The Journal of Open Source Software

NØMADE: Lightweight HPC Monitoring with Machine Learning-Based Failure Prediction

João Filipe Riva Tonini [1¶]

1 Academic Research Computing, University of Richmond, VA 23173, USA ¶ Corresponding author

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

NØMADE (NOde MAnagement DEvice) is a lightweight monitoring and predictive analytics tool for High-Performance Computing (HPC) clusters. It collects system metrics from SLURM-managed environments, stores time-series data in SQLite, and employs machine learning to predict job failures before they occur. The tool provides a real-time web dashboard and supports alerts via email, Slack, or webhooks. NØMADE requires no external databases or complex infrastructure—only Python and standard system tools.

A key innovation is the application of biogeographical network analysis concepts to HPC monitoring. Inspired by methods for identifying transition zones between bioregions (Vilhena & Antonelli, 2015), NØMADE treats HPC resource domains (compute, storage, network) as interconnected regions where failures cluster at domain boundaries—such as transitions between local scratch and network-attached storage (NAS), or between CPU and GPU workloads. This enables failure pattern recognition that emerges from the data rather than from predefined rules.

Statement of Need

HPC administrators face a persistent challenge: detecting job failures before they impact researchers. Enterprise monitoring solutions like Prometheus, Grafana, or Nagios require significant infrastructure and target general IT systems rather than HPC-specific workloads. Existing HPC tools such as TACC Stats (Evans et al., 2014), XDMoD (Palmer et al., 2015), and LLNL's Lightweight Distributed Metric Service (Agelastos et al., 2014) provide detailed metrics but require substantial deployment effort and focus on post-hoc analysis rather than real-time prediction.

Common HPC failure patterns include:

- **NFS saturation**: Jobs writing to network storage instead of local scratch
- **Memory leaks**: Gradual consumption leading to out-of-memory kills
- **GPU thermal throttling**: Temperature-induced performance degradation
- **Queue starvation**: Resource contention causing excessive wait times

These failures often exhibit warning signs minutes to hours before critical thresholds are breached. NØMADE addresses this gap by providing:

- **Zero-infrastructure deployment**: Single SQLite database, no external services
- **Real-time prediction**: ML ensemble identifies high-risk jobs before failure
- **Predictive alerts**: Derivative analysis detects accelerating resource consumption
- **Domain-aware analysis**: Recognizes HPC-specific failure patterns at resource boundaries

The tool is suited for small-to-medium HPC centers, research groups managing clusters, or as a complement to existing monitoring infrastructure.

Implementation

NØMADE is implemented in Python and follows a modular architecture (Figure 1):

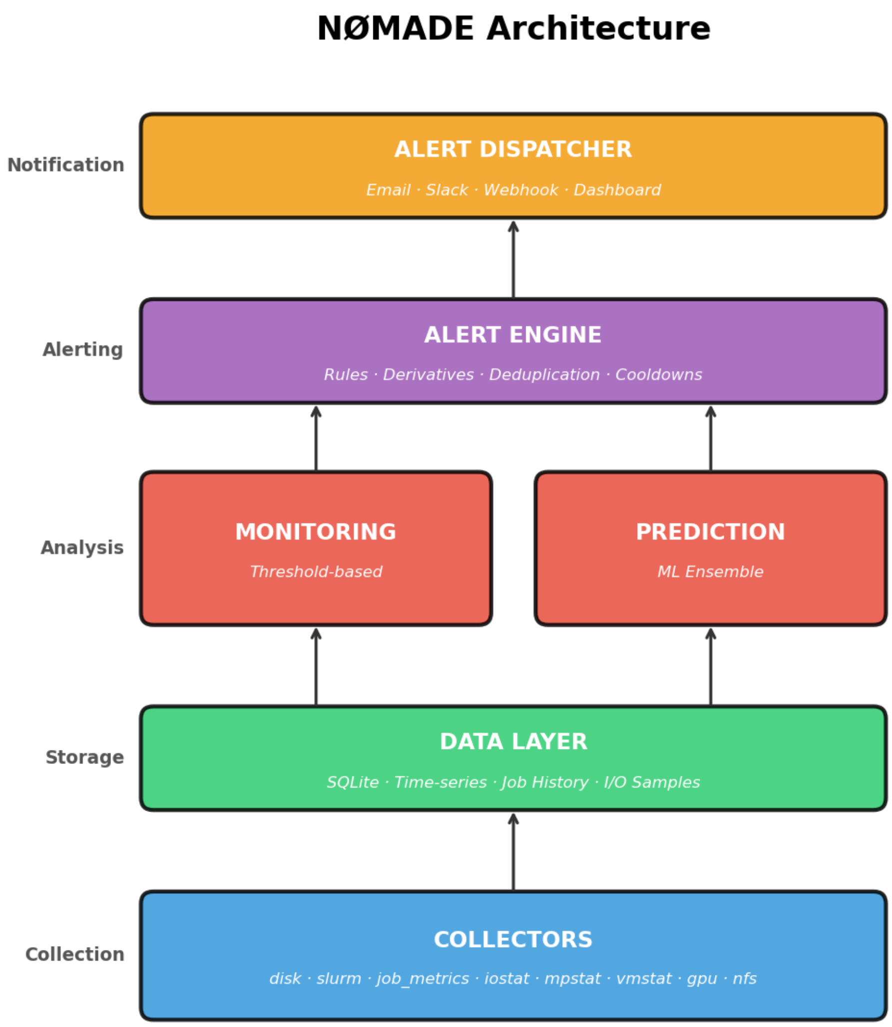

Figure 1: NØMADE architecture showing the data flow from collectors through the prediction engine to alert dispatch.

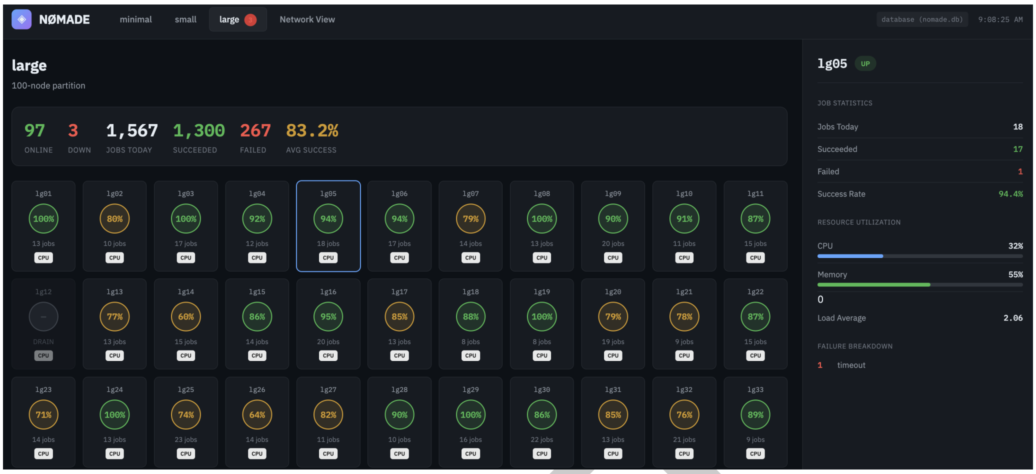

Figure 2: NØMADE dashboard showing cluster health with per-node job statistics, CPU utilization rings, and failure breakdown.

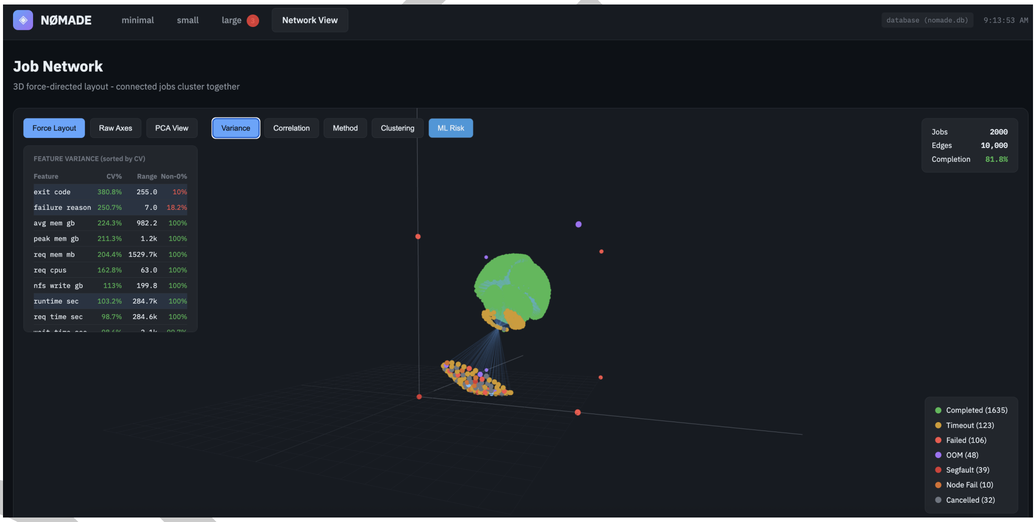

Figure 3: Network visualization showing jobs clustered by feature similarity. Failed jobs (red/orange) cluster separately from successful jobs (green), enabling pattern-based failure prediction.

Collectors gather metrics from system tools (`iostat`, `vmstat`, `nvidia-smi`), SLURM commands (`sacct`, `squeue`, `sinfo`), and per-job I/O statistics from `/proc/[pid]/io`. A SLURM prolog hook captures job context at submission time.

Feature Engineering transforms raw metrics into a 17-dimensional feature vector per job, including CPU and memory efficiency from `sacct`, NFS write ratios from the job monitor, and system-level indicators (I/O wait, memory pressure, swap activity). These features enable similarity-based analysis across jobs.

ML Prediction uses an ensemble of three models:

- Graph Neural Network (GNN): Captures relationships between similar jobs based on Simpson similarity of feature vectors
- LSTM: Detects temporal patterns and early warning trajectories
- Autoencoder: Identifies anomalous jobs that deviate from normal behavior

The ensemble outputs a continuous risk score (0–1) rather than binary classification, providing nuanced assessment of job health.

Alert System supports both threshold-based alerts (disk usage, GPU temperature) and predictive alerts using derivative analysis. When the rate of change indicates a threshold will be breached, alerts fire before the actual breach occurs. Notifications route through email, Slack, or webhooks with configurable cooldowns to prevent alert fatigue.

Usage

NØMADE installs via pip and initializes with two commands:

```
pip install nomade-hpc
nomade init
nomade collect     # Start data collection
nomade dashboard   # Launch web interface
```

For HPC-wide deployment, system installation configures systemd services and SLURM prolog hooks:

```
sudo nomade init --system
sudo systemctl enable --now nomade nomade-learn
```

The dashboard displays real-time cluster health, job risk assessments, and historical trends. Configuration uses TOML files supporting custom thresholds, alert destinations, and collection intervals.

Acknowledgements

The author thanks George Flanagin for advice and inspiration on HPC system administration, and the University of Richmond Research Computing group for cluster access and testing.

References

Agelastos, A., Allan, B., Brandt, J., Cassella, P., Enos, J., Fullop, J., Gentile, A., Monk, S., Naksinehaboon, N., Ogden, J., Rajan, M., Showerman, M., Stevenson, J., Taber, N., & Tucker, T. (2014). The lightweight distributed metric service: A scalable infrastructure for continuous monitoring of large scale computing systems and applications. *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 154–165. https://doi.org/10.1109/SC.2014.18

Evans, R. T., Barth, W. L., Browne, J. C., DeLeon, R. L., Furlani, T. R., Gallo, S. M., Jones, M. D., & Patra, A. K. (2014). TACC Stats: Analysis of HPC system resource usage. *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, 1–8. https://doi.org/10.1145/2616498.2616533

Palmer, J. T., Gallo, S. M., Furlani, T. R., Jones, M. D., DeLeon, R. L., White, J. P., Simakov, N., Patra, A. K., Sperhac, J., Yearke, T., Rathsam, R., Inber, M., Cornelius, C. D., & Browne, J. C. (2015). XDMoD: A tool for the comprehensive management of high-performance computing resources. *Computing in Science & Engineering*, *17*(2), 52–62. https://doi.org/10.1109/MCSE.2015.32

Vilhena, D. A., & Antonelli, A. (2015). A network approach for identifying and delimiting biogeographical regions. *Nature Communications*, *6*(1), 6848. https://doi.org/10.1038/ncomms7848