Software paper for submission to the Journal of Open Research Software

(1) Overview

Title

NØMAD: Lightweight HPC Monitoring and Diagnostics with Machine Learning-Based Failure Prediction

Paper Authors

1. Tonini, João Filipe Riva (corresponding author)

Paper Author Roles and Affiliations

1. Academic Research Computing, University of Richmond, Richmond, VA 23173, USA.
   Email: jtonini@richmond.edu
   ORCID: 0000-0002-4730-3805

Abstract

NØMAD (NOde Monitoring And Diagnostics) is a lightweight monitoring and predictive analytics tool designed for computing infrastructure that requires minimal deployment overhead. At its core, metric collectors scheduled via systemd timers gather system metrics—disk, CPU, memory, I/O, and GPU utilization—from standard Linux tools, storing everything in a single SQLite database. When SLURM is available, these collectors extend to capture job-level analytics from scheduler commands and per-process I/O statistics, enabling deeper insight into workload behavior. The system employs machine learning models to predict job failures before they occur, while a data readiness estimator helps administrators determine when sufficient historical data has been collected for reliable predictions. Beyond prediction, diagnostic tools provide targeted analysis of network performance, storage health, and node-level bottlenecks. A key innovation is modeling jobs as nodes in a similarity network, where edges connect jobs with comparable resource usage patterns; because jobs with similar characteristics tend to share similar outcomes, the network structure reveals failure-prone regions in the feature space. The tool includes a real-time web dashboard for visualization and supports alerts via email, Slack, or webhooks. By requiring no external databases or complex infrastructure, NØMAD is particularly well-suited for small-to-medium HPC centers and research groups seeking sophisticated monitoring without enterprise-scale overhead.

Keywords

HPC; high-performance computing; monitoring; machine learning; SLURM; predictive analytics; failure prediction

Introduction

HPC administrators face a persistent challenge: detecting job failures before they impact researchers. While traditional monitoring tools offer valuable capabilities, they have evolved along two separate paths—enterprise IT solutions and HPC-specific tools—neither of which fully addresses this need.

Enterprise monitoring solutions like Prometheus [2], Grafana [3], and Nagios [4] provide mature, feature-rich platforms but require significant infrastructure—database servers, message queues, and dedicated hardware—that may be impractical for small-to-medium HPC centers. These tools target general IT workloads and lack native understanding of HPC concepts such as job schedulers, batch queues, and compute allocations. HPC-specific tools address this domain gap but introduce their own complexity. TACC Stats [5], for example, provides comprehensive hardware counter data and correlates system metrics with job performance, enabling detailed post-hoc analysis of application behavior. At the institutional scale, XDMoD [6] offers sophisticated job accounting and resource utilization reporting, though its deployment requires multiple database backends and web services. For large-scale continuous monitoring, the Lightweight Distributed Metric Service (LDMS) [7] achieves impressive scalability through a distributed architecture. What these tools share, despite their different approaches, is an emphasis on retrospective analysis—understanding what happened after a failure—rather than real-time prediction of failures before they occur.

A complementary approach treats failure prediction as a pattern recognition problem. Jobs that fail often share common characteristics: excessive NFS writes, memory pressure, or inefficient CPU utilization. Machine learning methods have been applied to HPC failure prediction [8], typically using classification models trained on job features. However, these approaches often treat jobs as independent observations, ignoring the structural relationships between jobs with similar resource profiles.

NØMAD addresses these gaps by combining zero-infrastructure deployment with network-based failure prediction. The key insight is that jobs with similar resource usage patterns tend to experience similar outcomes. Rather than treating each job independently, NØMAD models jobs as nodes in a similarity network where edges connect jobs with comparable behavioral fingerprints. This network structure reveals failure-prone regions in the feature space—clusters of jobs that share characteristics associated with poor outcomes—enabling both risk assessment for individual jobs and actionable recommendations for users.

The choice of similarity metric is consequential for network topology. NØMAD uses cosine similarity on Z-score normalized feature vectors, which emphasizes the *shape* of resource usage profiles rather than absolute magnitudes. A job requesting 64GB of memory with 50% utilization has a similar profile to one requesting 8GB with 50% utilization—both represent reasonable memory sizing—even though their absolute consumption differs by an order of magnitude. This approach draws inspiration from network-based methods in biogeography, where similarity metrics identify emergent regions from species distribution patterns [9]. Just as biogeographical networks reveal ecological boundaries from observational data without predefined regions, job similarity networks reveal behavioral boundaries between successful and failing workloads without predefined failure categories. The similarity threshold (default ≥ 0.7) controls network density: higher thresholds create sparser networks with tighter clusters, while lower thresholds reveal broader patterns at the cost of specificity. In practice, NØMAD achieves zero-infrastructure deployment through a single SQLite database with no external services required. The system provides system-level monitoring that works on any Linux machine without requiring HPC software, while optional SLURM integration enables job-level analytics and predictive alerts. The similarity network analysis learns failure patterns directly from historical data, adapting to site-specific workload characteristics. Beyond operational monitoring, an educational analytics module tracks the development of computational proficiency over time, enabling instructors to generate insights such as “15 of 20 students improved memory efficiency over the semester.”

Implementation and Architecture

NØMAD is implemented in Python and follows a modular architecture with seven main components (Figure 1):

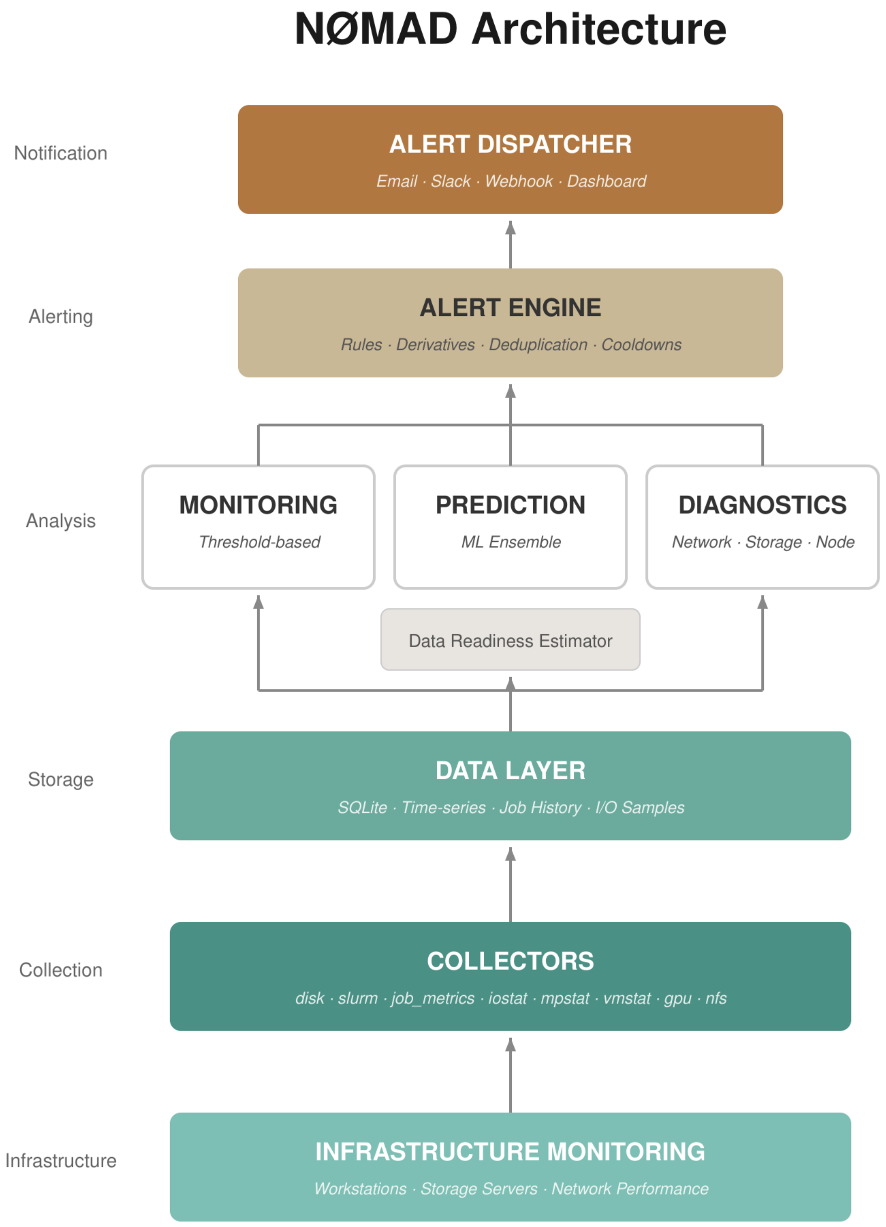

NØMAD Architecture
Notification
ALERT DISPATCHER
Email · Slack · Webhook · Dashboard
Alerting
ALERT ENGINE
Rules · Derivatives · Deduplication · Cooldowns
Analysis
MONITORING
Threshold-based
PREDICTION
ML Ensemble
DIAGNOSTICS
Network · Storage · Node
Data Readiness Estimator
Storage
DATA LAYER
SQLite · Time-series · Job History · I/O Samples
Collection
COLLECTORS
disk · slurm · job_metrics · iostat · mpstat · vmstat · gpu · nfs
Infrastructure
INFRASTRUCTURE MONITORING
Workstations · Storage Servers · Network Performance

Figure 1: NØMAD system architecture showing the data flow from collectors through the analysis engines to the alert dispatcher and web dashboard. The monitoring engine handles threshold-based alerts while the prediction engine uses ML models for proactive failure detection.

Collectors form the foundation of the system, gathering metrics via systemd timers at configurable intervals (default: 60 seconds). They pull data from standard Linux tools—`iostat`, `vmstat`, `nvidia-smi`, and `nfsiostat`—along with filesystem utilities. When SLURM is available, additional collectors capture queue state, job history, and per-job I/O statistics from

`/proc/[pid]/io`. Importantly, collectors degrade gracefully when their requirements are not met: a system without GPUs simply skips GPU monitoring rather than failing.

Raw metrics then flow through **Feature Engineering**, which transforms them into a standardized feature vector for each job. System-level features capture I/O wait, memory pressure, swap activity, and device utilization. With SLURM integration, job-specific features—CPU and memory efficiency, NFS write ratios, runtime characteristics—extend the vector to 17 dimensions, enabling meaningful similarity comparisons across jobs.

These features feed into the **ML Prediction** engine, which employs an ensemble of three complementary models: a Graph Neural Network (GNN) that captures relationships between similar jobs, an LSTM that detects temporal patterns and early warning trajectories, and an Autoencoder that identifies anomalous jobs deviating from learned normal behavior. Rather than producing binary classifications, the ensemble outputs a continuous risk score between 0 and 1, allowing administrators to set thresholds based on their operational needs.

The **Alert System** acts on both threshold-based triggers (such as disk usage or GPU temperature) and predictive signals from the ML engine. A key capability is derivative analysis: when the rate of change indicates an impending threshold breach, alerts fire proactively rather than reactively. Notifications route through email, Slack, or webhooks, with configurable cooldowns to prevent alert fatigue.

Before ML models can provide reliable predictions, sufficient training data must be collected. The **Data Readiness Estimator** addresses this bootstrap problem by assessing sample size adequacy, class balance between successful and failed jobs, feature coverage and variance, and data recency. Based on current collection rates, it forecasts time-to-readiness—for example, "At 125 jobs/day, recommended threshold will be reached in approximately 3 days." The `nomad readiness` command produces detailed reports with actionable recommendations.

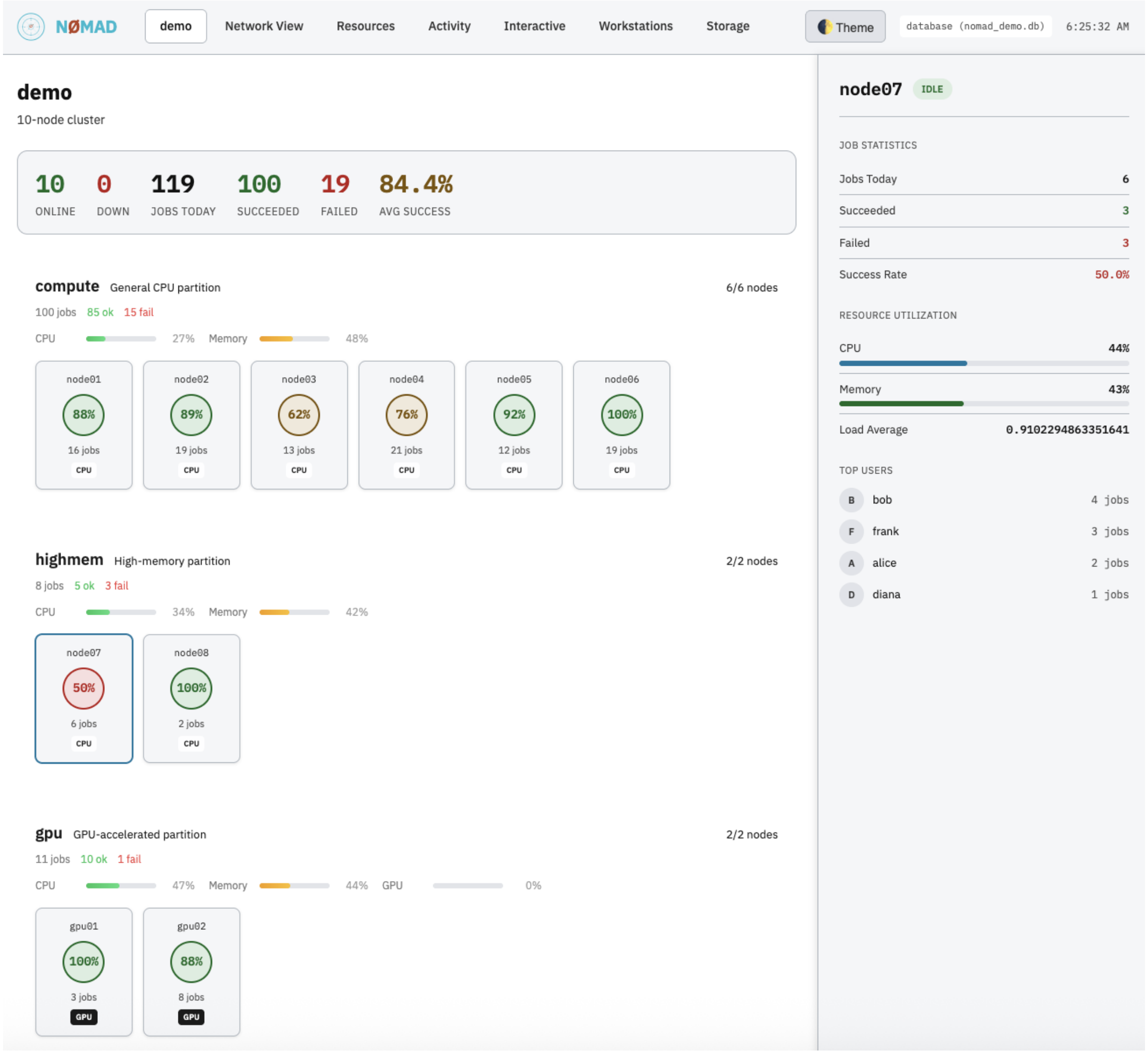
For troubleshooting, the **Diagnostics Module** provides targeted analysis through `nomad diag` commands that examine network performance between nodes, storage health and I/O patterns, or node-level resource bottlenecks. Each diagnostic produces both summary statistics and detailed breakdowns, accelerating root cause identification.

Finally, **Infrastructure Monitoring** extends visibility beyond compute nodes to encompass the broader research environment. The dashboard includes dedicated views for departmental workstations—tracking CPU, memory, disk usage, and logged-in users—as well as storage servers with ZFS pool health, NFS client connections, and I/O metrics. This holistic perspective enables administrators to correlate job failures with infrastructure-wide issues such as NFS server congestion or storage capacity constraints.

Web Dashboard

NØMAD provides a real-time web dashboard for monitoring cluster health and job status (Figure 2). The dashboard displays partition-specific metrics including queue depth, node utilization, and resource consumption patterns.

The dashboard provides additional analytics views for resource tracking and infrastructure monitoring. The usage analytics view (Figure 3) shows CPU and GPU usage by group and user, plus job submission patterns as a heatmap. Interactive computing sessions (Figure 4) display active RStudio and Jupyter sessions. The infrastructure monitoring view (Figure 5) extends visibility to departmental workstations and storage servers.



demo

10-node cluster

10	0	119	100	19	84.4%
ONLINE	DOWN	JOBS TODAY	SUCCEEDED	FAILED	AVG SUCCESS

compute General CPU partition 6/6 nodes

100 jobs 85 ok 15 fail

CPU 27% Memory 48%

node01	node02	node03	node04	node05	node06
88%	89%	62%	76%	92%	100%
16 jobs	19 jobs	13 jobs	21 jobs	12 jobs	19 jobs
CPU	CPU	CPU	CPU	CPU	CPU

highmem High-memory partition 2/2 nodes

8 jobs 5 ok 3 fail

CPU 34% Memory 42%

node07	node08
50%	100%
6 jobs	2 jobs
CPU	CPU

gpu GPU-accelerated partition 2/2 nodes

11 jobs 10 ok 1 fail

CPU 47% Memory 44% GPU 0%

gpu01	gpu02
100%	88%
3 jobs	8 jobs
GPU	GPU

node07 IDLE

JOB STATISTICS

Jobs Today	6
Succeeded	3
Failed	3
Success Rate	50.0%

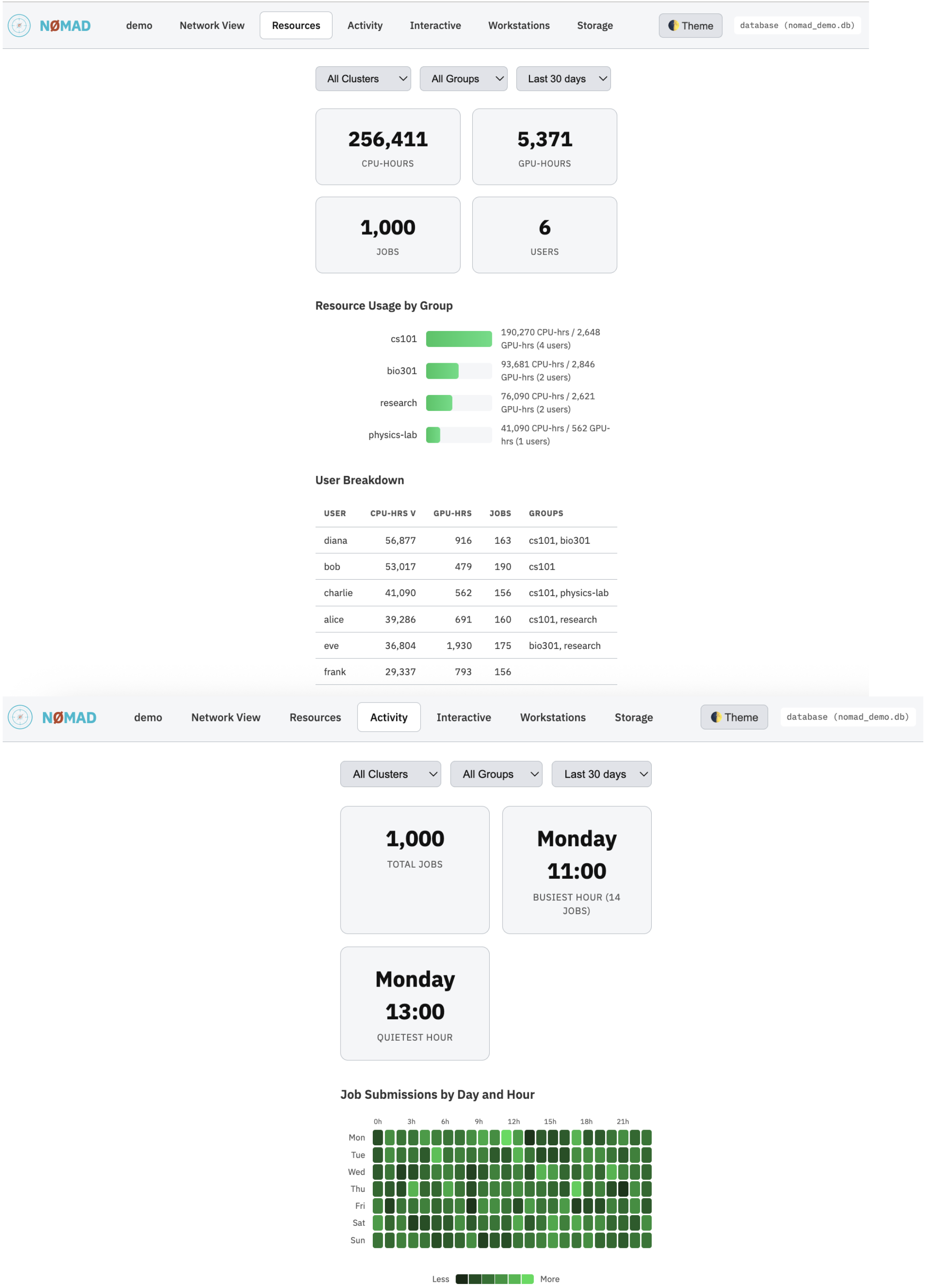
RESOURCE UTILIZATION

CPU	44%
Memory	43%
Load Average	0.9102294863351641

TOP USERS

B	bob	4 jobs
F	frank	3 jobs
A	alice	2 jobs
D	diana	1 jobs

Figure 2: NØMAD web dashboard in light theme showing real-time cluster status. The main view displays three partitions (compute, highmem, gpu) with node health rings indicating CPU utilization. The sidebar shows detailed statistics for the selected node including job counts, resource utilization, and top users.



256,411
CPU-HOURS

5,371
GPU-HOURS

1,000
JOBS

6
USERS

Resource Usage by Group

cs101	190,270 CPU-hrs / 2,648 GPU-hrs (4 users)
bio301	93,681 CPU-hrs / 2,846 GPU-hrs (2 users)
research	76,090 CPU-hrs / 2,621 GPU-hrs (2 users)
physics-lab	41,090 CPU-hrs / 562 GPU-hrs (1 users)

User Breakdown

USER	CPU-HRS ∨	GPU-HRS	JOBS	GROUPS
diana	56,877	916	163	cs101, bio301
bob	53,017	479	190	cs101
charlie	41,090	562	156	cs101, physics-lab
alice	39,286	691	160	cs101, research
eve	36,804	1,930	175	bio301, research
frank	29,337	793	156	

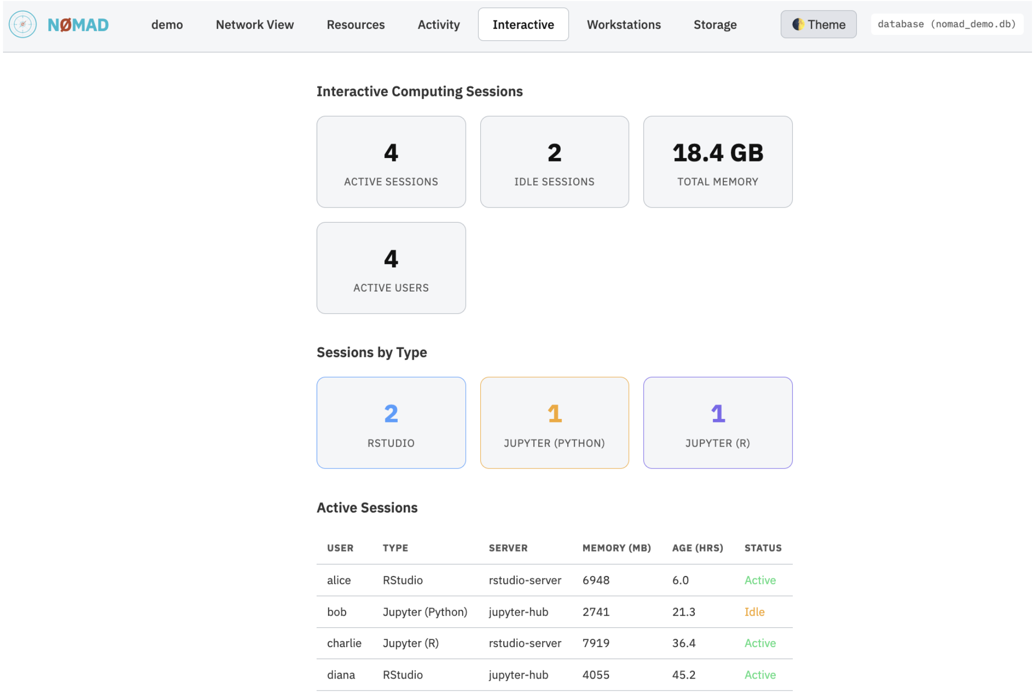
1,000
TOTAL JOBS

Monday 11:00
BUSIEST HOUR (14 JOBS)

Monday 13:00
QUIETEST HOUR

Job Submissions by Day and Hour

Figure 3: Usage analytics views. **Top**: Resource usage summary showing CPU-hours, GPU-hours, and breakdown by group and user. **Bottom**: Activity heatmap displaying job submissions by day and hour, revealing usage patterns.



Interactive Computing Sessions

4 ACTIVE SESSIONS

2 IDLE SESSIONS

18.4 GB TOTAL MEMORY

4 ACTIVE USERS

Sessions by Type

2 RSTUDIO

1 JUPYTER (PYTHON)

1 JUPYTER (R)

Active Sessions

USER	TYPE	SERVER	MEMORY (MB)	AGE (HRS)	STATUS
alice	RStudio	rstudio-server	6948	6.0	Active
bob	Jupyter (Python)	jupyter-hub	2741	21.3	Idle
charlie	Jupyter (R)	rstudio-server	7919	36.4	Active
diana	RStudio	jupyter-hub	4055	45.2	Active

Figure 4: Interactive computing sessions dashboard showing active RStudio and Jupyter sessions with memory usage, session age, and status.

Job Similarity Network

A distinguishing feature of NØMAD is the job similarity network visualization (Figure 6). Jobs are represented as nodes in a 3D space defined by their I/O characteristics: NFS write ratio, local write volume, and I/O wait percentage. When full job metrics are available, edges connect jobs with cosine similarity ≥ 0.7 based on their feature vectors.

The feature variance panel helps administrators verify normal cluster operation and identify which metrics contribute most to failure prediction.

Quality Control

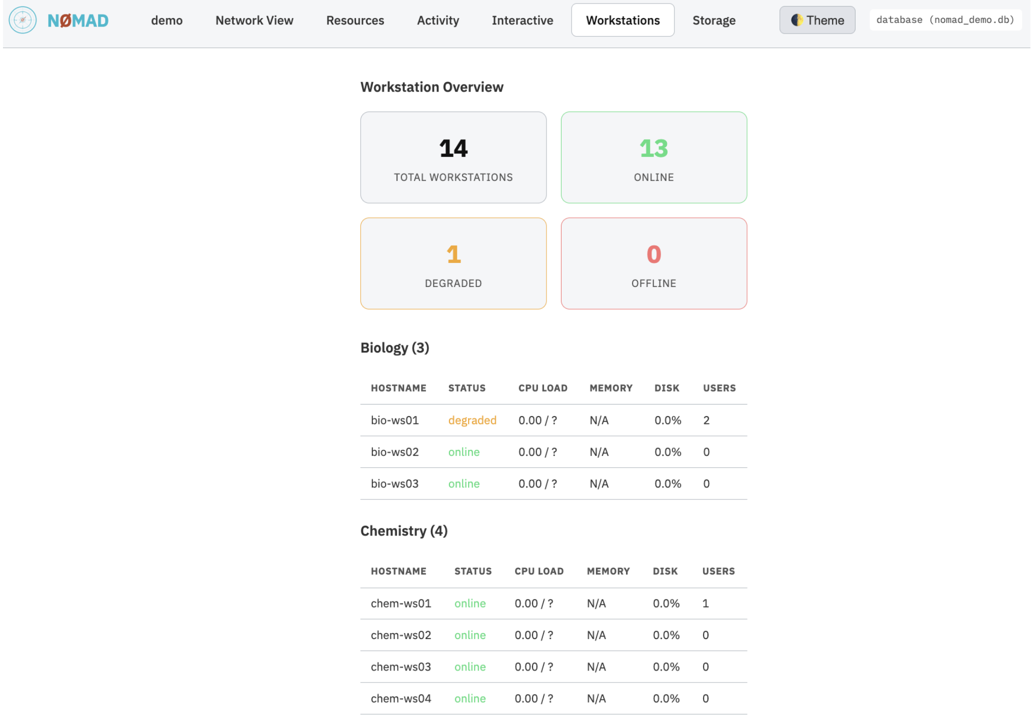
NØMAD includes a comprehensive test suite that ensures reliability across all components. Unit tests cover collectors, feature engineering, and ML components individually, while integration tests verify the full data pipeline from metric collection through prediction. For evaluation without a production cluster, a demo mode (`nomad demo`) generates synthetic HPC data with realistic patterns. Continuous integration via GitHub Actions runs the test suite on every commit.

The software has been tested on clusters ranging from 6 to 200 nodes, including both CPU-only and GPU-enabled partitions. The codebase follows PEP 8 style guidelines and includes type hints for improved maintainability.

Educational Analytics

NØMAD includes an educational analytics module (`nomad edu`) designed for classroom instruction, research mentorship, and HPC training programs. The module tracks the development of computational proficiency over time by analyzing per-job behavioral fingerprints across five dimensions: CPU efficiency, memory sizing, time estimation, I/O awareness, and GPU utilization.

The **Job Explanation** feature (`nomad edu explain`) translates raw job data into plain-language



Workstation Overview

14
TOTAL WORKSTATIONS

13
ONLINE

1
DEGRADED

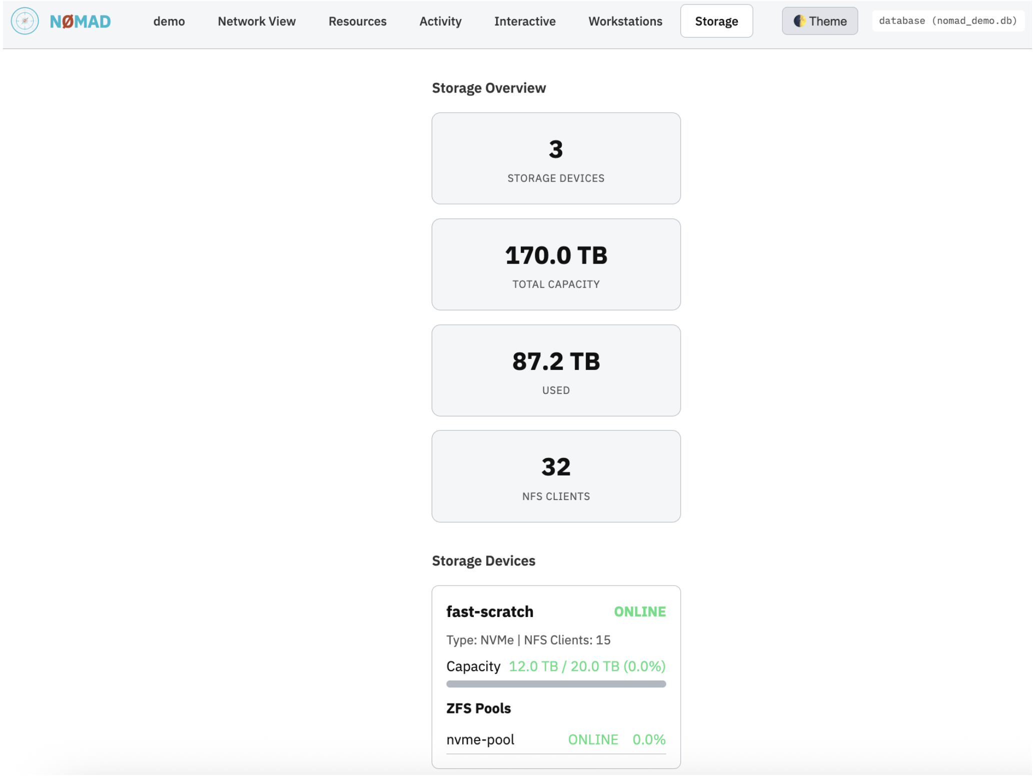
0
OFFLINE

Biology (3)

HOSTNAME	STATUS	CPU LOAD	MEMORY	DISK	USERS
bio-ws01	degraded	0.00 / ?	N/A	0.0%	2
bio-ws02	online	0.00 / ?	N/A	0.0%	0
bio-ws03	online	0.00 / ?	N/A	0.0%	0

Chemistry (4)

HOSTNAME	STATUS	CPU LOAD	MEMORY	DISK	USERS
chem-ws01	online	0.00 / ?	N/A	0.0%	1
chem-ws02	online	0.00 / ?	N/A	0.0%	0
chem-ws03	online	0.00 / ?	N/A	0.0%	0
chem-ws04	online	0.00 / ?	N/A	0.0%	0



Storage Overview

3
STORAGE DEVICES

170.0 TB
TOTAL CAPACITY

87.2 TB
USED

32
NFS CLIENTS

Storage Devices

fast-scratch ONLINE

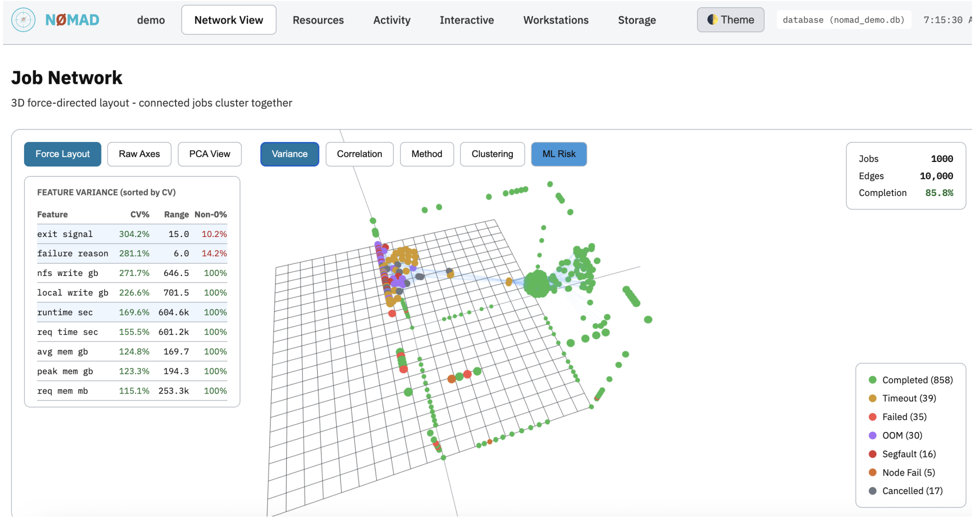
Type: NVMe | NFS Clients: 15

Capacity 12.0 TB / 20.0 TB (0.0%)

ZFS Pools

nvme-pool ONLINE 0.0%

Figure 5: Infrastructure monitoring views. **Top**: Workstation overview showing departmental machines (Biology, Chemistry, Physics, Math/CS) with status, CPU load, memory, disk usage, and logged-in users. **Bottom**: Storage overview displaying NFS servers with capacity, usage, ZFS pool health, and connected clients.



Job Network

3D force-directed layout - connected jobs cluster together

FEATURE VARIANCE (sorted by CV)

Feature	CV%	Range	Non-0%
exit signal	304.2%	15.0	10.2%
failure reason	281.1%	6.0	14.2%
nfs write gb	271.7%	646.5	100%
local write gb	226.6%	701.5	100%
runtime sec	169.6%	604.6k	100%
req time sec	155.5%	601.2k	100%
avg mem gb	124.8%	169.7	100%
peak mem gb	123.3%	194.3	100%
req mem mb	115.1%	253.3k	100%

Jobs	1000
Edges	10,000
Completion	85.8%

Figure 6: Job similarity network visualization with feature variance analysis. The 3D view shows jobs colored by outcome (green=completed, red=failed, orange=timeout, purple=OOM). The left panel displays feature variance statistics sorted by coefficient of variation (CV); high CV values for exit_signal and failure_reason (exceeding 200%) reflect normal cluster operation where most jobs succeed. The legend shows job outcome distribution across 1000 jobs.

feedback that students can immediately act upon. For each completed job, the system provides dimension-specific scores (0–100), proficiency levels (Excellent, Good, Developing, Needs Work), and actionable recommendations. For example, a job with 21% CPU utilization receives the feedback: “Very low CPU utilization—requested 4 cores but used ∼1. Try: `#SBATCH -ntasks=1`.”

Beyond individual job feedback, **User Trajectories** (`nomad edu trajectory`) track each person’s improvement across their job submissions. The system computes sliding-window averages to identify trends—improving, stable, or declining—for each proficiency dimension, enabling mentors to provide targeted guidance where it is most needed.

At the class level, **Group Reports** (`nomad edu report`) aggregate proficiency data across course sections or research groups. Instructors can generate summaries such as “15 of 20 students improved memory efficiency over the semester” or identify that a majority of students struggle with I/O patterns, suggesting curriculum adjustments. These features address a common challenge in HPC education: students submit jobs that technically complete but exhibit significant resource waste or suboptimal patterns. Traditional approaches require manual inspection of `sacct` output, which is impractical for classes with hundreds of students. NØMAD automates this assessment while providing pedagogically useful feedback that helps students develop genuine computational proficiency rather than simply learning to avoid errors.

Proficiency scores are persisted to the database, enabling longitudinal studies of HPC training effectiveness and research into factors that accelerate skill development.

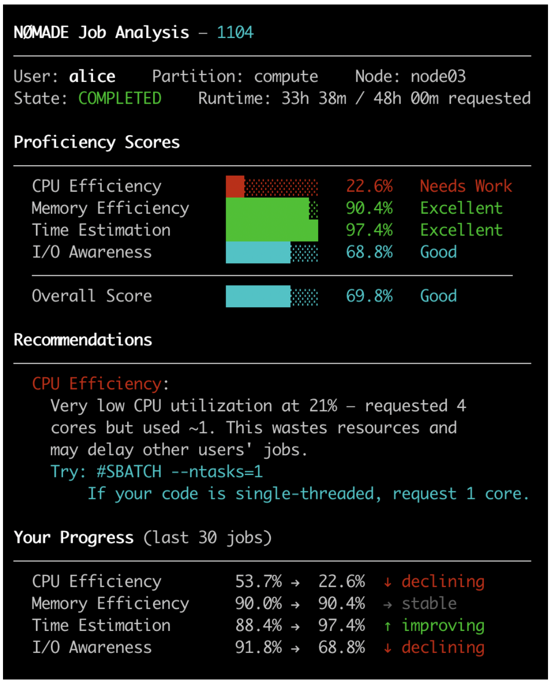
(2) Availability

Operating System

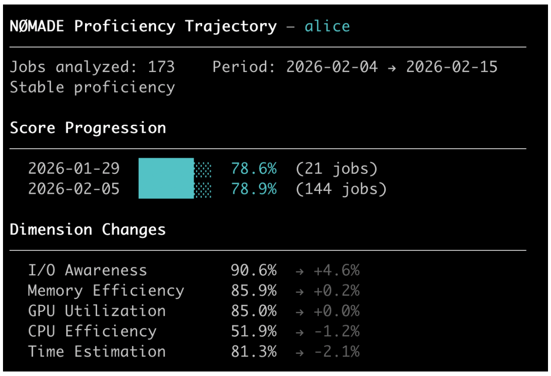
Linux (tested on Ubuntu 22.04, Rocky Linux 9, CentOS 7)

Programming Language

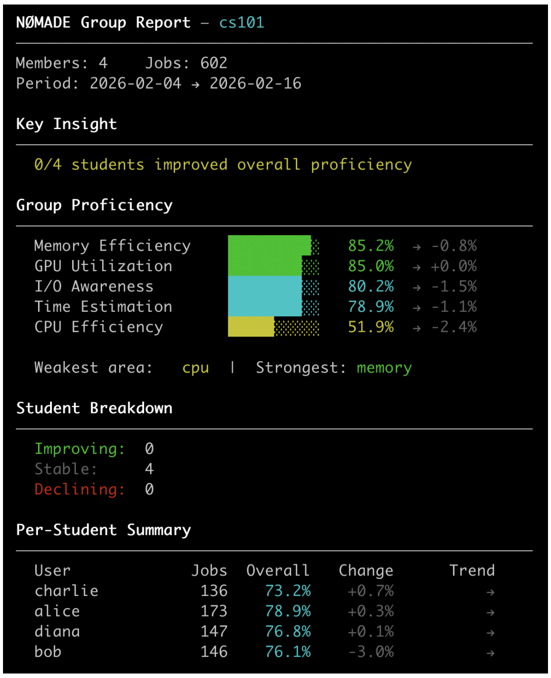
Python 3.9+



```
NØMADE Job Analysis — 1104

User: alice    Partition: compute    Node: node03
State: COMPLETED    Runtime: 33h 38m / 48h 00m requested

Proficiency Scores

  CPU Efficiency      ██░░░░░░░░  22.6%  Needs Work
  Memory Efficiency   █████████░  90.4%  Excellent
  Time Estimation     ██████████  97.4%  Excellent
  I/O Awareness       ███████░░░  68.8%  Good

  Overall Score       ███████░░░  69.8%  Good

Recommendations

  CPU Efficiency:
    Very low CPU utilization at 21% — requested 4
    cores but used ~1. This wastes resources and
    may delay other users' jobs.
    Try: #SBATCH --ntasks=1
        If your code is single-threaded, request 1 core.

Your Progress (last 30 jobs)

  CPU Efficiency       53.7% → 22.6%  ↓ declining
  Memory Efficiency    90.0% → 90.4%  → stable
  Time Estimation      88.4% → 97.4%  ↑ improving
  I/O Awareness        91.8% → 68.8%  ↓ declining
```



```
NØMADE Proficiency Trajectory — alice

Jobs analyzed: 173    Period: 2026-02-04 → 2026-02-15
Stable proficiency

Score Progression

  2026-01-29  ████████░░  78.6%  (21 jobs)
  2026-02-05  ████████░░  78.9%  (144 jobs)

Dimension Changes

  I/O Awareness         90.6%  → +4.6%
  Memory Efficiency     85.9%  → +0.2%
  GPU Utilization       85.0%  → +0.0%
  CPU Efficiency        51.9%  → -1.2%
  Time Estimation       81.3%  → -2.1%
```



```
NØMADE Group Report — cs101

Members: 4    Jobs: 602
Period: 2026-02-04 → 2026-02-16

Key Insight

  0/4 students improved overall proficiency

Group Proficiency

  Memory Efficiency    █████████░  85.2%  → -0.8%
  GPU Utilization      ████████░░  85.0%  → +0.0%
  I/O Awareness        ████████░░  80.2%  → -1.5%
  Time Estimation      ████████░░  78.9%  → -1.1%
  CPU Efficiency       █████░░░░░  51.9%  → -2.4%

  Weakest area:   cpu  |  Strongest: memory

Student Breakdown

  Improving:  0
  Stable:     4
  Declining:  0

Per-Student Summary

  User              Jobs  Overall   Change      Trend
  charlie            136    73.2%    +0.7%          →
  alice              173    78.9%    +0.3%          →
  diana              147    76.8%    +0.1%          →
  bob                146    76.1%    -3.0%          →
```

Figure 7: Educational analytics outputs. **Top left** (`nomad edu explain 1104`): Job explanation with proficiency scores and recommendations. **Top right** (`nomad edu trajectory alice`): User trajectory tracking improvement over 173 jobs. **Bottom** (`nomad edu report cs101`): Group report for a course section with per-student breakdown.

Additional System Requirements

- SLURM workload manager (optional, enables job-level analytics)
- sysstat package (`iostat`, `mpstat`) for system metrics
- Network access for dashboard (default port 5000)
- Minimal disk space for SQLite database (∼1MB per 10,000 samples)

Dependencies

- click $\geq$ 8.0
- toml $\geq$ 0.10
- numpy $\geq$ 1.21
- pandas $\geq$ 1.3
- scipy $\geq$ 1.7
- scikit-learn [10] (optional, for ML features)
- torch (optional, for GNN/LSTM models)
- torch-geometric (optional, for graph neural networks)

List of Contributors

1. Tonini, João Filipe Riva (Lead developer)

Software Location

Archive

Name:	Zenodo
Persistent identifier:	`https://doi.org/10.5281/zenodo.18614517`
Licence:	AGPL-3.0-or-later
Publisher:	João Filipe Riva Tonini
Version:	1.2.2
Date published:	February 2026

Code Repository

Name:	GitHub
Identifier:	`https://github.com/jtonini/nomad-hpc`
Licence:	AGPL-3.0-or-later
Date published:	December 2025

Package Repository

Name:	PyPI
Identifier:	`https://pypi.org/project/nomad-hpc/`
Installation:	`pip install nomad-hpc`

Project Website

URL:	`https://nomad-hpc.com`
Documentation:	`https://jtonini.github.io/nomad-hpc/`

Language

English

(3) Reuse Potential

NØMAD is designed for broad reuse across HPC environments of varying scales. The software can be deployed in several contexts:

Small Research Groups: Groups managing individual workstations or small clusters can use NØMAD's zero-infrastructure design to implement sophisticated monitoring without enterprise tools. The `nomad demo` mode allows evaluation without requiring existing HPC infrastructure.

Medium HPC Centers: University and institutional HPC centers can deploy NØMAD as either a primary monitoring solution or as a complement to existing tools like Grafana or Nagios, adding predictive capabilities to traditional threshold-based alerting.

Research Applications: The similarity network approach provides a framework for studying HPC failure patterns. The network structure, combined with job-level proficiency scores, enables quantitative analysis of resource usage patterns and their relationship to job outcomes.

Community Data Sharing: The `nomad community export` command generates anonymized datasets suitable for cross-institutional research. Export formats include:

- **Job fingerprints**: Feature vectors with hashed user/job identifiers
- **Failure patterns**: Aggregated statistics on failure modes by resource profile
- **Proficiency distributions**: Anonymized skill development trajectories

These exports enable collaborative research on HPC usage patterns without exposing sensitive institutional data. Researchers can study questions such as: Which failure modes are universal vs. site-specific? Do proficiency improvement rates vary by discipline? What resource configurations correlate with job success across different cluster architectures?

The anonymization process removes usernames, job names, and absolute timestamps while preserving the relational structure needed for network-based analysis. Exported data uses relative time offsets and hashed identifiers, making it suitable for publication as supplementary materials or inclusion in shared research datasets.

Educational Use: Beyond the demo mode, NØMAD provides a complete educational analytics pipeline. Instructors can assign Linux groups to course sections (e.g., `cs101`, `bio301`) and use `nomad edu report` to track class-wide proficiency development. Individual students receive automated feedback via `nomad edu explain`, reducing instructor workload while providing immediate, actionable guidance. The module is particularly valuable for:

- **HPC workshops**: Pre/post assessment of participant skills
- **Research onboarding**: Tracking new graduate students' development
- **Classroom instruction**: Automated grading of computational assignments
- **Self-directed learning**: Students can independently review their job history

Extension Points

Developers can extend NØMAD in several ways:

- **Custom Collectors**: Add new metric sources by implementing the collector interface

- **Alert Channels**: Integrate additional notification services beyond email/Slack/webhooks
- **ML Models**: Train site-specific models using the provided feature engineering pipeline
- **Dashboard Widgets**: Extend the web dashboard with custom visualizations

Support

- GitHub Issues: `https://github.com/jtonini/nomad-hpc/issues`
- Email: jtonini@richmond.edu
- Documentation: `https://jtonini.github.io/nomad-hpc/`

Users and developers are welcome to submit bug reports, feature requests, and pull requests via GitHub.

Acknowledgements

The author thanks George Flanagin for advice and inspiration on HPC system administration, and the University of Richmond's Office of the Provost for financial and resource support.

Funding Statement

This work was supported by the University of Richmond.

Competing Interests

The author declares no competing interests.

References

[1] Yoo, A.B., Jette, M.A., Grondona, M. (2003). SLURM: Simple Linux Utility for Resource Management. In: Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science, vol. 2862, Springer, pp. 44–60. DOI: `https://doi.org/10.1007/10968987_3`

[2] Prometheus Authors (2012–present). Prometheus: From metrics to insight. URL: `https://prometheus.io/`

[3] Grafana Labs (2014–present). Grafana: The open observability platform. URL: `https://grafana.com/`

[4] Galstad, E. (1999–present). Nagios: The industry standard in IT infrastructure monitoring. URL: `https://www.nagios.org/`

[5] Evans, R.T., Browne, J.C., Barth, W.L. (2014). Comprehensive resource use monitoring for HPC systems with TACC Stats. In: Proceedings of the First International Workshop on HPC User Support Tools, IEEE, pp. 13–21. DOI: `https://doi.org/10.1109/SC.2014.18`

[6] 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., Guillen, O., Cornelius, C.D. (2015). Open XDMoD: A tool for the comprehensive management of high-performance computing resources. Computing in Science & Engineering 17(4):52–62. DOI: `https://doi.org/10.1109/MCSE.2015.32`

[7] 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., Taerat, N., Tucker, T. (2014). The Lightweight Distributed Metric Service: A scalable infrastructure for continuous monitoring of large scale computing systems and applications. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, pp. 154–165. DOI: https://doi.org/10.1145/2616498.2616533

[8] Tuncer, O., Ates, E., Zhang, Y., Turber, A., Brandt, J., Leung, V.J., Egele, M., Coskun, A.K. (2017). Diagnosing performance variations in HPC applications using machine learning. In: High Performance Computing, Lecture Notes in Computer Science, vol. 10266, Springer, pp. 355–373. DOI: https://doi.org/10.1007/978-3-319-58667-0_19

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

[10] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12:2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html