricity Demand from Data Centres*, 2024

Table of contents

Background

Three Maintenance Strategies



What Makes HPC Systems Unique?

- **Scale & Complexity** — Thousands of components per cluster
- **High Utilization** — 24/7 operation with minimal downtime tolerance
- **Resource Intensity** — Massive power consumption and thermal loads
- **Failure Characteristics** — Rare but costly failures with cascading effects

HPC systems require specialized maintenance approaches due to their scale and criticality

Operational Data Analytics Framework

Three-Layer Architecture

1. **Monitoring system** — Collects operational data
2. **Machine learning applications** — Analyze data and provide insights
3. **Combination layer** — Relates results to end users

Key principle:

- Only tight integration of all three layers provides added value

Netti, Müller, et al., “From Facility to Application Sensor Data: Modular, Continuous and Holistic Monitoring with DCDB”, 2021

Monitoring Systems in HPC

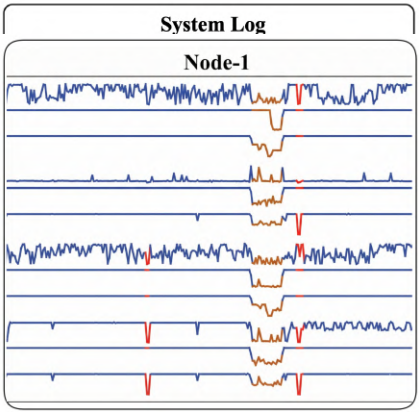
- **Log Monitoring Data:**
  - System logs, error messages
  - Application logs
  - Unstructured/semi-structured text

System Log	
1	Receiving block blk_-5632101276183739500 src: /10.251.27.63:36433 dest: /10.251.27.63:50010
2	BLOCK* NameSystem.allocateBlock: /user/root/randtxt4/_temporary/task_200811101024/part-00999. blk_-5632101276183739500
3	Receiving block blk_-4506306395053060141 src: /10.251.193.175:40709 dest: /10.251.193.175:50010
4	PacketResponder 0 for block blk_-5632101276183739500 terminating
...	...

LogLLM: Log-based Anomaly Detection
DOI: 10.48550/arXiv.2411.08561

Monitoring Systems in HPC

- **Log Monitoring Data:**
  - System logs, error messages
  - Application logs
  - Unstructured/semi-structured text
- **Node Telemetry Data:**
  - Hardware sensors (temperature, power)
  - CPU/GPU metrics
  - Network metrics
  - Time-series data



Effective Node-Level Anomaly Detection
DOI: 10.1145/3712285.3759794

Taxonomy of Methods for HPC Predictive Maintenance

- Traditional ML
  - Random Forest
  - XGBoost/LGBM
- Autoencoders
  - Dense AE
  - VAE
- Deep Learning
  - LSTM
  - LSTM-AE
  - Transformer MoE
- LLM-Based
  - BERT + Llama
- Graph-Based
  - Graph2Vec + CatBoost

Traditional ML

Traditional ML Approaches

Netti, et al. (2019)

- Random Forest
- Per-resource type models

E2EWatch (2021)

- LGBM/XGBoost ensemble
- Multi-class diagnosis

Key Observation

Both use **supervised learning** — require extensive labeled anomaly data

Netti, Kiziltan, et al., “Online Fault Classification in HPC Systems Through Machine Learning”, 2019; Aksar et al., “E2EWatch: An End-to-End Anomaly Diagnosis Framework for Production HPC Systems”, 2021

Random Forest Architecture

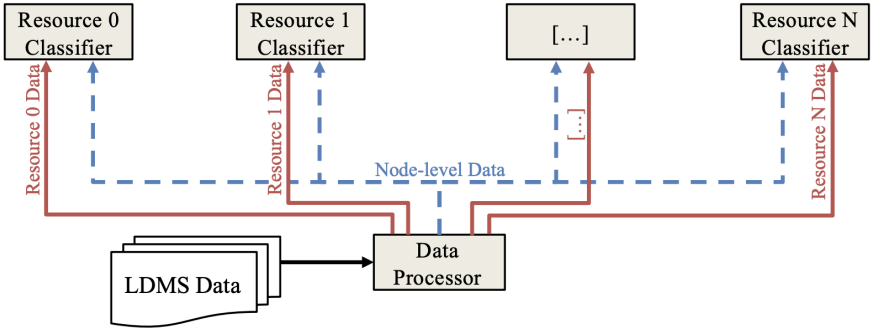


Image source: Netti et al. Online Fault Classification in HPC Systems Through Machine Learning. Euro-Par 2019. DOI: 10.1007/978-3-030-29400-7_1

Comparison: Traditional ML Methods

Feature	Random Forest	E2EWatch
Dataset	Antarex (ETH Zurich)	Eclipse (Sandia, 1,488 nodes)
Data Type	Time-series metrics	
Injection Method	Synthetic (FINJ tool / HPAS suite)	
Window Size	60s (10s step)	60s (15s step)
Features	Statistical features from different metrics	
Model	Random Forest	LGBM/XGBoost
F1-Score	0.98	0.91 (LGBM)

Limitations of Traditional ML

- **Supervised Learning Challenges**
  - Requires **labeled anomalies** for each class
  - May struggle with novel/unseen patterns
- **Limitation**
  - Use of synthetic data may not reflect real-world anomalies

Autoencoders

Autoencoders vs Variational Autoencoders

- **Autoencoder:**
  - Learns deterministic encoding-decoding mapping
  - Direct reconstruction of input dat
  - Fixed latent representation

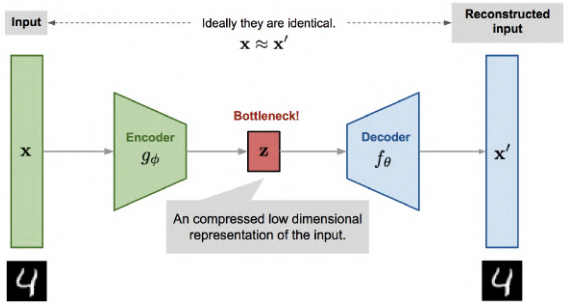


Image source: https://lilianweng.github.io/posts/2018-08-12-vae/

Autoencoders vs Variational Autoencoders

- **Autoencoder:**
  - Learns deterministic encoding-decoding mapping
  - Direct reconstruction of input dat
  - Fixed latent representation
- **Variational Autoencoder:**
  - Learns probabilistic latent space
  - Enables better generalization
  - Handles uncertainty in data

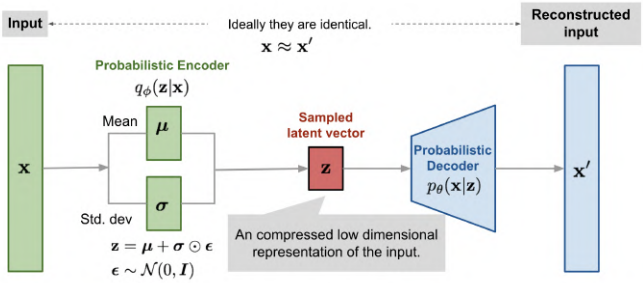


Image source: https://lilianweng.github.io/posts/2018-08-12-vae/

Comparison: Autoencoder Methods

Feature	Refine (VAE)	ExaMon-X (AE)
Dataset	Eclipse, Volta	Marconi, D.A.V.I.D.E.
Architecture	VAE	AE
Training	Unsupervised	Semi-supervised
Contamination	Up to 10%	Real anomalies
Key Innovation	Iterative filtering	Hybrid AE+Classifier
F1-Score	0.88 (10% contam.)	0.85

Sencan et al., “Refine: A Robust Approach to Unsupervised Anomaly Detection for Production HPC Systems”, 2025; Borghesi et al., “ExaMon-X: Anomaly Detection in Production HPC Systems”, 2021

Refine: Training Process

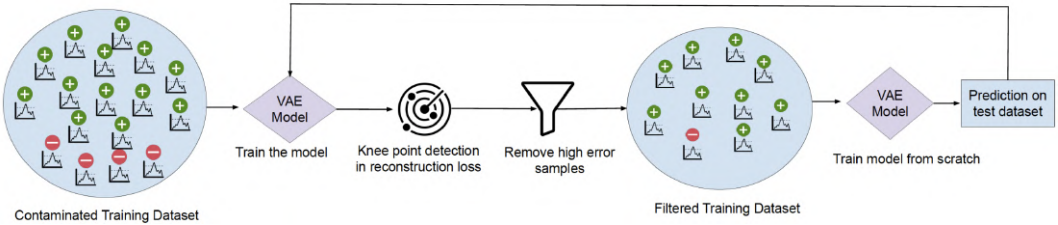


Image source: Sencan et al. Refine: A Robust Approach to Unsupervised Anomaly Detection for Production HPC Systems. ISC High Performance 2025. DOI: 10.23919/ISC.2025.11018307

Refine: Online Detection

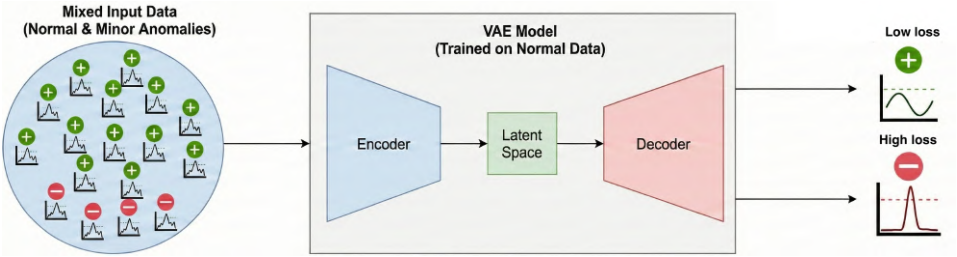


Image source: Sencan et al. Refine: A Robust Approach to Unsupervised Anomaly Detection for Production HPC Systems. ISC High Performance 2025. DOI: 10.23919/ISC.2025.11018307

Takeaways and Limitations

Key Takeaways

- Data contamination critical in production data
- VAE handles contaminated training sets
  - Achieves 100% accuracy on real production data
- Iterative filtering improves performance on noisy data

Limitations

- Performance decreases with higher contamination
- Stopping criteria needed (no ground truth)

Deep Learning

Deep Learning Approaches

Why Deep Learning?

Captures temporal dependencies—failures unfold over time

Methods:

- RNN-Based: Desh (logs), RUAD (sensors)
- Transformer-Based: NodeSentry (job-aware)

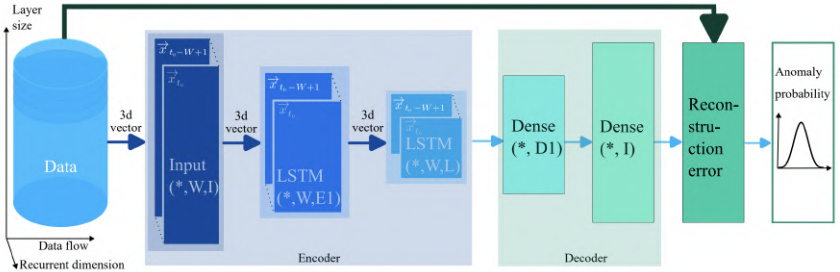
Desh: Log-Based Failure Prediction

- Architecture
  - Stacked LSTM (2 layers)
  - Skip-gram embeddings
- Method
  1. Learn failure chains
  2. Add time deltas (ΔT)
  3. Predict lead time & node (3 minutes lead time)
- Innovation
  - Predicts which node in how many minutes

Das, Mueller, and Rountree, “Desh: Deep Learning for System Health Prediction of Lead Times to Failure in HPC”, 2018

RUAD: Unsupervised Sensor Detection

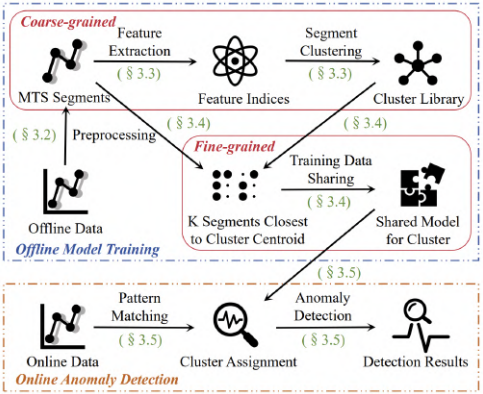
- How LSTM-AE Works
  - Sensor sequence → LSTM Encoder → Latent representation → Dense Decoder → Reconstruction
- LSTM vs Normal Encoder: LSTM maintains hidden state to capture temporal dependencies, unlike normal encoders that process each time step



Molan et al., “RUAD: Recurrent Unsupervised Anomaly Detection in High Performance Computing Systems”, 2023

NodeSentry: Overall Process

- Challenge
  - Normal behavior changes with every job transition
- Architecture
  - Two-stage: Clustering + Transformer MoE
- Method
  - Segment by jobs (Slurm)
  - Cluster patterns (coarse-grained)
  - MoE routes to experts (fine-grained)



Xia et al. DOI: 10.1145/3712285.3759794

Xia et al., “Effective Node-Level Anomaly Detection in HPC Systems via Coarse-Grained Clustering and Fine-Grained Model Sharing”, 2025

Comparison: Deep Learning Methods

Feature	Desh	RUAD	NodeSentry
Data	Logs	Sensors	Sensors
Dataset Source	Cray XC30	Marconi100	Production HPC (D1/D2)
Model	LSTM	LSTM-AE	Transformer MoE
Supervision	Supervised	Unsupervised	Semi-supervised
Performance	85% recall	AUC 0.767	F1 0.876–0.891
Anomalies		Real	

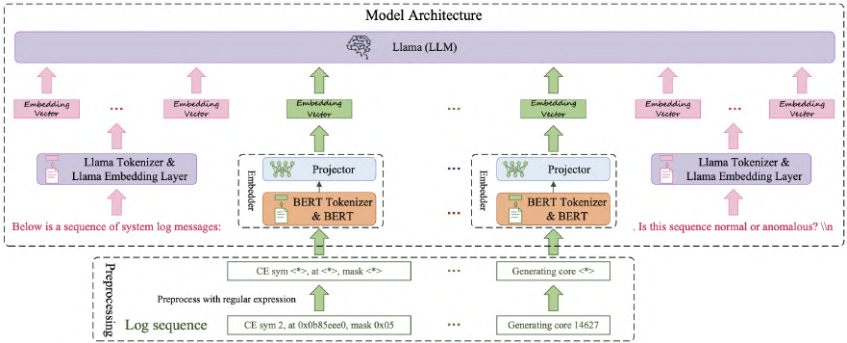
Takeaways and Limitations

- Takeaways
  - Temporal context essential for failure prediction
  - Unsupervised methods work when anomalies are rare
  - Job awareness critical for HPC systems
  - Use of real anomalies improves model reliability
- Limitations
  - Lead time vs. false positive trade-off
  - Pattern matching delay

LLMs

LogLLM: Log-based Anomaly Detection

- **Architecture:**
  - **BERT (encoder)** Extracts semantic vectors from logs
  - **Projector** Aligns BERT and Llama embedding spaces
  - **Llama (decoder)** Classifies log sequences



Guan et al., “LogLLM: Log-based Anomaly Detection Using Large Language Models”, 2024

LogLLM: Performance and Promise for HPC

- Performance
  - Average F1-score: 0.959
  - Outperforms NeuralLog (0.893)
  - Outperforms LogBERT (0.456)
  - Outperforms FastLogAD (0.341)
  - Outperforms RAPID (0.602)
- Promise for HPC
  - Semantic understanding adapts to software evolution without retraining
  - Higher context LLMs can receive huge logs as inputs

Guan et al., “LogLLM: Log-based Anomaly Detection Using Large Language Models”, 2024

Graph-Based

Graph-Based Approach: A Unique Perspective

- Unique Approach
  - Represents time series as graphs to capture topological patterns
- Method
  - Convert sensor time series $\rightarrow$ natural visibility graphs
  - Enrich with sensor type/ID $\rightarrow$ Graph2Vec embeddings
  - Train CatBoost to predict compute node downtimes
- Results
  - AUC ROC up to 0.8085 on Marconi 100 (16 nodes, 2020–2022)

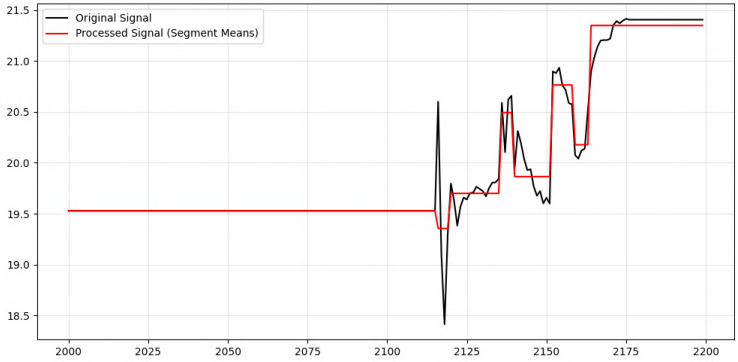
Rozanec, Luštrek, and Fortuna, “Learning Anomalies from Graph: Predicting Compute Node Failures on HPC Clusters”, 2025

Data Preprocessing

Time	Sensor_X
1	12.5
2	N/A
3	13.2
4	N/A

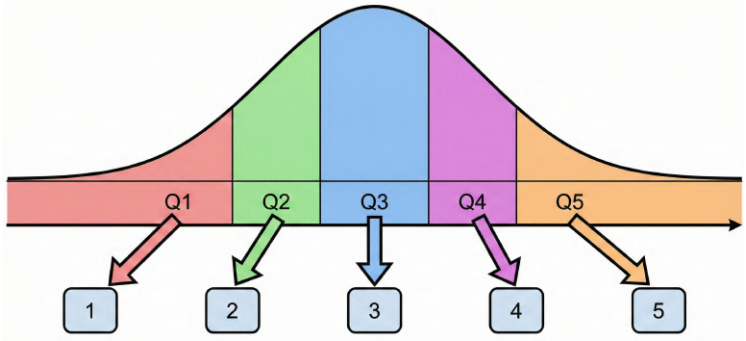
Step 1: Forward-Fill Missing Values

Data Preprocessing



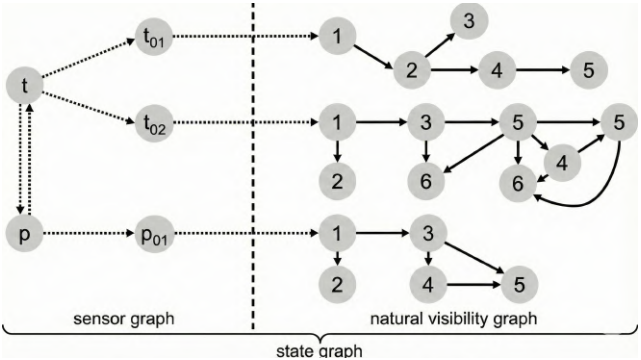
Step 2: Change Detection and Value Replacement

Data Preprocessing



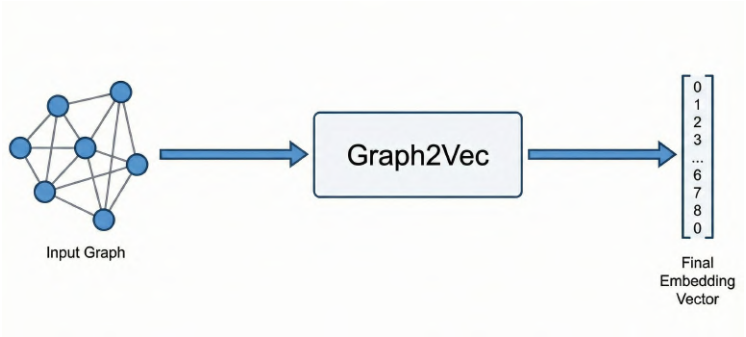
Step 3: Quantization into 5 Bins

Data Preprocessing



Step 4: Natural Visibility Graph Construction

Data Preprocessing



Step 5: Graph Enrichment and Embedding

Practical Implementation

Practical Implementation

- Experimentation
  - Reproduced graph-based algorithm on different node sets
  - Original study used small subset of complete Marconi 100 dataset
  - Tested on two distinct node sets to validate generalizability

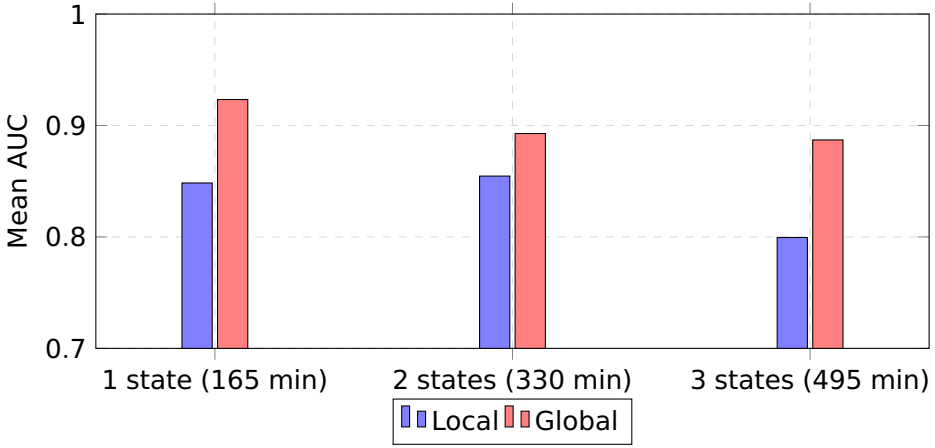
Selected Features

Sensor Features used in the implementation

Feature Name	Description
ambient_avg	Ambient temperature
dimm0_temp_avg	DIMM0 temperature
fan_disk_power_avg	Fan disk power
gpu0_core_temp_avg	GPU0 core temperature
gpu0_mem_temp_avg	GPU0 memory temperature
p0_io_power_avg	P0 IO power
p0_mem_power_avg	P0 memory power
p0_power_avg	P0 power
ps0_input_power_avg	PS0 input power
ps0_output_curre_avg	PS0 output current
ps0_output_volta_avg	PS0 output voltage
fan0_0_avg	Fan0_0 speed
p0_vdd_temp_avg	P0 VDD temperature

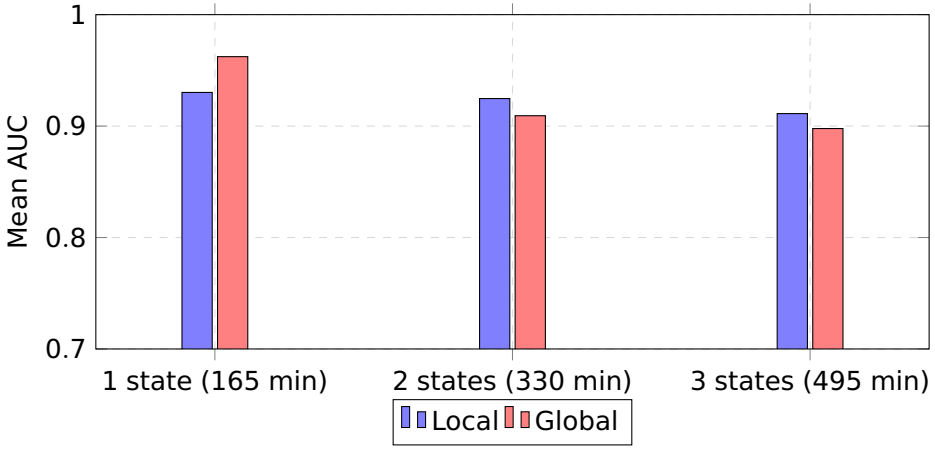
Node Set 1: Results

Nodes: 17 nodes (IDs: 20–29, 31–33, 35–38)



Node Set 2: Results

Nodes: 17 nodes (IDs: 40–59, excluding some)



Comparison

Method Comparison: Important Note

Fairness Disclaimer

This comparison is not completely fair due to:

- Different datasets used (various HPC systems, scales, time periods)
- Different anomaly types (synthetic vs. real, different failure modes)
- Different evaluation metrics and thresholds
- Different data types (logs vs. sensors)

Comprehensive Method Comparison

Method	Metric	Anomalies	Supervision	Data Type	Scalability
Random Forest	**F1 0.98**	Synthetic	Supervised	Statistical metrics	Global
E2EWatch	F1 0.91	Synthetic	Supervised	Statistical metrics	Global
LogLLM	F1 0.959	Real	Unsupervised	Textual logs	Global
NodeSentry	F1 0.876–0.891	Real	Semi-supervised	Time series	Per-node
Refine (VAE)	F1 0.88	Real	Unsupervised	Time series	Per-node
ExaMon-X	F1 0.85	Real	Semi-supervised	Time series	Per-node
RUAD	AUC 0.767	Real	Unsupervised	Time series	Per-node
Desh	Recall 85%	Real	Supervised	Textual logs	Per-node
Graph-Based	AUC 0.809	Real	Supervised	Embeddings	Global/ Per-Node

Conclusion

Conclusion

- Explored various ML approaches for HPC predictive maintenance
  - Traditional ML, Autoencoders, Deep Learning, LLM-based, Graph-based methods
- Key challenges identified
  - Label scarcity, data contamination, system scale
- State-of-the-art methods
  - LogLLM for log-based detection, VAE for sensor-based detection

What's Next?

- Comprehensive benchmarking for HPCs
  - Need for exclusive HPC benchmarking framework
  - Compare different methods using same settings
- Data labelling challenges
  - Data labelling has a long way to go
  - Need for better annotation tools and standards
- GWDG HPC system integration
  - Viability of extracting data from GWDG's HPC nodes
  - Building anomaly detection system on production data

References

Borghesi, Andrea et al. “ExaMon-X: Anomaly Detection in Production HPC Systems”. In: *IEEE Transactions on Industrial Informatics* (2021). DOI: 10.1109/TII.2021.3125885.

Guan, Wei et al. “LogLLM: Log-based Anomaly Detection Using Large Language Models”. In: *arXiv preprint arXiv:2411.08561* (2024). DOI: 10.48550/arXiv.2411.08561. URL: https://arxiv.org/abs/2411.08561.

International Energy Agency. *AI is Set to Drive Surging Electricity Demand from Data Centres*. 2024. URL: https://www.iea.org/news/ai-is-set-to-drive-surging-electricity-demand-from-data-centres-while-offering-the-potential-to-transform-how-the-energy-sector-works (visited on 12/29/2024).

Molan, Matteo et al. “RUAD: Recurrent Unsupervised Anomaly Detection in High Performance Computing Systems”. In: *Future Generation Computer Systems* 141 (2023), pp. 1–15. DOI: 10.1016/j.future.2022.12.001.

Netti, Alessio, Matthias Müller, et al. “From Facility to Application Sensor Data: Modular, Continuous and Holistic Monitoring with DCDB”. In: *ACM Transactions on Parallel Computing* 8.2 (2021). Operational Data Analytics (ODA) framework: monitoring, ML applications, combination layer.

References (continued)

Aksar, Buse et al. “E2EWatch: An End-to-End Anomaly Diagnosis Framework for Production HPC Systems”. In: *Euro-Par 2021: Parallel Processing*. Springer, 2021, pp. 61–76. DOI: 10.1007/978-3-030-85665-6_5.

Das, Anwesha, Frank Mueller, and Barry Rountree. “Desh: Deep Learning for System Health Prediction of Lead Times to Failure in HPC”. In: *Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing*. ACM, 2018, pp. 40–51. DOI: 10.1145/3208040.3208051.

Netti, Alessio, Zeynep Kiziltan, et al. “Online Fault Classification in HPC Systems Through Machine Learning”. In: *Euro-Par 2019: Parallel Processing*. Ed. by Ramin Yahyapour. Vol. 11725. Lecture Notes in Computer Science. Cham: Springer, 2019, pp. 3–14. DOI: 10.1007/978-3-030-29400-7_1. URL: https://doi.org/10.1007/978-3-030-29400-7_1.

Rozanec, Jože M., Mitja Luštrek, and Blaž Fortuna. “Learning Anomalies from Graph: Predicting Compute Node Failures on HPC Clusters”. In: *Proceedings of the Northern Lights Deep Learning Conference*. OpenReview ID: SPRdfOkuHw. 2025.

Sencan, Efe et al. “Refine: A Robust Approach to Unsupervised Anomaly Detection for Production HPC Systems”. In: *ISC High Performance 2025 Research Paper Proceedings (40th International Conference)*. 2025, pp. 1–12. DOI: 10.23919/ISC.2025.11018307.

Xia, Sibo et al. “Effective Node-Level Anomaly Detection in HPC Systems via Coarse-Grained Clustering and Fine-Grained Model Sharing”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC ’25. Association for Computing Machinery, 2025, pp. 1180–1194. ISBN: 9798400714665. DOI: 10.1145/3712285.3759794. URL: https://doi.org/10.1145/3712285.3759794.