Real-time Performance Monitoring of HPC Clusters: Techniques and Challenges

M S Sandeep Kamath[1], Ashwini K B[2]

[1]Department of Information Science and Engineering, RV College of Engineering, Mysuru Road, Bengaluru, 560059 , Karnataka, India

[2]Department of Information Science and Engineering, RV College of Engineering, Mysuru Road, Bengaluru, 560059, Karnataka, India

---***---

Abstract - In the era of exascale computing, High-Performance Computing (HPC) clusters have become essential for addressing complex computational challenges across various domains. The increasing scale and complexity of HPC systems pose significant challenges in ensuring optimal performance and resource utilization. Real-time performance monitoring has emerged as a critical requirement to address these challenges effectively. The paper offers a comprehensive analysis of real-time performance monitoring techniques and challenges within HPC clusters focusing on three key monitoring approaches: Prometheus for agent-based monitoring, the ELK (Elasticsearch, Logstash, Kibana) stack for log-based monitoring, and machine learning/artificial intelligence techniques for proactive analysis. The implementation of Prometheus and Node Exporter for real-time metrics collection, the utilization of the ELK stack for log aggregation and analysis, and the integration of ML/AI for anomaly detection and predictive analytics are explored. Challenges such as scalability, heterogeneity, and dynamic workloads are addressed in the context of each monitoring approach. The findings demonstrate that each technique offers unique advantages and trade-offs, providing valuable insights for improving HPC performance monitoring practices.

***Key Words*:** High-Performance Computing, real-time performance monitoring, Prometheus, ELK stack, Machine Learning, Artificial Intelligence, HPC clusters.

1.INTRODUCTION

High-Performance Computing (HPC) clusters have emerged as essential tools for addressing computationally intensive tasks across various domains, including scientific research, engineering simulations, and data analytics. These clusters, comprising interconnected nodes, operate collectively to process vast datasets and execute complex algorithms[1]. In recent years, the scale and complexity of HPC systems have grown exponentially, with clusters consisting of thousands to tens of thousands of nodes.

The architecture of HPC clusters varies widely, ranging from flat clusters with uniform node configurations to hierarchical clusters with leader-follower architectures[2]. In flat clusters, compute possesses equal capabilities and responsibilities, facilitating straightforward management and resource allocation using a high availability admin node. Conversely, hierarchical clusters employ leader nodes to coordinate and manage the activities of subordinate nodes, enabling more efficient task distribution and fault tolerance.

Despite the remarkable capabilities of modern HPC clusters, managing and optimizing their performance pose significant challenges. Issues such high system usage, hardware failures can impact cluster availability and reliability. System and hardware failures, in particular, can lead to downtime and disrupt ongoing computations, affecting overall productivity and efficiency[3]. Furthermore, the dynamic nature of HPC workloads, characterized by fluctuating demands and unpredictable resource requirements, adds another layer of complexity to cluster management.

In this paper, we delve into the techniques and challenges associated with real time system performance monitoring of HPC clusters. We focus on three primary monitoring approaches: agent-based monitoring with Prometheus[4], log-based monitoring using the ELK (Elasticsearch, Logstash, Kibana) stack, and the integration of machine learning and artificial intelligence for proactive analysis[5]. However, real-time monitoring in HPC environments presents unique challenges, including scalability, heterogeneity, and dynamic workload patterns.

The remainder of this paper is organized as follows. Section 2 reviews the existing literature related to HPC performance monitoring. Section 3 describes the techniques in detail. Section 4 discusses the challenges associated with HPC performance monitoring how each technique addresses these challenges. Section 6 showcases the results from implementing these techniques, including a comparison between Prometheus and the ELK stack. Finally, Section 7 discusses the comparative analysis, and Section 8 concludes the paper with a summary of findings and future research directions.

2. LITERATURE REVIEW

In the field of High-Performance Computing (HPC), the imperative for robust monitoring systems has grown substantially as clusters expand in size and complexity. This demand arises from the need to optimize resource utilization and system performance while catering to the diverse requirements of both end-users and system administrators. The following literature review synthesizes key insights from several seminal works in the field, providing a comprehensive overview of monitoring architectures, methodologies, and challenges in the HPC domain.

Agarwala et al [6]. highlight the significance of system-level resource monitoring in HPC environments, emphasizing its pivotal role in enabling runtime management activities essential for optimizing application behavior and resource allocation. The authors advocate for the development of flexible and extensible monitoring tools capable of capturing real-time system dynamics to facilitate adaptive management strategies. Li and Zhang [3] propose a novel HPC cluster monitoring system architecture designed to address the evolving demands of service-oriented computing paradigms. Their framework focuses on providing on-demand, re-configurable monitoring services tailored to diverse user requirements, thereby enhancing the adaptability and usability of HPC facilities within grid environments. Sanchez et al. [7] delve into the challenges posed by the scale and complexity of modern HPC platforms, particularly in the context of large-scale deployments like Trinity. Their work emphasizes the need for scalable monitoring systems equipped with flexible data processing pipelines to effectively analyze and respond to the vast amounts of monitoring data generated by such platforms. Schwaller et al. [8] underscore the escalating importance of monitoring HPC systems as they advance towards exascale computing. The authors advocate for datadriven insights facilitated by analysis-driven visualizations, leveraging sophisticated data pipeline architectures to extract meaningful information from the plethora of system metrics generated in HPC environments. Sukhija and Bautista [4] address the challenges of monitoring and managing extreme-scale HPC environments, particularly focusing on the integration of automation and orchestration systems like Kubernetes and Prometheus. Their proposed framework aims to provide a comprehensive infrastructure for real-time monitoring, efficient management, and proactive alerting, catering to the complexities of emerging exascale systems. Sharifi et al. [6] introduce PROMON, a framework for production monitoring of HPC applications, emphasizing its capability to provide granular insights into application performance and resource utilization without imposing significant monitoring overhead. Their work emphasizes the importance of real-time monitoring in enhancing productivity and resource efficiency in HPC environments. Hristova and Goranov [9] contribute a user-level framework for performance monitoring of HPC clusters, focusing on tools like Nagios, CACTI, and MRTG for system management and monitoring. Their work underscores the significance of comprehensive monitoring infrastructures in optimizing cluster performance and facilitating informed decision-making for both users and administrators. Collectively, these studies underscore the evolving landscape of HPC monitoring, emphasizing the need for adaptive, scalable, and user-centric monitoring solutions to address the challenges posed by increasingly complex computing environments.

3. METHODS

In HPC environments, monitoring system and hardware performance is crucial for maintaining cluster efficiency and reliability. Key metrics include CPU utilization, memory usage, disk I/O statistics, network traffic, and temperature sensors. In this section, we explore three key techniques used for real-time system performance monitoring:

3.1 Agent-Based Monitoring with Prometheus and Node Exporter

Agent-based monitoring involves deploying lightweight agents on individual nodes within the HPC cluster to collect system and application metrics. These agents gather data locally and transmit it to a central monitoring server for analysis and visualization. This approach offers several advantages, including low overhead, real-time data collection, and scalability. Agent-based monitoring solutions come in various forms, each tailored to specific monitoring needs. These solutions include tools such as Ganglia, Zabbix, Collectd, Nagios, and Icinga [4]. While the specifics vary, the core concept remains consistent: deploying agents on individual nodes to collect performance data and transmit it to a centralized monitoring server.

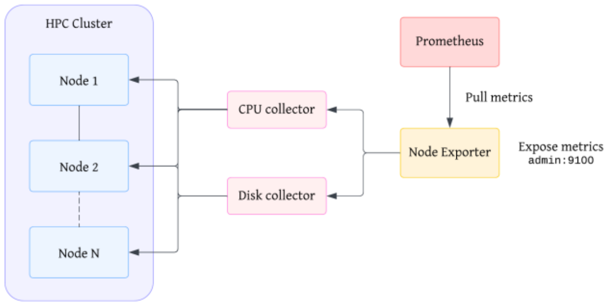

Fig. 1 Node Exporter and Prometheus in action, showcasing real-time data collection and monitoring in an HPC cluster

Prometheus, an open-source monitoring and alerting toolkit, has gained prominence in agent-based monitoring setups. Developed by SoundCloud, Prometheus boasts reliability, scalability, and extensibility, making it suitable for monitoring dynamic cloud-native environments and HPC clusters alike[4]. Using a pull-based approach, Prometheus scrapes metrics from instrumented targets (nodes) at regular intervals. Node Exporter serves as a critical component in Prometheus-based monitoring setups, facilitating the collection of a diverse range of system-level metrics from Linux and Unix-like systems. Metrics encompass CPU utilization, memory allocation, disk I/O statistics, network traffic, and more. Node Exporter exposes these metrics in a format compatible with Prometheus, allowing for seamless integration into monitoring pipelines.

Table 1 Important metrics exposed by Node Exporter [10]

Metric	Description	Operating System
cpu	Exposes CPU statistics.	Darwin, Linux
meminfo	Exposes memory statistics.	Darwin, Linux
diskstats	Exposes disk I/O statistics.	Darwin, Linux
filesystem	Exposes filesystem statistics, such as disk space used.	Darwin, Linux
loadavg	Exposes load average.	Darwin, Linux
netdev	Exposes network interface statistics such as bytes transferred.	Darwin, Linux
netstat	Exposes network statistics from /proc/net/netstat.	Linux
nvme	Exposes NVMe info from /sys/class/nvme/	Linux
os	Expose OS release info from /etc/os-release or /usr/lib/os-release	any
thermal	Exposes thermal statistics like pmset -g therm.	Darwin
time	Exposes the current system time.	any
uname	Exposes system information as provided by the uname system call.	Darwin, Linux
vmstat	Exposes statistics from /proc/vmstat.	Linux

As shown in Figure 1, Prometheus periodically scrapes metrics from Node Exporter endpoints exposed by individual nodes within the HPC cluster. These metrics as shown in the Table 1 are then stored in a time-series database, enabling querying, visualization, and analysis using Prometheus's query language (PromQL) and visualization tools like Grafana. This integrated approach streamlines the monitoring workflow, providing administrators with comprehensive insights into system performance. Node Exporter's broad spectrum of collected metrics spans critical aspects of system health and performance. This includes CPU utilization metrics detailing processor load, memory metrics revealing memory usage patterns, disk I/O statistics elucidating disk activity, network traffic metrics capturing network throughput, and various systemlevel indicators crucial for diagnosing performance issues and optimizing resource allocation.

3.2 Log-Based Monitoring using the ELK Stack

Log-based monitoring is another essential technique for gaining insights into the behavior and performance of HPC clusters. By analyzing log files generated by various components and applications running on the cluster, administrators can detect anomalies, troubleshoot issues, and optimize system performance[8]. The ELK stack comprises three core components: Elasticsearch, Logstash, and Kibana.

Table 2 Important metrics from syslog monitoring using ELK

Metric	Description
Kernel Messages	Events, errors, warnings, and hardware issues reported by the kernel.
System Events	System startups, shutdowns, reboots, and configuration changes.
Hardware Logs	Component failures, temperature alerts, disk errors, and hardware issues.
Authentication Logs	Login attempts, authentication failures, and user activity.
Service Logs	Service startups, shutdowns, errors, and service-specific events.
Network Logs	Connection attempts, configuration changes, errors, and traffic statistics.
Application Logs	Performance, errors, warnings, and usage patterns of HPC applications.

As shown in Figure 2, Elasticsearch serves as the distributed search and analytics engine, providing real-time indexing and search capabilities for log data. Logstash facilitates log ingestion, processing, and enrichment, enabling the transformation of raw log data into structured information. Kibana offers a powerful visualization platform, allowing administrators to create dashboards and explore log data interactively. Logstash plays a central role in log-based monitoring by ingesting log data from various sources, including log files, syslog, hardware logs, and other log collection agents. It processes and enriches the incoming log data using filters, allowing for data transformation, parsing, and normalization. Logstash then forwards the processed data to Elasticsearch for indexing and storage. Kibana complements Elasticsearch by providing a user-friendly interface for exploring and visualizing log data. Administrators can create custom dashboards, charts, and graphs to monitor key performance indicators, detect trends, and investigate issues. Kibana's powerful search and filtering capabilities enable users to drill down into specific log events and analyze them in real-time.

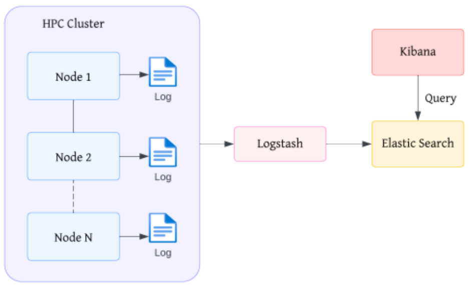

Fig. 2 Log monitoring architecture using ELK in an HPC cluster

In HPC environments, log-based monitoring using the ELK stack can provide valuable insights into system behavior, application performance, and user activities. By centralizing log data from various sources within the cluster, administrators can gain a holistic view of cluster health and performance. Syslog and hardware logs, in particular, offer critical information about system events, errors, and hardware failures and more as shown in Table 2, thus enabling proactive monitoring and troubleshooting [6]. Log-based monitoring offers several benefits for monitoring HPC clusters, including real-time visibility into system events, proactive detection of issues, and enhanced troubleshooting capabilities. By leveraging the ELK stack, administrators can streamline log management, improve incident response times, and optimize cluster performance effectively.

3.3 Integration of Machine Learning and Artificial Intelligence

ML and AI techniques enable HPC administrators to move beyond reactive monitoring and towards proactive analysis. By analyzing historical performance data, these techniques can identify patterns, predict future trends, and anticipate potential issues before they occur. This proactive approach allows administrators to take preventive measures, optimize system performance, and mitigate potential risks. As shown in Table 3 Anomaly detection is a key application of ML and AI in HPC performance monitoring. By learning normal system behavior from historical data, ML models can detect deviations from expected patterns indicative of anomalies or performance issues. Predictive analytics further extends this capability by forecasting future performance trends based on historical data, enabling administrators to anticipate resource demands and plan accordingly. While ML and AI offer significant potential for enhancing real-time performance monitoring in HPC clusters, several challenges must be addressed. These include the need for high-quality training data, the complexity of ML model selection and tuning, and the interpretability of MLdriven insights. Additionally, the dynamic nature of HPC workloads and environments presents unique challenges for ML and AI integration, requiring robust and adaptive monitoring solutions

Table 3 Machine Learning and AI methods for HPC

Task	Methods
Anomaly Detection	Statistical Methods (e.g., Z-score, Grubbs' Test) & Machine Learning Methods (e.g., Isolation Forest, One-Class SVM) & Deep Learning Methods (e.g., Autoencoders)
Prediction	Regression Models (e.g., Linear Regression, Polynomial Regression) & Classification Models (e.g., Logistic Regression, Decision Trees) & Ensemble Models (e.g., Random Forest, Gradient Boosting) & Neural Networks (e.g., Feedforward Neural Networks, Recurrent Neural Networks)
Forecasting	Time Series Models (e.g., ARIMA, SARIMA) & Exponential Smoothing Models (e.g., Holt-Winters) & Machine Learning Models (e.g., XGBoost, LSTM) & Neural Networks (e.g., CNN, RNN)

Effectively integrating ML and AI techniques into existing monitoring pipelines necessitates meticulous attention to various stages of the pipeline, including data collection, preprocessing, model training, and inference. Open-source frameworks like TensorFlow, PyTorch, and scikit-learn furnish robust tools and libraries for developing and deploying ML models in HPC environments[5]. By seamlessly incorporating MLdriven insights into monitoring pipelines, administrators can fortify their capacity to identify anomalies, prognosticate performance trends, and optimize cluster resources judiciously. Future research avenues encompass the formulation of specialized ML algorithms tailored to HPC workloads, the exploration of edge computing and distributed ML techniques to accommodate the scale and dynamics of HPC clusters, and the infusion of domain-specific expertise into ML-driven monitoring solutions. By advancing the frontier of ML and AI in HPC performance monitoring, researchers can unlock novel vistas for enhancing cluster efficiency and performance.

4. CHALLENGES

Despite the significant advancements in real-time performance monitoring techniques for HPC clusters, several challenges persist that hinder their effective implementation and utilization. In this section, we discuss some of the key challenges encountered in the monitoring of HPC clusters and propose potential strategies to address them.

4.1 Scalability

One of the primary challenges in real-time performance monitoring is scalability, particularly in large-scale HPC clusters comprising thousands of nodes[7]. As the cluster size increases, the volume of monitoring data escalates exponentially, posing significant challenges in data collection, storage, and analysis [9]. Traditional monitoring solutions may struggle to handle the sheer volume of data generated, leading to performance degradation and scalability limitations. Prometheus addresses scalability through its distributed architecture and efficient data collection mechanisms. It can scrape metrics from thousands of targets with minimal overhead, making it suitable for large-scale HPC environments. The use of federation allows Prometheus to scale horizontally, aggregating data from multiple Prometheus servers.

4.2 Heterogeneity

HPC clusters often exhibit heterogeneous architectures, with nodes varying in terms of hardware configurations, operating systems, and application workloads. This heterogeneity introduces complexity into the monitoring process, as monitoring tools must be capable of collecting and analyzing data from diverse sources while ensuring compatibility and consistency across the cluster. Managing heterogeneous environments requires careful consideration of interoperability, data normalization, and resource allocation strategies to ensure accurate and meaningful monitoring results. The ELK stack's flexibility and extensibility make it well-suited for heterogeneous environments. Logstash can be configured to process and transform log data from various sources, while Elasticsearch can index and search across different data formats. Kibana's visualization capabilities allow users to create custom dashboards tailored to specific systems and performance metrics.

4.3 Dynamic Workloads

The dynamic nature of HPC workloads as shown in Table 4 poses another significant challenge for real-time performance monitoring. Workload patterns in HPC clusters can vary unpredictably, with fluctuations in resource demands, job priorities, and application behavior. Traditional monitoring approaches that rely on static thresholds or predefined rules may struggle to adapt to these dynamic conditions, resulting in inaccurate or delayed insights. Addressing the challenge of dynamic workloads requires

Table 4 Heterogeneity in HPC Clusters

Aspect of Heterogeneity	**Description**
Node Hardware	Diverse hardware configurations: e.g., Node 1: Intel Xeon, Node 2: AMD EPYC, Node 3: Custom ARM
Operating Systems	Different OS: Linux, Unix, specialized HPC OS
Application Workloads	Varied characteristics: compute-intensive, data-intensive
Network Topology	Various networks: Ethernet, InfiniBand, RDMA
Job Scheduling Policies	Heterogeneous scheduling policies: e.g., Slurm, PBS, optimize resource allocation

the development of adaptive monitoring techniques capable of dynamically adjusting to changing conditions and detecting anomalies in real-time. Machine learning techniques excel at handling dynamic workloads by continuously learning from new data and adapting to changing patterns. Models can be trained to recognize normal and anomalous behavior, enabling real-time detection of performance issues and proactive resource management. Reinforcement learning, in particular, can optimize system performance by dynamically adjusting configurations based on realtime feedback.

4.4 Resource Overhead

Real-time performance monitoring imposes additional overhead on HPC clusters, potentially impacting system performance and resource utilization. Monitoring agents, data collectors, and analysis tools consume computational resources and network bandwidth, competing with user applications for valuable resources. Minimizing the resource overhead associated with monitoring without compromising the accuracy and effectiveness of the monitoring process is a critical challenge. Strategies such as lightweight monitoring agents, distributed data processing, and resource-aware scheduling algorithms can help mitigate this challenge and ensure minimal disruption to cluster operations. Each of the discussed techniques is designed to minimize resource overhead in different ways. Prometheus uses a pull-based model and efficient storage formats to reduce the impact on system resources. The ELK stack, while powerful, requires careful configuration to avoid excessive resource consumption, such as optimizing Logstash pipelines and Elasticsearch indices. Machine learning models can be computationally intensive; however, leveraging lightweight algorithms and periodic retraining can help balance resource usage with monitoring effectiveness.

5. RESULTS

In this section, we present the results of implementing (1) Prometheus with Node Exporter and (2) ELK syslog monitoring for real-time performance monitoring in our HPC cluster environment. By leveraging these tools, we aimed to gain insights into system-level metrics and optimize resource allocation based on the observed performance trends.

5.1 Agent-Based System Monitoring with Prometheus, Node Exporter and Grafana

Prometheus is deployed and configured to scrape metrics from Node Exporter endpoints exposed by individual nodes within our HPC cluster. Node Exporter facilitated the collection of a diverse range of system-level metrics, including CPU utilization, memory allocation, disk I/O statistics, and network traffic. These metrics were then stored in a time-series database for querying and visualization.

Using Grafana, we created custom dashboards to visualize the collected systemlevel metrics in real-time. The dashboards provided insights into CPU utilization trends, memory usage patterns, disk activity, and network throughput across different nodes within the cluster[11]. This visualization enabled us to identify performance bottlenecks, optimize resource allocation, and enhance overall system efficiency.

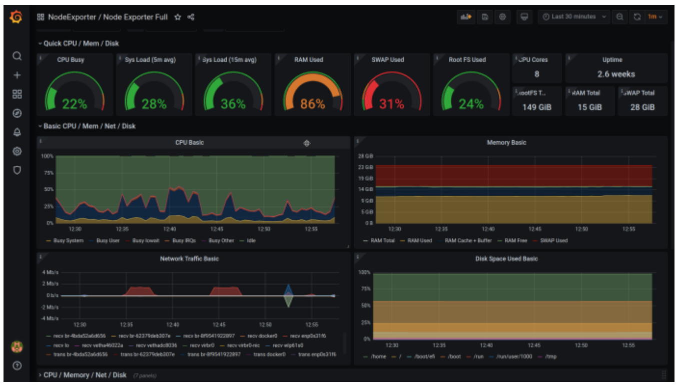

Fig. 3 Comprehensive visualization of HPC cluster system metrics through the Grafana Node Graph dashboard, showcasing real-time insights into CPU utilization, memory usage, disk I/O, network traffic, and other vital performance indicators

As an illustrative example, 3 depicts a node graph generated from the collected metrics using Grafana. The graph displays CPU utilization trends for individual nodes within the HPC cluster over a specific time period. By analyzing these trends, we were able to identify nodes experiencing high CPU load and take proactive measures to mitigate potential performance issues.

5.2 System Log-Based Monitoring using the ELK Stack

Logstash was configured to ingest log data from multiple sources, including system logs, application logs, and network logs. Custom pipelines were defined to parse, filter, and enrich the incoming log entries to ensure compatibility with Elasticsearch for efficient indexing and storage. Through this process, we successfully collected a comprehensive dataset of log events generated by different components of the HPC cluster.

Kibana served as the visualization platform, allowing us to explore and analyze the log data in real-time. Custom dashboards were created to monitor key system metrics, track performance trends, and identify anomalies. With Kibana’s powerful search and filtering capabilities, we were able to drill down into specific log events and extract actionable insights to optimize cluster performance effectively.

As an illustrative example, 4 depicts Our implementation seamlessly integrated Syslog-based monitoring with our HPC cluster infrastructure. By centralizing log data from diverse sources within the cluster, we gained a holistic view of system health and performance. This facilitated proactive management and troubleshooting, enabling us to identify and address issues promptly to ensure optimal cluster operation.

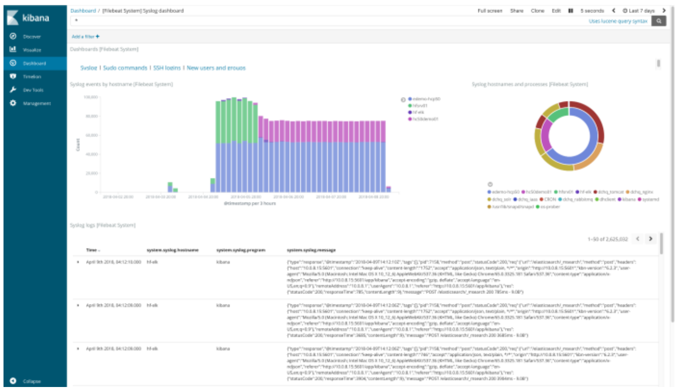
kibana
Discover
Visualize
Dashboard
Timelion
Dev Tools
Management
Dashboard / [Filebeat System] Syslog dashboard
Syslog events by hostname [Filebeat System]
Syslog hostnames and processes [Filebeat System]
Syslog logs [Filebeat System]
Collapse

Fig. 4 Comprehensive visualization of HPC cluster system metrics through the Kibana dashboard, showcasing real-time insights into syslog performance indicators.

Table 5 Comparison between Prometheus and ELK stack

Criteria	Prometheus	ELK Stack
Scalability	Efficiently collects and stores metrics from thousands of nodes with minimal impact on system resources.	Scalable but requires careful management of Elasticsearch indices and Logstash pipelines to prevent excessive resource consumption.
Heterogeneity	Effective in homogeneous environments; struggles with diverse data formats without significant customization.	Excels in handling heterogeneous data sources, providing flexible configurations for various log formats.
Dynamic Workloads	Adapts well to dynamic workloads, providing real-time insights through continuous metric scraping.	Manages dynamic workloads effectively by aggregating and analyzing log data in real-time, though sometimes lagged due to log processing overhead.
Resource Overhead	Maintains low resource overhead with its efficient pull-based model and storage formats.	Requires careful tuning to keep resource overhead manageable, especially in large-scale deployments.
Ease of Implementation	Relatively straightforward to implement with robust community support and documentation.	More complex implementation requiring integration of multiple components (Elasticsearch, Logstash, Kibana), but offers powerful log analysis capabilities.

The comparison presented in Table 5 highlights the strengths and weaknesses of two widely used monitoring and logging solutions, Prometheus and ELK Stack. Prometheus demonstrates efficiency in real-time metric collection and scalability, making it suitable for environments with numerous nodes. However, its effectiveness diminishes in heterogeneous environments due to limited support for diverse data formats. On the other hand, ELK Stack excels in handling various log formats and providing robust log analysis capabilities, albeit with higher complexity in deployment and management. Understanding these differences is crucial for selecting the most appropriate solution based on the specific requirements and characteristics of the system under consideration.

6. CONCLUSIONS

In conclusion, The real-time performance monitoring in HPC clusters has provided invaluable insights into managing infrastructure effectively. Utilizing tools like Prometheus and Node Exporter for agent-based monitoring, alongside the ELK stack for log-based analysis, offers a robust framework for gathering detailed metrics and insightful analysis, empowering administrators with proactive system oversight. The integration of machine learning and artificial intelligence introduces a promising avenue for predictive analytics, enabling early anomaly detection and actionable insights to optimize cluster performance and reliability. Each technique addresses specific challenges associated with scalability, heterogeneity, dynamic workloads, and resource overhead. Our findings highlight the importance of selecting the appropriate monitoring solution based on the unique characteristics of the HPC environment. Future research should explore hybrid approaches that combine the strengths of these techniques to further enhance HPC performance monitoring.

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to Sagar B M, K. N. Subramanya, and Merin Meleet for their invaluable contributions to this research. Their expertise, guidance, and support have been instrumental in shaping the trajectory of this study. We would also like to thank the reviewers for their insightful comments and suggestions, which have greatly improved the quality of this paper.

REFERENCES

1. Moore, C.L., Khalsa, P.S., Yilk, T.A., Mason, M.: Monitoring high performance computing systems for the end user, 714–716 (2015) https://doi.org/10.1109/ CLUSTER.2015.124
2. Almeida, S.: An introduction to high performance computing, vol. 28, p. 40021 (2013). https://doi.org/10.1142/S0217751X13400216
3. Li, M., Zhang, Y.: Hpc cluster monitoring system architecture design and implement, 325–327 (2009) https://doi.org/10.1109/ICICTA.2009.314
4. Sukhija, N., Bautista, E.: Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus. In: 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Unknown (2019)
5. Anton, L., Willemot, S., Gougeaud, S., Zertal, S.: ML-Based Methodology for HPC Facilities Supervision, pp. 307–319 (2023). https://doi.org/10.1007/ 978-3-031-40843-4 23
6. Agarwala, S., Poellabauer, C., Kong, J., Schwan, K., Wolf, M.: System-level resource monitoring in high-performance computing environments. J. Grid Comput. 1, 273–289 (2003) https://doi.org/10.1023/B:GRID.0000035189.80518.5d
7. Moore, C.L., Khalsa, P.S., Yilk, T.A., Mason, M.: Monitoring high performance computing systems for the end user, 714–716 (2015) https://doi.org/10.1109/ CLUSTER.2015.124
8. Almeida, S.: An introduction to high performance computing, vol. 28, p. 40021 (2013). https://doi.org/10.1142/S0217751X13400216
9. Li, M., Zhang, Y.: Hpc cluster monitoring system architecture design and implement, 325–327 (2009) https://doi.org/10.1109/ICICTA.2009.314
10. Sukhija, N., Bautista, E.: Towards a framework for monitoring and analyzing high performance computing environments using kubernetes and prometheus. In: 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Unknown (2019)
11. Anton, L., Willemot, S., Gougeaud, S., Zertal, S.: ML-Based Methodology for HPC Facilities Supervision, pp. 307–319 (2023). https://doi.org/10.1007/ 978-3-031-40843-4 23