SUSTech HPC Monitor System

MiLe Shi
12212921@mail.sustech.edu.cn

YuXian Wu
12212614@mail.sustech.edu.cn

YuHang Wang
12212910@mail.sustech.edu.cn

***Abstract*— The SUSTech HPC Monitor System is a scalable platform for monitoring and managing high-performance computing clusters. Built on Prometheus with Node Exporter, IPMI Exporter, and Alertmanager, it offers real-time metrics collection, automated alerting, and role-based access control (RBAC). Successfully deployed on test clusters, the system enhances resource utilization, anomaly detection, and operational stability, while showcasing advancements in monitoring, backend development, and deployment practices.**

Keywords—Prometheus, Node Exporter, Alertmanager, Java Application, HPC, Monitoring

I. INTRODUCTION

A. Background

The SUSTech high-performance computing (HPC) platform includes large-scale clusters, such as Taiyi with over 800 compute nodes and Qiming 2.0 with nearly 300 nodes, interconnected by high-speed networks and utilizing parallel file systems. With a combined theoretical peak performance exceeding 25 petaflops, these clusters support computationally intensive tasks across various research disciplines.

A key limitation of many traditional monitoring systems is their inability to handle large-scale, distributed architectures with ease. Additionally, some systems may not provide real-time, fine-grained metrics on both software and hardware components or may require extensive configuration and maintenance

Given these challenges, there is a need for a customized monitoring solution that integrates seamlessly with the existing HPC infrastructure. This system must be capable of scaling with the growing platform, providing real-time insights into performance, and offering automated alerts to identify and address issues proactively. By leveraging tools like Prometheus, known for its scalability and powerful querying capabilities, and Java for its integration flexibility, SUSTech can develop a monitoring system tailored to its unique needs.

Setting up this system will ensure continuous monitoring, early detection of failures, and optimal resource management. The system will provide administrators with the tools to respond quickly to issues, prevent downtime, and support the growing demands of SUSTech's HPC platform.

B. Our Goal

The goal is to develop a comprehensive monitoring system to manage multiple high-performance computing (HPC) clusters at SUSTech. The system will leverage Prometheus, a powerful open-source monitoring and alerting toolkit, and integrate essential components such as Node Exporter for collecting server metrics and Alertmanager for managing and routing alerts.

The system will be developed using Java Spring Boot as the backend framework, with MyBatis for database access, MySQL for storing and managing metadata, and Docker for containerized deployment. These tools will ensure scalability, maintainability, and ease of deployment in the HPC environment.

Key features of the system include:

- Real-time monitoring of server performance metrics, such as CPU usage, memory consumption, disk I/O, network traffic, and GPU utilization, across multiple clusters.
- Automated alerting, where alerts are dynamically generated and managed based on predefined thresholds or anomalies detected in server metrics.
- Visualization dashboards to provide intuitive insights into cluster health and performance.
- Seamless integration with existing HPC infrastructure to ensure the system supports the operational needs of SUSTech's computing environment.

This system will play a crucial role in ensuring the reliability, efficiency, and proactive management of SUSTech's HPC clusters, enabling administrators to detect and resolve issues early, minimize downtime, and optimize resource utilization.

C. Our Work

Building upon existing work, our project incorporates significant innovations and requires substantial development efforts to achieve its full potential. The current progress includes foundational elements, but there are key areas where further work is being actively carried out:

- RBAC (Role-Based Access Control) Permission System:
  We are designing and implementing a robust RBAC system to ensure secure and efficient user management. This system will allow administrators to define and assign roles with specific permissions, enabling fine-grained control over access to monitoring data, alert management, and system configurations. This feature is critical for ensuring the system's usability in a multi-user environment while maintaining strict security standards.
- Further Exploration of Prometheus, Node Exporter, and Alertmanager:
  While the system already integrates Prometheus for monitoring, Node Exporter for metric collection, and Alertmanager for alert routing, there is still room for deeper exploration and optimization. This includes refining metrics collection, enhancing alerting logic, and optimizing data query performance. By diving deeper into Prometheus's advanced features (e.g., recording rules and query optimizations), the system can provide even more actionable insights and scalability.
- Code Refactoring:
  Significant portions of the existing codebase require refactoring to improve maintainability, readability, and performance. This process involves restructuring

and optimizing code, adhering to best practices in software development, and ensuring compatibility with the evolving system requirements. The refactoring effort is critical for ensuring long-term scalability and ease of further development.

- Testing and Deployment:
  The system has been successfully deployed on a testing cluster, where it is currently operational and performing well. This deployment phase allows us to evaluate its stability, collect user feedback, and identify areas for further improvement. Stress testing, edge-case handling, and resource optimization are being actively conducted to prepare the system for deployment on the production HPC clusters.

While significant progress has been made, the project still demands continuous innovation and development to achieve its ultimate vision. These efforts will ensure the system is not only functional but also scalable, secure, and fully optimized for the needs of SUSTech's high-performance computing environment.

II. Overview of Project

A. Architecture:

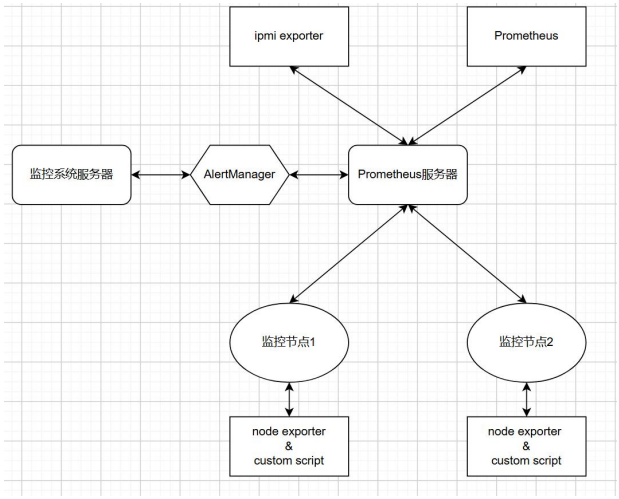

Fig. 1. Server Architecture

Monitoring System Architecture

The custom monitoring system at SUSTech builds upon the Prometheus architecture and integrates additional components for enhanced functionality. The workflow is illustrated in Figure 1:

- Metrics Collection: Compute nodes run Node Exporter and custom scripts to expose metrics. Prometheus scrapes these metrics periodically. IPMI Exporter is used to gather hardware-specific data.
- Alert Processing: Prometheus evaluates scraped metrics against alerting rules. For instance, it can trigger alerts when GPU utilization exceeds a threshold or when a node becomes unresponsive. Alerts are sent to Alertmanager, which routes them to administrators via email or other channels.
- Data Management and Visualization: The Main Server interacts with Prometheus and Alertmanager to provide a user interface for monitoring and managing the system. Data visualizations and dashboards are generated using tools like Grafana, allowing administrators to monitor the health of the clusters in real time.
- System Administration: The main server hosts a Java-based application that provides RBAC for managing user permissions and allows administrators to configure alerting rules, manage targets, and analyze historical data.

B. Prometheus

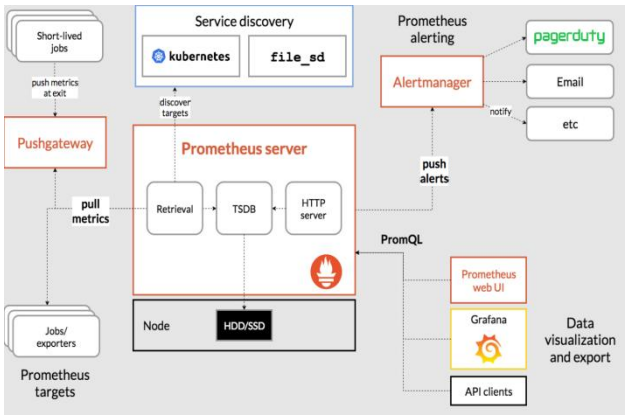

Fig. 2. Prometheus Architecture

The monitoring system leverages Prometheus as the core component for metric collection, storage, and alerting. The architecture integrates Node Exporter, IPMI Exporter, and Alertmanager to provide real-time monitoring, proactive alerting, and seamless integration with the custom monitoring system.

- Metric Collection via Exporters
  - Prometheus employs a pull-based approach to collect metrics from Node Exporter and IPMI Exporter:
    - Node Exporter collects system-level metrics, including CPU, memory, disk, and network usage, from compute nodes in the cluster. These metrics are exposed via HTTP endpoints and scraped by Prometheus at regular intervals.
    - IPMI Exporter gathers hardware-level metrics, such as temperature, power usage, and fan speeds, by communicating with the Baseboard Management Controller (BMC). These metrics provide critical insights into the physical health of the cluster.
  - The scraped data is stored in Prometheus's Time-Series Database (TSDB), enabling historical analysis and real-time querying
- Alerting Workflow
  - Prometheus evaluates metrics against predefined alert rules stored in its configuration. When a rule's condition is met, an alert is generated. Alerts typically capture issues like:
    - High CPU or memory usage.
    - Hardware failures or overheating.

        - Network bottlenecks or unresponsive nodes.
    - Generated alerts are forwarded to Alertmanager for processing.
- Integration with Alertmanager
    - Alertmanager manages and routes alerts generated by Prometheus. It performs tasks such as:
        - Grouping and Deduplication: Combining similar alerts to reduce noise.
        - Routing: Directing alerts to specific channels, including email, Slack, or custom systems via webhooks.
    - For custom integration, Alertmanager pushes alerts to the monitoring system via webhook. These alerts are then processed by the custom Java-based application, which logs, visualizes, and notifies administrators.

III. Main Issue and Solution

A. *Lack of permission management*

1) Issue

The current monitoring system lacks proper permission management, allowing all users to:

- Modify alerting rules and monitoring targets.
- Access sensitive metrics without restrictions.

This poses significant security and operational risks, such as unauthorized changes, data breaches, and lack of accountability.

2) Solution:Role-Based Access Control (RBAC)

To address these challenges, an RBAC system has been implemented, providing structured access control based on user roles and permissions. The key components of this solution include:

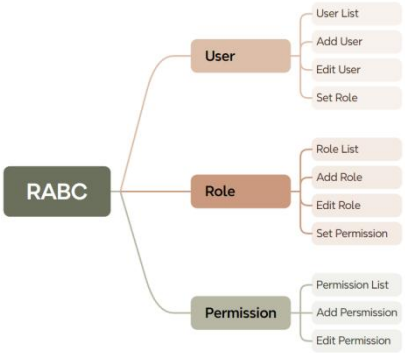

Fig. 3. RBAC Architecture

a) Core Components

- User: Represents the individual accessing the system.
- Role: Defines a set of permissions, such as "Administrator" or "Viewer."
- Permission: Specifies granular actions, like "Edit Alert Rules" or "View Metrics."

b) Workflow

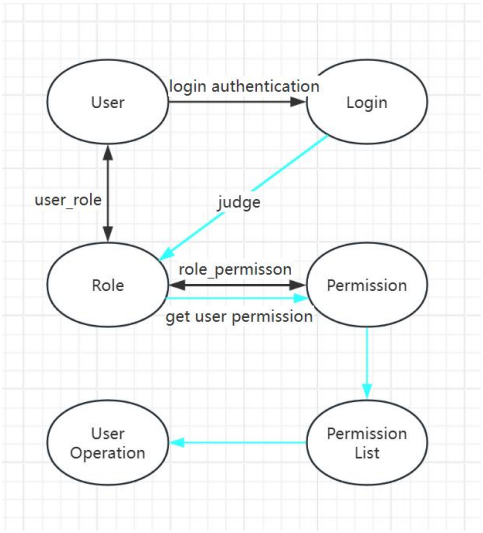

Fig. 4. RBAC Relationship

AUTHENTICATION AND ROLE ASSIGNMENT:

Users log into the system and are assigned specific roles, determining their access level.

```
<el-button ··· v-if="permission_role.includes('role:update')">更新权限</el-button>
<el-button ··· v-if="permission_role.includes('role:delete')">删除权限</el-button>
<el-button ··· v-if="permission_role.includes('role:add')">新建权限</el-button>
```

Fig. 5. Permission Verification in Front-end

PERMISSION ENFORCEMENT:

Permissions are mapped to roles, controlling which actions users can perform. For example:

- Administrators can modify alert rules and monitoring targets.
- Viewers can only view metrics without modifying settings.

Fig. 6. Role Management

Fig. 7. User Management

ROLE AND PERMISSION MANAGEMENT:

Administrators can manage users, roles, and permissions, ensuring flexibility and scalability. For example:

- Add or edit roles.
- Assign permissions to roles.
- Assign roles to users.

B. *Need of software-level metrics collection*

ISSUE

The current monitoring system only exposes hardware metrics through BMC, while critical OS-level metrics such as CPU usage, memory consumption, and disk I/O are omitted. This limits visibility into the overall health and performance of the compute nodes.

SOLUTION: NODE EXPORTER

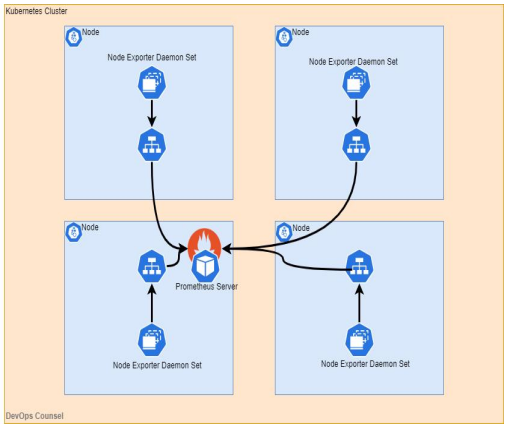

Fig. 8. Node Exporter Architecture

To address this, Node Exporter is deployed on each compute node to collect OS-level metrics, including:

- CPU usage, memory, disk I/O, and network traffic.

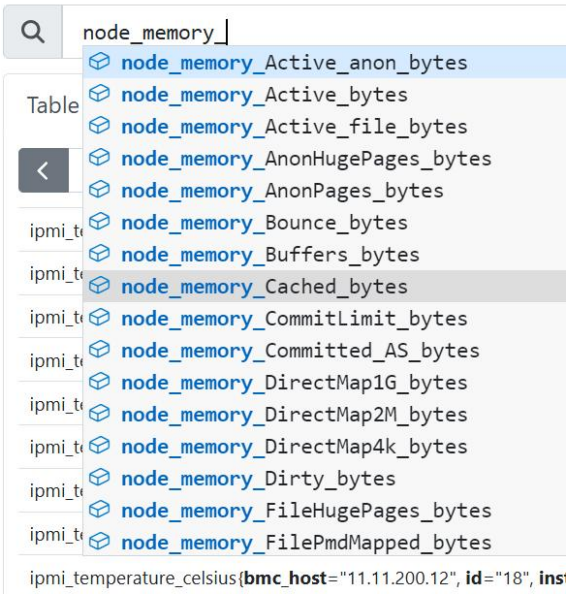

Fig. 9. Memory Metrics

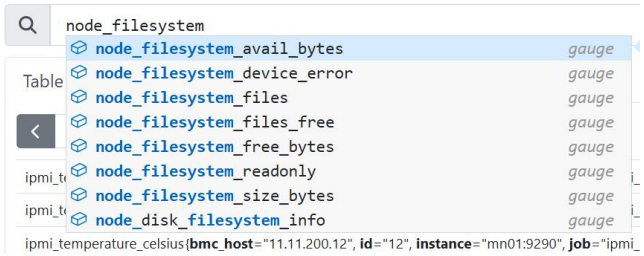

Fig. 10. File System Metrics

Additionally, custom scripts are implemented to gather application-specific or HPC workload-related metrics, such as:

- GPU utilization and job performance.

C. *Incompleteness of Alerts management and classification*

ISSUE

The current system lacks effective alert management, leading to issues such as:

- Duplicate alerts for the same issue.
- Difficulty in categorizing and routing alerts to appropriate channels

SOLUTION

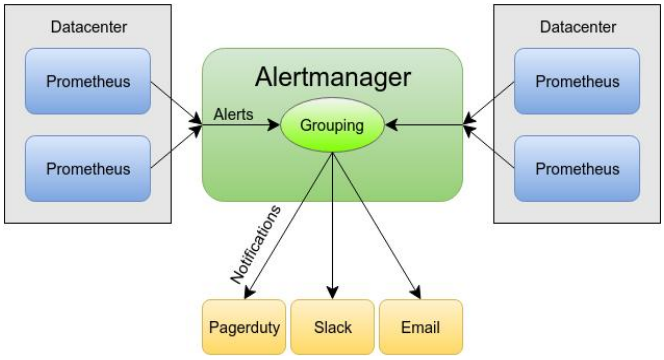

Fig. 11. Alertmanager

To resolve these issues, Alertmanager is introduced to:

- Deduplicate alerts: Combine duplicate alerts to avoid unnecessary noise.
- Group and classify alerts: Organize alerts based on labels, such as severity or source.
- Route alerts: Send alerts to appropriate notification channels (e.g., email, webhook) based on predefined rules.

IV. ACHIEVEMENT

A. *Successful Deployment and Operational Efficiency*

The system has been successfully deployed on the test clusters Taiyi and Qiming and is accessible via the URL http://172.18.6.108/. This deployment covers the monitoring of three compute nodes, demonstrating its effectiveness in managing and visualizing metrics at scale. Initial testing confirms the system's stability, reliability, and alignment with predefined monitoring and alerting requirements.

B. Comprehensive Monitoring and Alerting Framework

The system integrates Prometheus, Node Exporter, and Alertmanager, enhanced with customized back-end functionalities. Alerts are generated based on predefined rules, ensuring timely and actionable notifications. Advanced features include support for custom metrics such as GPU utilization and user activity, critical for optimizing computational workloads.

Through the integration of the Prometheus TextCollector, the system extends its capabilities to dynamically collect metrics beyond standard hardware and software parameters. For instance, administrators can monitor cluster-specific performance metrics and configure alerts based on specific thresholds, improving decision-making and operational oversight.

C. Dynamic Device and Cluster Management

Device and cluster management functionalities are at the core of the system, enabling administrators to dynamically add, modify, or delete devices and clusters. Batch addition of devices via Excel uploads and JSON-based customization for device-specific attributes streamline operations. This functionality enhances the system's scalability and simplifies the onboarding of new hardware.

D. Advanced Alert Statistics and Analysis

The system introduces an alert statistics module, allowing administrators to analyze historical alerts across dimensions such as type, severity, and resolution status. These analytics enable trend identification and resource optimization while providing actionable insights into system performance. Alerts are visually categorized, facilitating prioritization and efficient resolution of critical issues.

E. Role-Based Access Control (RBAC)

A sophisticated RBAC mechanism ensures that users can access only the resources and functionalities pertinent to their roles. Permissions and alerting rules are dynamically managed through an intuitive interface, enhancing the security and administrative control of the monitoring infrastructure. The system supports the management of user roles and permissions, ensuring fine-grained access control and organizational alignment.

F. Intuitive and Interactive User Interface

The Vue.js-based frontend provides a streamlined and visually engaging interface. Users can navigate seamlessly through dashboards, device management pages, and alert configurations. Key features include:

- Real-time visualization of CPU and GPU utilization across clusters.
- Dynamic device displays tailored to specific clusters.
- Alert rule management for configuring and refining alert conditions.

The system's design prioritizes usability, enabling administrators to interact with complex monitoring data effectively.

G. Key Outcomes and Impact

The system monitors multiple compute nodes and dynamically manages devices and clusters.

Alerting and notification functionalities ensure timely responses to potential issues.

Advanced analytics and role management enhance administrative efficiency.

The system's deployment has validated its design principles, setting a strong foundation for production deployment and scalability.

V. GAIN

The development of the SUSTech HPC Monitor System provided a valuable opportunity for the team to enhance their technical expertise and collaborative abilities. This project not only delivered a robust monitoring solution but also facilitated growth in key areas:

A. Advanced Java Web Development

Building the backend with Spring Boot, MyBatis, and MySQL deepened the team's understanding of modern web development practices, including RESTful API design, ORM, and dependency injection. Integrating tools like Hutool and Jackson further refined their ability to optimize and maintain complex systems.

B. Expertise in Monitoring Systems

The integration of Prometheus, Node Exporter, and Alertmanager provided hands-on experience in building a comprehensive monitoring framework. The team gained proficiency in PromQL for extracting and visualizing metrics, as well as configuring alerting mechanisms for real-time anomaly detection and resolution.

C. Proficiency in Linux and Deployment

Deploying and managing the system in a Linux environment improved the team's command-line, scripting, and system administration skills. Experience with Docker and YAML configuration enhanced their ability to manage containerized applications and automate deployments.

VI. TIMELINE

The development of the SUSTech HPC Monitor System followed a structured timeline, which encompassed significant milestones across four months, ensuring a systematic and goal-oriented approach to the project.

A. September: Project Initiation

The project commenced with a thorough review of the existing code, aimed at understanding its structure and functionality. Efforts were focused on deploying the system on a server, resolving compatibility issues, and fixing bugs to ensure that the code was runnable. This phase established the foundational groundwork for the project.

B. October: Role-Based Access Control (RBAC) Development

The next phase involved designing and implementing an RBAC mechanism to enhance system security and administrative control. Key activities included:

Adding permission control to regulate user access.

Deploying the Node Exporter across compute nodes to facilitate hardware and software metric collection.

Allowing the configuration of custom alerting rules, enabling tailored monitoring solutions.

C. November: AlertManager Integration

During this phase, the focus shifted to integrating Alertmanager for efficient alert management. The following tasks were accomplished:

Utilizing Alertmanager to handle alerts, including grouping, deduplication, and delivery.

Adding visualizations to the dashboard for a more intuitive display of system metrics and alerts.

Developing custom scripts to enhance monitoring capabilities and streamline system operations.

D. December: Dynamic Cluster Refactor

The final phase centered on enhancing the system's adaptability and scalability through dynamic cluster management. This involved:

- Allowing the dynamic configuration of clusters to accommodate the evolving needs of the HPC Center.
- Deploying the system onto test clusters, such as Taiyi and Qiming, to evaluate performance and reliability.
- Creating a comprehensive user manual to support system deployment, configuration, and usage.

VII. References

[1] Prometheus, "Prometheus: Monitoring system and time series database," [Online]. Available: https://prometheus.io/. [Accessed: Dec. 25, 2024].

[2] Prometheus, "Node Exporter," [Online]. Available: https://prometheus.io/docs/guides/node-exporter/. [Accessed: Dec. 25, 2024].

[3] Prometheus, "Alertmanager," [Online]. Available: https://prometheus.io/docs/alerting/latest/alertmanager/. [Accessed: Dec. 25, 2024]