Project Report On

High Availability HPC Cluster

Submitted

In partial fulfilment

For the award of the Degree of

Post Graduate Diploma in HPC System Adminstration

(C-DAC, ACTS (Pune))

Guided by:

Mr. Roshan G

Submitted by:

Yogesh Mahajan	PRN: 250840127011
Satyam Kasar	PRN: 250840127009
Rushikesh Patil	PRN: 250840127013
Prince Philip	PRN: 250840127016
Rohanshi Gawande	PRN: 250840127017

Centre of Development of Advanced Computing (C-DAC) ACTS (Pune – 411008)

CERTIFICATE

TO WHOMSOEVER IT MAY CONCERN

This is to certify that

Yogesh Mahajan	PRN: 250840127011
Satyam Kasar	PRN: 250840127009
Rushikesh Patil	PRN: 250840127013
Prince Philip	PRN: 250840127016
Rohanshi Gawande	PRN: 250840127017

Have successfully completed their project on

“High Availability HPC Cluster”

Under the guidance of Mr. Roshan G.

Project Guide

Mr. Roshn G.

Course Coordinator

Ms. Divya Patel

HOD ACTS

Mr. Gaur Sunder

ACKNOWLEDGEMENT

This is to acknowledge our indebtedness to our Project Guide, **Mr. Roshan G** , CDAC ACTS, Pune for his constant guidance and helpful suggestions for preparing this project **High Availability HPC Cluster.**

We express our deep gratitude towards his inspiration, personal involvement, constructive criticism that he provided us along with technical guidance during the course of this project.

We take the opportunity to thank head of department **Mr. Gaur Sunder** for providing us such a great infrastructure and environment of our overall development.

We express sincere thanks to **Mr. Soumyakant Behera**, Process Owner, for their kind cooperation and their extendible support towards the completion of our project.

It is our great pleasure in expressing sincere and deep gratitude towards **Mrs. Risha P.R.** (Program Head) and **Ms. Divya Patel**(Course Coordinator, PG-DHPCSA) for their valuable guidance and constant support throughout this work and help to pursue additional studies.

Also, our warm thanks to **C-DAC ACTS** Pune, which provided us this opportunity to carry out this prestigious Project and enhance our learning in various technical fields.

Yogesh Mahajan	PRN: 250840127011
Satyam Kasar	PRN: 250840127009
Rushikesh Patil	PRN: 250840127013
Prince Philip	PRN: 250840127016
Rohanshi Gawande	PRN: 250840127017

ABSTRACT

This project demonstrates the design and implementation of a **High Availability (HA) High-Performance Computing (HPC) cluster** engineered for maximum resilience and fault tolerance. Utilizing a five-machine architecture, the system integrates a centralized login node, **active-passive master nodes**, and **active-passive worker nodes** to eliminate single points of failure.

The orchestration layer leverages **SLURM** for automated workload management, ensuring that in the event of a hardware or service failure, jobs are seamlessly transitioned between active and standby nodes without data loss. Infrastructure reliability is anchored by essential services including **DNS, DHCP, NFS (via LVM), and NTP**, while cluster health is maintained through **Corosync, PCS, and DRBD** for real-time data replication. Security is enforced through a multi-layered approach involving **LDAP-based access control**, private networking, and **WireGuard VPN** encryption. Final system validation is achieved through **HPL benchmarking** and continuous observability via a **Prometheus and Grafana** monitoring stack. The resulting environment provides a scalable, self-healing, and secure platform for modern computational demands.

TABLE OF CONTENTS

1. INTRODUCTION AND OVERVIEW OF PROJECT

1.1 PURPOSE

The purpose of the project titled **"High Availability HPC Cluster"** is to design and implement a secure, highly available, and fault-tolerant High-Performance Computing (HPC) cluster that ensures uninterrupted job execution and efficient resource utilization. The project focuses on providing automated workload management with self-healing capabilities, where computational jobs are automatically migrated to passive nodes in the case of failures. By integrating high-availability mechanisms, centralized authentication, and real-time monitoring, the project aims to deliver a resilient HPC environment suitable for compute-intensive workloads while maintaining strong security, scalability, and operational reliability.

1.2 AIM AND OBJECTIVE

Aim: The primary aim of this project is to build a multi-node HPC cluster with automated job scheduling, failover, and self-healing mechanisms using standard tools and services to ensure high availability, secure access, and efficient performance.

Objective: The project seeks to achieve the following objectives:

- **HPC Cluster Implementation:**

Design and deploy a multi-node HPC cluster comprising a login node, active–passive master nodes, and active–passive worker nodes to support scalable, reliable computational workloads.

- **Centralized User Access:**

Set up a dedicated login node as the sole access point for users, ensuring secure authentication and regulated job submission.

- **Job Scheduling and Resource Management:**

Use SLURM for efficient job scheduling, workload distribution, and resource allocation across worker nodes..

- **Self-Healing and High Availability:**

Implement active–passive failover mechanisms for both master and worker nodes using **Corosync** , ensuring automatic workload migration during node failures.

- **Network and Configuration Management:**

Configure essential network services such as **DNS, DHCP, NTP, and NFS** to enable reliable communication, time synchronization, and shared storage across the cluster.

- **Secure Access and Authentication:**

Enforce user-specific access control using **LDAP**, private IP networking, firewall rules via **pfSense**, and secure remote access through **SSH**.

- **Monitoring and Performance Visualization:**

Integrate **Prometheus** for metrics collection and **Grafana** for real-time visualisation of cluster health, resource utilisation, and job performance.

- **Scalability and Resource Optimization:**

Design the cluster to efficiently utilise available hardware resources, while allowing future expansion.

By achieving these objectives, the project delivers a robust and fault-tolerant HPC infrastructure that ensures continuous availability, secure user access, and efficient execution of high-performance computing workloads

2. PREREQUISITES

2.1 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

The proposed HPC cluster is deployed using multiple interconnected machines, each assigned a specific role to ensure high availability, scalability, and fault tolerance. The hardware requirements for each machine are as follows:

1. **Machine 1 – Login Node**

(User Access, Security, and Network Services)

RAM: 64 GB

CPU: 8 Cores

Disk Space: 1 TB

Installed Services:

NTP, DHCP, DNS, LDAP, shared storage (NFS), UFW, WireGuard, SSH, and Virtual IP configuration

Purpose:

Acts as the single-entry point for users, handling authentication, secure access, and job submission.

2. **Machine 2 – Active Master Node**

(Cluster Management and Scheduling)

RAM: 64 GB

CPU: 8 Cores

Disk Space: 1 TB

Installed Services:

NTP, SLURM Controller (slurmctld), SLURM Database (slurmdbd), Corosync, rsync, NFS, Prometheus, Grafana, DRBD, PCS

Purpose:

Manages job scheduling, resource allocation, and maintains cluster state.

3. **Machine 5 – Passive Master Node**

(High Availability and Failover)

RAM: 64 GB

CPU: 8 Cores

Disk Space: 1 TB

Installed Services:

NTP, PCS, SLURM Controller, SLURM Database, Corosync, NFS

Purpose:

Takes over automatically in case of active master node failure.

4. **Machine 3 and Machine 4 – Worker Nodes**

(Job Execution Nodes)

RAM: 64 GB (each)

CPU: 8 Cores (each)

Disk Space: 1 TB (each)

Installed Services:

slurmd, NTP, NFS,UFW

Purpose:

Execute computational jobs assigned by the SLURM scheduler.

5. **Passive Worker Nodes**

(Self-Healing and Load Redistribution)

Configuration:

Identical to active worker nodes

Purpose:

Automatically handle job execution when an active worker node becomes unavailable, ensuring uninterrupted processing.

All machines are interconnected using a **private IP network through a switch**, enhancing security and reducing external exposure.

Software Requirements:

The software components used in the project enable cluster architecture, scheduling, monitoring, security, and configuration management.

- **Operating System:**

Linux-based operating system (preferably Ubuntu Server 20.04 or higher).

- **SLURM Workload Manager:**

Used for job scheduling, queue management, and resource allocation across the HPC cluster.

- **High Availability Tools:**

Corosync for synchronization and failover between active and passive master nodes.

- **Authentication and Access Control:**

LDAP for centralized, user-specific authentication and authorization.

- **Networking and Configuration Services:**

DNS, DHCP, NTP for name resolution, dynamic IP allocation, and time synchronization.

- **Storage Services:**

NFS for shared file systems across master and worker nodes.

- **Security Tools:**

UFW for host-based rule management to control allowed services**, iptables** for low-level packet filtering, port forwarding, **WireGuard** offers fast, secure VPN-based remote access to the HPC cluster and **SSH** for administrative control.

- **Monitoring and Visualization:**

Prometheus for system and performance monitoring, and **Grafana** for real-time visualization of metrics.

2.2 TOOLS AND TECHNOLOGIES

Ensure that the following tools and technologies are available and properly configured to implement the proposed HPC cluster with high availability, security, and self-healing capabilities.

A) HPC Cluster and Architecture Tools

- **SLURM Workload Manager**

Must be installed and configured on **Machine 2 (Active Master Node)** and **Machine 5 (Passive Master Node)**.

- Components:

`slurmctld` for job scheduling and cluster control.

`slurmdbd` for accounting and job history tracking.

Used to manage job submission, scheduling, and resource allocation across worker nodes.

- **Worker Node Services**

- `slurmd` must be installed on **Machine 3 and Machine 4** (active and passive worker nodes).

Responsible for executing jobs assigned by the SLURM controller.

- **High Availability and Failover Tools**

- **Corosync**:

Used to monitor master node health and manage cluster communication.

- **Pacemaker**:

It provides high availability by monitoring critical services and automatically failing them over to healthy nodes.

PCS (Pacemaker/Corosync Configuration System) :

A command-line tool used to configure, manage, and monitor Pacemaker high availability clusters.

B) Networking, Storage, and Access Tools

- **Login Node Services (Machine 1)**

DNS & DHCP: Provide name resolution and dynamic IP address allocation within the private cluster network.

NTP: Ensures time synchronization across all cluster nodes.

NFS (Shared Storage): Provides shared storage accessible by master and worker nodes.

Virtual IP (VIP): Ensures seamless failover and uninterrupted access during master node failure.

SSH: Enables secure remote administration and node access.

- **Secure Network Access**

UFW: Provides host-based firewall rule management to control allowed services.

iptables: Enables low-level packet filtering, NAT, and port forwarding.

WireGuard: Offers fast, secure VPN-based remote access to the HPC cluster.

SSH: Allows secure remote administrative access to the HPC cluster.

C) Monitoring and Performance Management

- **Prometheus**
  - Installed on **Machine 2 (Active Master Node)**.
  - Collects system-level and cluster-level metrics such as CPU usage, memory consumption, and node health.
- **Grafana**
  - Installed alongside Prometheus.
  - Provides real-time dashboards and visual representation of cluster performance and resource utilization.

D) Authentication and Access Control

- **LDAP (Lightweight Directory Access Protocol)**
  - Installed and configured on **Machine 1 (Login Node)**.
  - Provides centralized, user-specific authentication and authorisation.
  - Ensures only authorized users can access the cluster and submit jobs.
- **Private IP Networking**
  - All machines operate within a private IP range connected through a switch.
  - Prevents direct external access and enhances overall security.

E) Basic Configuration Knowledge

To successfully deploy and manage the HPC cluster, the following configuration knowledge is required:

- **DNS & DHCP Configuration**
  - For hostname resolution and automated IP address assignment.
- **NTP Configuration**
  - To maintain time consistency across all nodes, essential for cluster coordination.

- **NFS Configuration**
  - For shared file systems used during job execution.
- **SLURM Configuration**
  - Understanding job queues, partitions, and node states.
- **High Availability Concepts**
  - Knowledge of active–passive architecture and failover mechanisms.

F) Security Considerations

- **User Authentication and Authorization**

Enforce centralized access control with secure authentication for authorized users.

- **Network Security**

Use **UFW** and **iptables** firewall rules to control inbound/outbound traffic and restrict unauthorized access.

- **Secure Communication**

Ensure all administrative and inter-node access is encrypted using **WireGuard VPN** and **SSH**.

- **Cluster Coordination and Availability**

Use **Corosync** for reliable cluster messaging and node membership management.

- **Monitoring and Alerts**

Configure Prometheus alerts for node failures, high resource usage, and service downtime.

- **Data Integrity**

Implement high-availability mechanisms through Pacemaker-managed services to ensure consistent cluster operation

3.OVERALL DESCRIPTION

3.1 INTRODUCTION:

In today's data-driven and computation-intensive environments, the demand for reliable, scalable, and highly available High-Performance Computing (HPC) infrastructure has increased significantly. Traditional HPC setups often rely on static configurations and manual intervention, which can result in resource underutilization, service downtime, and delayed job execution when system failures occur. These limitations highlight the need for automated, fault-tolerant, and secure HPC architectures that can adapt dynamically to changing workloads and infrastructure conditions.

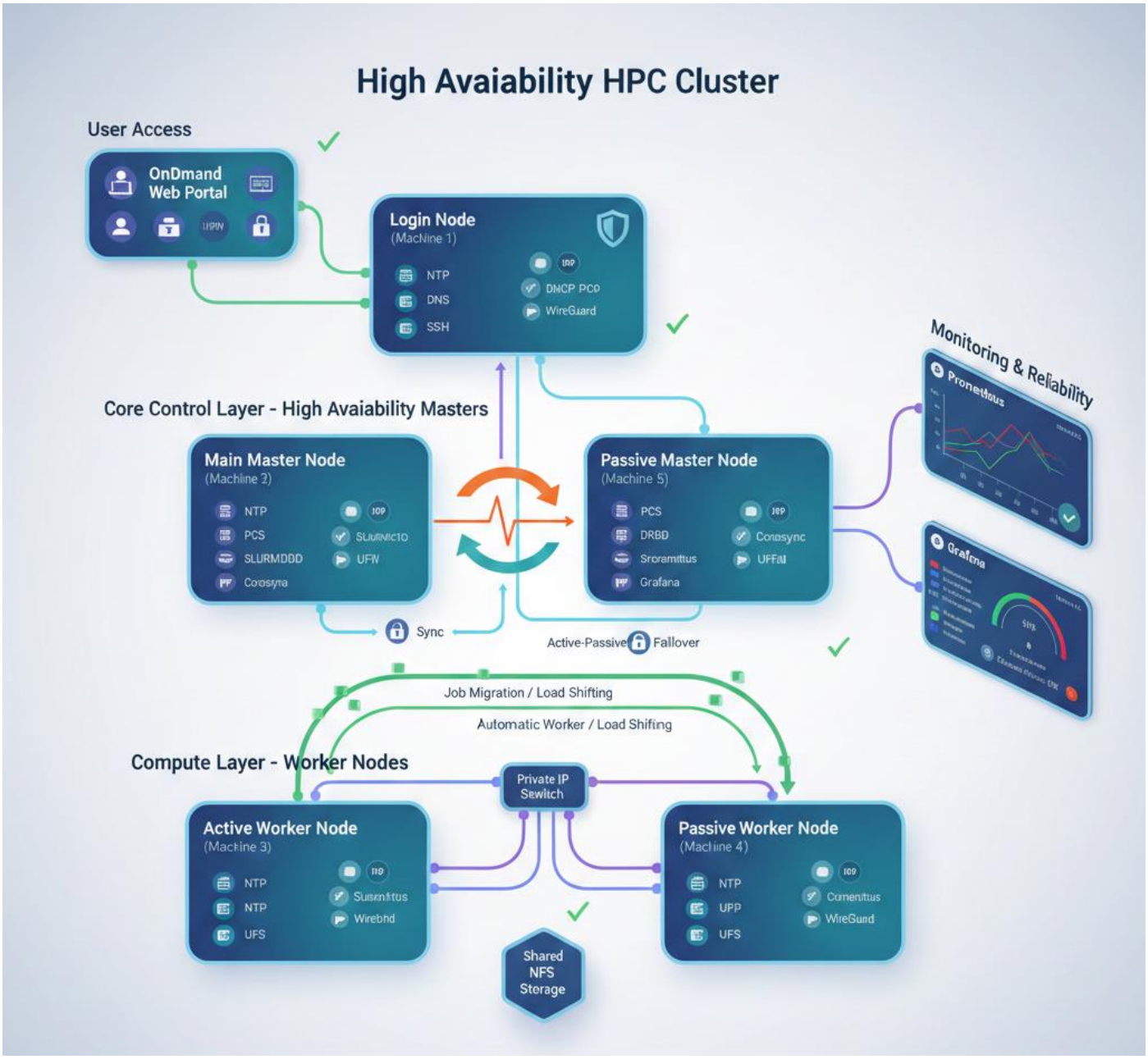
This project, titled **"High Availability HPC Cluster,"** focuses on designing and implementing a resilient HPC cluster that ensures continuous job execution through automated failover and self-healing mechanisms. The architecture is built using multiple interconnected machines, including a centralized login node, active–passive master nodes, and active–passive worker nodes. The login node serves as the single access point for users, providing secure authentication and controlled job submission, while the master nodes manage job scheduling and resource allocation using the SLURM workload manager.

To achieve high availability and fault tolerance, the project integrates PCS (Pacemaker/Corosync Configuration System) to monitor node health and synchronize critical configurations between active and passive master nodes. In the event of a master or worker node failure, workloads are automatically redirected to available passive nodes, ensuring uninterrupted computational processing. Shared storage is provided through NFS, enabling seamless data access across all nodes in the cluster.

Security and access control are key aspects of the system design. The cluster operates within a private IP network to minimize external exposure, while LDAP-based authentication enforces user-specific access control. Additional security measures, including UFW and iptables rules, WireGuard for secure remote access, and SSH for administrative control, further strengthen the system's security posture.

To maintain operational visibility and performance optimization, Prometheus and Grafana are integrated for real-time monitoring and visualization of system health, resource utilization, and job execution metrics. By combining automation, redundancy, monitoring, and strong security practices, this project delivers a scalable and self-healing HPC infrastructure.

3.2 DATA FLOW DIAGRAM



The **High Availability HPC Cluster** operates through a coordinated flow of data and control signals, now enhanced with an **Open OnDemand** web portal for streamlined user interaction and **HPL Benchmarking** for performance validation.

1. User Access & Authentication

Users no longer rely solely on terminal-based SSH. Access is initiated via an **OnDemand Web Page** hosted on the **Login Node**.

- **Credential Validation:** The web portal interfaces with **LDAP** to authenticate users.
- **File Management:** Upon logging in, users can view and manage their specific **NFS-mounted home directory** directly through the web browser, allowing for easy file uploads and script edits without command-line complexity.

2. Job Orchestration & High Availability

Once a job is submitted via the web portal or CLI:

- **Management Failover:** The **Main Master (Machine 2)** processes the request via slurmctld. **Corosync and PCS** monitor node health; if the Main Master fails, the **Passive Master (Machine 5)** assumes the Virtual IP and continues managing the queue using synchronized data from **DRBD**.
- **Compute Redundancy:** Jobs are dispatched to the **Active Worker Nodes**. If a node heartbeat is lost, SLURM re-queues the task to the **Passive Worker Nodes**, ensuring the computational process completes without manual intervention.

3. Storage and Synchronization

All nodes (Login, Masters, and Workers) communicate over a **Private Network**.

- **NFS (LVM):** Provides a unified namespace, ensuring that a user's home directory and project data are identical across all five machines.

- **NTP:** Maintains strict time synchronization, which is critical for Slurm log consistency and security certificate validation.

4. Performance Validation (HPL Benchmarking)

To verify the cluster's "High Performance" capabilities, the **High-Performance Linpack (HPL)** benchmark is deployed.

- **Execution:** HPL solves a dense system of linear equations across the distributed worker nodes using MPI (Message Passing Interface).

- **Analysis:** The results provide the actual **FLOPS** (Floating Point Operations Per Second) rating of the cluster, confirming that the high-availability configuration hasn't introduced significant latency or overhead.

- Our Cluster obtain **"3GFLOPS"** rating .

5. Monitoring

Throughout the data flow, **Prometheus** scrapes metrics from every node, while **Grafana** provides a visual real-time heat map of CPU load, memory usage, and failover status

3.3 ARCHITECTURE

The architecture of the proposed system is designed to provide a secure, highly available, and self-healing High-Performance Computing (HPC) environment. It consists of multiple interconnected nodes, each assigned a specific role to ensure efficient job execution, fault tolerance, and centralized access. The overall architecture is divided into the following key components:

Centralized User Access via Login Node

- **Single Entry Point:**

The login node acts as the primary access point for users to interact with the HPC cluster.

- **User Authentication:**

LDAP is used to authenticate users and enforce role-based access control.

- **Network Services:**

DNS and DHCP provide hostname resolution and dynamic IP allocation, while NTP ensures time synchronization across all nodes.

- **Secure Access:**

User and administrative access is secured using SSH and WireGuard, with iptables enforcing firewall rules.

- **Shared Storage Access:**

Centralized storage is configured using NFS, enabling users to store and access job-related data across the cluster.

- **Virtual IP (VIP):**

A virtual IP ensures uninterrupted access during failover scenarios.

Master Node Architecture and Job Scheduling

- **Active–Passive Master Nodes:**

The cluster uses two master nodes—one active and one passive—to maintain high availability.

- **SLURM Controller Services:**

The active master node runs slurmctld and slurmdbd to handle job scheduling, accounting, and resource management.

- **Failover Management:**

Corosync continuously monitors the health of the active master node. In case of failure, the passive master node automatically takes over control.

- **Data Synchronization:**

PCS and Corosync ensure that Slurm configurations and cluster state information remain consistent and highly available across the master nodes.

Worker Node Execution Architecture

- **Active and Passive Worker Nodes:**

Worker nodes are responsible for executing computational jobs assigned by SLURM.

- **Job Execution Service:**

The slurmd daemon runs on each worker node to manage job execution.

- **Self-Healing Mechanism:**

If an active worker node fails, SLURM automatically redistributes the workload to available passive worker nodes, ensuring uninterrupted job processing.

- **Shared Storage:**

NFS enables worker nodes to access input data and store output results in a centralized location.

Monitoring and Performance Management

- **Prometheus Integration:**

Prometheus collects system metrics such as CPU usage, memory utilization, disk I/O, and node availability across the cluster.

- **Grafana Dashboards:**

Grafana visualizes real-time performance data, providing insights into cluster health and resource utilization.

- **Failure Detection:**

Monitoring tools assist in early detection of node failures and performance bottlenecks.

Security and Network Architecture

- **Private Network Communication:**

All nodes communicate over a private IP network through a switch, minimizing external exposure.

- **Access Control:**

LDAP enforces user-specific permissions, ensuring secure and controlled access to cluster resources.

- **Secure Communication:**

SSH and WireGuard VPN tunnels protect data transmission between users and the cluster.

Fault Tolerance and High Availability

- **Master Node Failover:**

Automatic failover ensures continuous cluster control if the active master node becomes unavailable.

- **Worker Node Load Shifting:**

Job execution is dynamically shifted to passive worker nodes during failures.

- **Continuous Availability:**

The combined use of redundancy, monitoring, and automation ensures uninterrupted HPC operations.

4.HPC CLUSTER LAB SETUP

To set up the **High Availability HPC Cluster**, a multi-node laboratory environment is configured using **five machines connected through a private network**. Each machine is assigned a specific role to ensure **high availability**, **secure access**, **automated job scheduling**, and **fault tolerance**.

Machine Role Overview

Machine	Role	Purpose
Machine 1	Login Node	User access point, authentication, networking, security, and shared storage services
Machine 2	Active Master Node	SLURM job scheduling, cluster control, monitoring, and performance management
Machine 5	Passive Master Node	Standby master for failover, synchronized with active master
Machine 3 & 4	Worker Nodes	Execute HPC jobs, handle active and passive workloads

All machines are configured with **64 GB RAM, 8 CPU cores, and 1 TB storage** and communicate via a **private IP network through a switch**.

1. Machine 1: Login Node Setup

The login node acts as the central access point for users and administrators.

1.1 Network and Time Configuration

- Install and configure **NTP** to ensure time synchronization across all nodes.
- Configure **DNS** for hostname resolution.
- Configure **DHCP** to dynamically assign IP addresses within the private network.

1.2 User Authentication and Access Control

- Install and configure **LDAP** for centralized, user-specific authentication.
- Define user roles and permissions to restrict unauthorized access.
- Enable **SSH** for secure administrative and user access.

1.3 Secure Remote Access

- Configure **WireGuard VPN** for secure remote connectivity to the cluster.
- Set up **UFW and iptables** rules to control inbound and outbound traffic.
- Assign a **Virtual IP (VIP)** to ensure continuous accessibility during failover scenarios.

1.4 Storage Configuration

- Configure **NFS (LVM-based) shared storage**.
- Ensure storage is accessible to master and worker nodes for job input and output data.

2. Machines 2 & 5: Master Node Setup

2.1 SLURM Installation and Configuration

- Install **SLURM workload manager** on both machines.
- Configure:
  - **slurmctld** for job scheduling and resource management
  - **slurmdbd** for job accounting and logging
- Designate Machine 2 as **Active Master** and Machine 5 as **Passive Master**.

2.2 High Availability Configuration

- Install and configure **Corosync** to monitor master node health and manage cluster messaging.
- Use **PCS** to configure high availability and ensure synchronization between active and passive master nodes.
- Configure **DRBD** for data replication to maintain consistency of critical cluster data.
- Enable **automatic failover**, allowing the passive master to take control if the active master fails.

2.3 Shared Storage Access

- Mount **NFS shared storage** from the login node.
- Ensure consistent file access across all cluster components.

3. Machines 3 & 4: Worker Node Setup

3.1 Worker Node Services

- Install **slurmd** on each worker node.
- Configure nodes to register with the SLURM master.
- Enable **NTP** for time synchronization.
- Mount **NFS shared storage**.
- Apply **UFW** rules for host-based security.

3.2 Self-Healing Configuration

- Define worker nodes as **active and passive** in SLURM configuration.
- Configure SLURM to automatically reschedule jobs to available passive worker nodes in case of failure.

4. Monitoring and Performance Management Setup

4.1 Prometheus Installation

- Install **Prometheus** on the active master node.
- Configure exporters to collect:
  - CPU usage
  - Memory utilization
  - Disk I/O
  - Node availability

4.2 Grafana Dashboard Setup

- Install **Grafana** alongside Prometheus.
- Create dashboards to visualize:
  - Cluster health

- Job execution performance
- Resource utilization

5. Cluster Integration and Validation

5.1 Node Connectivity Validation

- Verify communication between login, master, and worker nodes.
- Ensure **private IP networking** is functioning correctly.

5.2 Job Submission Testing

- Submit sample HPC jobs from the login node.
- Verify successful execution on worker nodes.

5.3 Failover Testing

- Simulate failure scenarios:
  - Active master node → passive master takes control
  - Active worker node → jobs shift to passive worker
- Validate uninterrupted job execution.

6. Final Setup and Verification

- Confirm **LDAP authentication** for all users.
- Validate secure access using **SSH and WireGuard VPN**.
- Monitor real-time cluster performance using **Prometheus and Grafana**.
- Ensure storage consistency across nodes.
- Document system logs and performance metrics.

5. WORKING

This section explains the working of the **High Availability HPC Cluster**, detailing how different nodes, services, and technologies interact to provide high availability, fault tolerance, secure access, and efficient job execution.

5.1 Login Node – User Access and Control

Machine 1 serves as the single entry point for all users accessing the HPC cluster.

Working:

- Users connect securely to the cluster using **SSH** or **WireGuard VPN**, ensuring encrypted communication.
- **LDAP** authenticates users and enforces user-specific access control.
- **DNS** and **DHCP** dynamically manage hostname resolution and IP address assignment.
- **NTP** synchronizes time across all cluster nodes, ensuring accurate job scheduling and logging.
- **NFS (LVM-based) shared storage** allows users to access job scripts, datasets, and results.
- **UFW**, **iptables**, and **private IP networking via a switch** isolate the cluster from public networks, enhancing security.
- A **Virtual IP (VIP)** ensures uninterrupted access even during failover.

Role in the System:

- Centralized user authentication and access control
- Secure gateway to the HPC environment
- Shared storage and configuration management

5.2 Source Code and Job Management Workflow

In this HPC environment, users submit **job scripts and workloads** rather than application source code.

Working:

- Users submit jobs from the **Login Node** using SLURM commands.

- Job scripts and input data are stored on **shared NFS storage**, accessible by all compute nodes.
- **LDAP** ensures that users can only access their authorized jobs and files.

Role in the System:

- Centralized job submission
- Secure and controlled workload management
- Consistent access to job data across the cluster

5.3 Master Nodes – Job Scheduling and Control

The cluster consists of **two master nodes**:

Machine	Role
Machine 2	Active (Main) Master
Machine 5	Passive (Standby) Master

Working:

- The **Active Master Node** runs:
  - **slurmctld** for job scheduling
  - **slurmdbd** for job accounting and logging
- **PCS and Corosync** maintain synchronization between the active and passive master.
- **DRBD** replicates critical configuration and state data between masters.
- If the active master fails, the passive master automatically takes over, ensuring uninterrupted job scheduling.
- **NFS** provides consistent access to job files and logs.
- **NTP** maintains time synchronization across masters and workers.

Role in the System:

- Central job scheduling and resource allocation
- High availability through active–passive failover

- Reliable job tracking and accounting

5.4 Worker Nodes – Job Execution

The cluster uses multiple worker nodes for computational workloads:

Working:

- Each worker node runs **slurmd**, which communicates with the master node.
- Jobs assigned by the master are executed on available worker nodes.
- **NFS** provides access to job data and output directories.
- **NTP** ensures synchronized execution timing.
- **UFW** rules provide host-level security.
- If an active worker node fails, **SLURM automatically redirects jobs** to a passive worker node.
- This ensures minimal or zero disruption in job execution.

Role in the System:

- High-performance job execution
- Automatic workload redistribution
- Fault tolerance and reliability

5.5 Self-Healing and Failover Mechanism

Working:

- **SLURM** continuously monitors node health.
- On detecting a failure:
  - Jobs running on failed worker nodes are reassigned.
  - Passive worker nodes become active automatically.
- If the active master fails:
  - The passive master takes control using **Corosync and PCS**.
  - Job scheduling resumes without user intervention.
- The login node remains continuously accessible via the **Virtual IP**.

Role in the System:

- Automatic fault detection and recovery
- High availability of compute resources
- Reliable long-running job execution

5.6 Monitoring – Prometheus

Working:

- Deployed on the **Active Master Node**.
- Collects metrics such as:
  - CPU utilization
  - Memory usage
  - Node availability
  - Job execution status
- Stores metrics as **time-series data**.
- Alerts can be configured for node failures, high resource usage, or service downtime.

Role in the System:

- Real-time performance monitoring
- Early detection of failures
- Infrastructure observability

5.7 Grafana – Visualization and Alerts

Working:

- Integrates with Prometheus to display dashboards for:
  - Cluster resource usage
  - Worker node performance
  - Job execution statistics
- Enables alerts via email or other notification channels.

- Assists administrators in trend analysis and resource optimization.

Role in the System:

- Visual performance analysis
- Proactive issue detection
- Operational decision support

5.8 Security and Network Architecture

Working:

- Cluster operates entirely on **private IP addresses** via a switch.
- Access is controlled through:
  - **LDAP authentication**
  - **UFW and iptables** rules
  - **WireGuard VPN** tunnels
- **SSH access** is limited to authenticated users.
- **DNS, DHCP, NFS, and NTP** ensure stable and secure configuration management.

Role in the System:

- Secure user access
- Controlled internal communication
- Protection against unauthorized usage

5.9 Overall System Workflow Summary

1. User logs in through the **Login Node** via SSH or WireGuard.
2. Authentication is verified via **LDAP**.
3. Jobs are submitted using **SLURM** commands.
4. The **Active Master** schedules jobs to active workers.
5. Passive nodes automatically take over in case of failures.
6. **Prometheus and Grafana** monitor performance continuously.
7. Job results are stored and accessed via **shared NFS storage**.

IMPLEMENTATION & OUTPUT

Host file:

```
127.0.0.1       localhost
127.0.1.1       acts-DIT400TR-55L

# PHYSICAL IPs (Crucial for Corosync/Pacemaker to find each other)
192.168.100.2   MasterNode.HPCCluster.com   MasterNode
192.168.100.5   PassiveMaster.HPCCluster.com PassiveMaster

# VIRTUAL IP (The Floating HA Name)
192.168.100.10   SlurmVIP SlurmVIP.HPCCluster.com

# WORKER NODES
192.168.100.3   WorkerNode1
192.168.100.4   WorkerNode2
192.168.100.6   PassiveWorker
# BEGIN ANSIBLE MANAGED CLUSTER BLOCK
192.168.100.2 MasterNode
192.168.100.3 WorkerNode1
192.168.100.4 WorkerNode2
192.168.100.5 PassiveMaster
192.168.100.6 PassiveWorker
# END ANSIBLE MANAGED CLUSTER BLOCK
~
~
~
~
~
~
```

DNS Configuration:

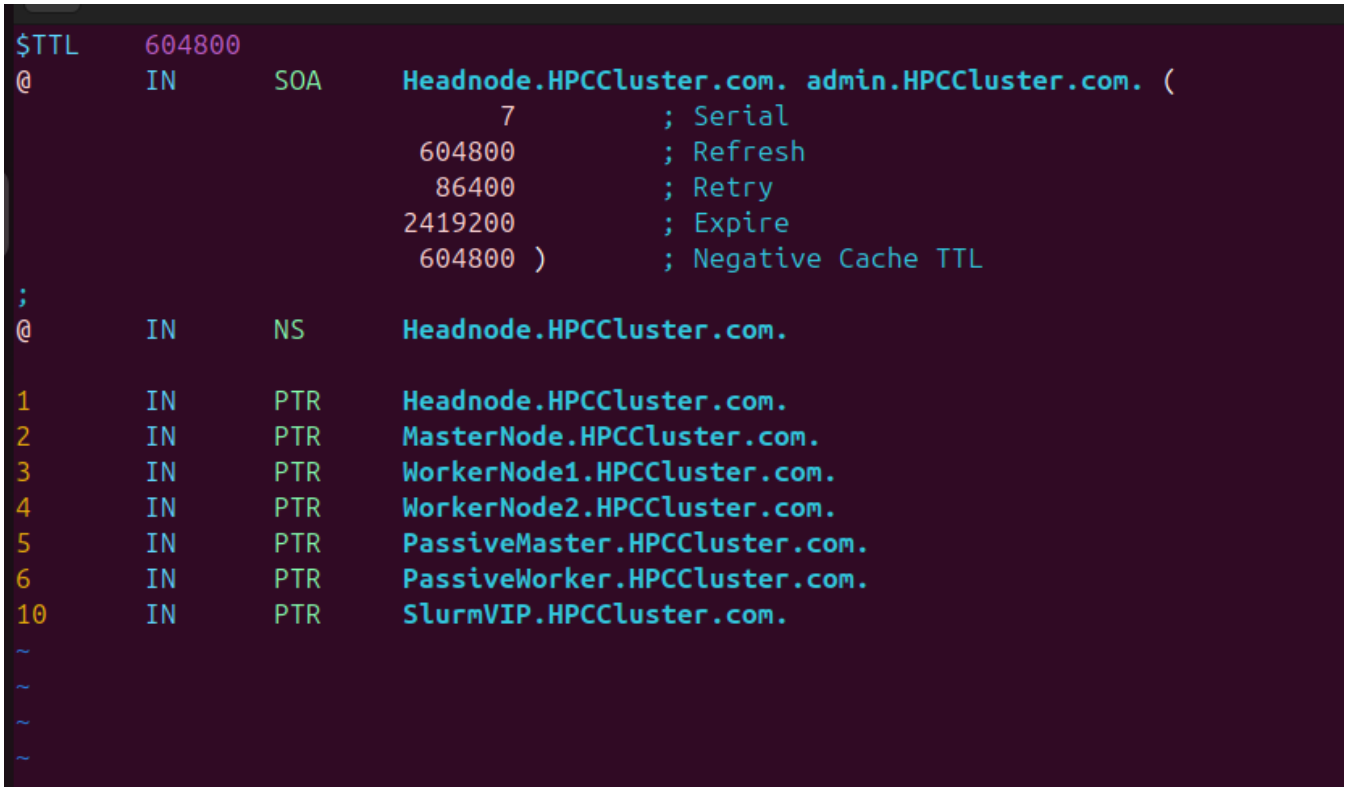
Named.conf:

```
//
// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";


// Forward Zone: Name to IP
zone "HPCCluster.com" {
    type master;
    file "/etc/bind/db.forward";
};

// Reverse Zone: IP to Name
zone "100.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.reverse";
};
~
~
~
~
~
```

Forward file:

```
$TTL    604800
@       IN      SOA     Headnode.HPCCluster.com. admin.HPCCluster.com. (
                              7         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      Headnode.HPCCluster.com.

Headnode        IN      A       192.168.100.1
MasterNode      IN      A       192.168.100.2
WorkerNode1     IN      A       192.168.100.3
WorkerNode2     IN      A       192.168.100.4
PassiveMaster   IN      A       192.168.100.5
PassiveWorker   IN      A       192.168.100.6
SlurmVIP        IN      A       192.168.100.10
~
~
~
```

Reverse file:



```
$TTL    604800
@       IN      SOA     Headnode.HPCCluster.com. admin.HPCCluster.com. (
                              7         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      Headnode.HPCCluster.com.

1       IN      PTR     Headnode.HPCCluster.com.
2       IN      PTR     MasterNode.HPCCluster.com.
3       IN      PTR     WorkerNode1.HPCCluster.com.
4       IN      PTR     WorkerNode2.HPCCluster.com.
5       IN      PTR     PassiveMaster.HPCCluster.com.
6       IN      PTR     PassiveWorker.HPCCluster.com.
10      IN      PTR     SlurmVIP.HPCCluster.com.
~
~
~
~
```

Slurm.conf:

```
# slurm.conf file for HPCCluster
ClusterName=hpc_cluster

# Update this line to use the VIP name
SlurmctldHost=MasterNode.HPCCluster.com(192.168.100.2)
SlurmctldHost=PassiveMaster.HPCCluster.com(192.168.100.5)
#SlurmctldHost=SlurmVIP.HPCCluster.com(192.168.100.10)
#ControlMachine=SlurmVIP.HPCCluster.com
#SlurmctldHost=SlurmVIP(192.168.100.10)
SlurmctldAddr=192.168.100.10


# ADD THIS LINE to allow the service to start even if hostname doesn't match
SlurmctldParameters=ignore_hostnames

# How long the controller waits for a compute node
SlurmctldTimeout=60

# How long compute nodes wait before giving up on a controller
SlurmdTimeout=300
MessageTimeout=5
InactiveLimit=0

# Job Survival Policy
JobRequeue=1
ReturnToService=2
RequeueExit=1
KillWait=30
CompleteWait=0

# DAEMONS & PATHS
SlurmctldPidFile=/var/run/slurmctld.pid
SlurmdPidFile=/var/run/slurmd.pid
SlurmdSpoolDir=/var/spool/slurmd
SlurmUser=slurm
StateSaveLocation=/var/spool/slurm/ctld
ProctrackType=proctrack/pgid

# SCHEDULING & LOGGING
SchedulerType=sched/backfill
SelectType=select/cons_tres
SlurmctldLogFile=/var/log/slurm/slurmctld.log
SlurmdLogFile=/var/log/slurm/slurmd.log
# ACCOUNTING
AccountingStorageType=accounting_storage/slurmdbd
AccountingStorageHost=SlurmVIP.HPCCluster.com
AccountingStorageUser=slurm
AccountingStoragePort=6819

# COMPUTE NODES
NodeName=WorkerNode1 NodeAddr=192.168.100.3 CPUs=16 State=UNKNOWN
NodeName=WorkerNode2 NodeAddr=192.168.100.4 CPUs=16 State=UNKNOWN
NodeName=PassiveWorker NodeAddr=192.168.100.6 CPUs=16 State=UNKNOWN

# PARTITIONS
PartitionName=debug Nodes=ALL Default=YES MaxTime=INFINITE State=UP
~
```

SlurmDBD.conf:

```
# slurmdbd.conf — HA safe

# LOGGING
DebugLevel=info
LogFile=/var/log/slurm/slurmdbd.log
PidFile=/var/run/slurmdbd.pid

# AUTH
AuthType=auth/munge
SlurmUser=slurm
MessageTimeout=10

# CONTROLLER IDENTITY (CRITICAL)
#DbdHost=SlurmVIP
DbdHost=PassiveMaster.HPCCluster.com
DbdBackupHost=MasterNode.HPCCluster.com
DbdAddr=192.168.100.10
DbdPort=6819

# DATABASE BACKEND
StorageType=accounting_storage/mysql
#StorageParameters=socket=/run/mysqld/mysqld.sock
StorageHost=192.168.100.10
StoragePort=3306
StorageUser=slurm
StoragePass=1234
StorageLoc=slurm_acct_db

# OPTIONAL ARCHIVING
=yes
ArchiveJobs=yes
ArchiveSteps=no
ArchiveUsage=no

~
~
```

Text Editor

PCS Working:

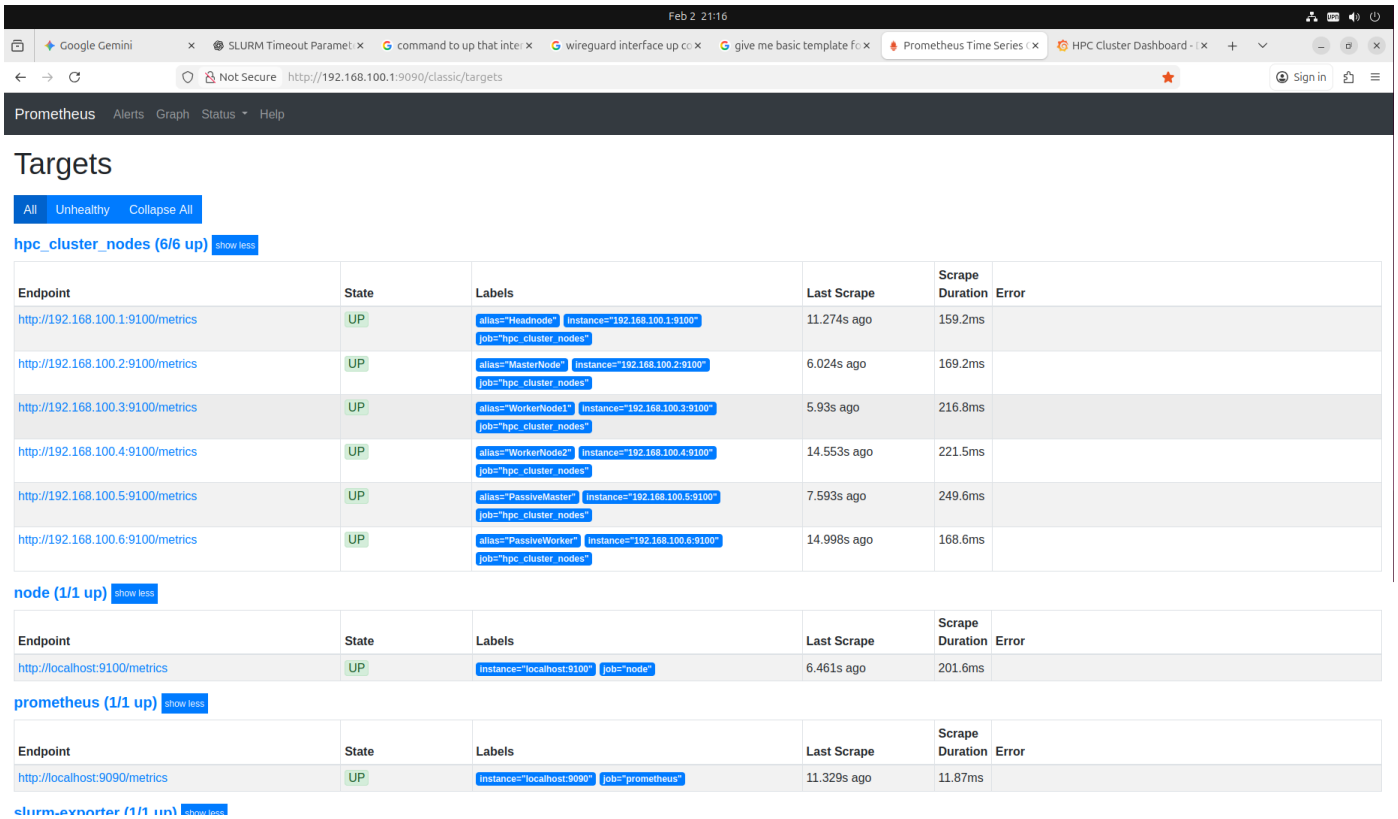
On Main Master:

```
root@PassiveMaster:/etc/drbd.d# pcs status
Cluster name: SlurmHA
Cluster Summary:
  * Stack: corosync (Pacemaker is running)
  * Current DC: PassiveMaster.HPCCluster.com (version 2.1.6-6fdc9deea29) - partition with quorum
  * Last updated: Mon Feb  2 20:27:19 2026 on PassiveMaster.HPCCluster.com
  * Last change:  Mon Feb  2 20:11:38 2026 by root via cibadmin on PassiveMaster.HPCCluster.com
  * 2 nodes configured
  * 7 resource instances configured

Node List:
  * Online: [ MasterNode.HPCCluster.com PassiveMaster.HPCCluster.com ]

Full List of Resources:
  * Clone Set: promotable-meta [slurm_data_res] (promotable):
    * Promoted: [ MasterNode.HPCCluster.com ]
    * Unpromoted: [ PassiveMaster.HPCCluster.com ]
  * mariadb     (systemd:mariadb):       Started MasterNode.HPCCluster.com
  * slurm_fs    (ocf:heartbeat:Filesystem):      Started MasterNode.HPCCluster.com
  * Resource Group: slurm_group:
    * virtual_ip        (ocf:heartbeat:IPaddr2):         Started MasterNode.HPCCluster.com
    * slurmdbd_res      (systemd:slurmdbd):      Started MasterNode.HPCCluster.com
    * slurm_ctld_res    (systemd:slurmctld):     Started MasterNode.HPCCluster.com

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
root@PassiveMaster:/etc/drbd.d#
```

On Passive Master:

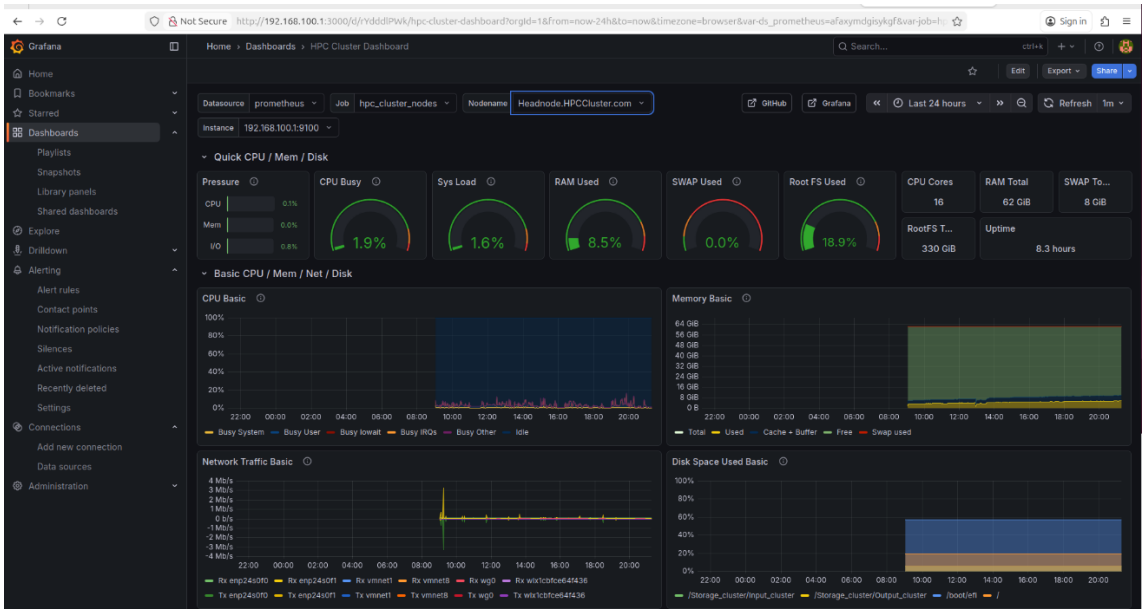
```
root@PassiveMaster:~# pcs status
Cluster name: SlurmHA
Cluster Summary:
  * Stack: corosync (Pacemaker is running)
  * Current DC: PassiveMaster.HPCCluster.com (version 2.1.6-6fdc9deea29) - partition with quorum
  * Last updated: Sat Jan 31 11:21:03 2026 on PassiveMaster.HPCCluster.com
  * Last change:  Sat Jan 31 11:08:49 2026 by hacluster via crmd on MasterNode.HPCCluster.com
  * 2 nodes configured
  * 7 resource instances configured

Node List:
  * Online: [ PassiveMaster.HPCCluster.com ]
  * OFFLINE: [ MasterNode.HPCCluster.com ]

Full List of Resources:
  * Clone Set: promotable-meta [slurm_data_res] (promotable):
    * Promoted: [ PassiveMaster.HPCCluster.com ]
    * Stopped: [ MasterNode.HPCCluster.com ]
  * Resource Group: slurm_group:
    * slurm_fs  (ocf:heartbeat:Filesystem):      Started PassiveMaster.HPCCluster.com
    * virtual_ip        (ocf:heartbeat:IPaddr2):         Started PassiveMaster.HPCCluster.com
    * slurmdbd_res      (systemd:slurmdbd):      Started PassiveMaster.HPCCluster.com
    * slurm_ctld_res    (systemd:slurmctld):     Started PassiveMaster.HPCCluster.com
  * mariadb     (systemd:mariadb):       Started PassiveMaster.HPCCluster.com

Daemon Status:
  corosync: active/enabled
  pacemaker: active/enabled
  pcsd: active/enabled
root@PassiveMaster:~# df -Th
Filesystem     Type      Size  Used Avail Use% Mounted on
tmpfs          tmpfs     6.3G  2.5M  6.3G   1% /run
/dev/nvme0n1p5 ext4      330G   22G  292G   7% /
tmpfs          tmpfs      32G   48M   32G   1% /dev/shm
tmpfs          tmpfs     5.0M   12K  5.0M   1% /run/lock
efivarfs       efivarfs  512K   50K  458K  10% /sys/firmware/efi/efivars
/dev/nvme0n1p1 vfat       96M   55M   42M  58% /boot/efi
tmpfs          tmpfs     6.3G  112K  6.3G   1% /run/user/1000
tmpfs          tmpfs     6.3G   76K  6.3G   1% /run/user/0
/dev/drbd0     ext4      458G  182M  434G   1% /var/spool/slurm
root@PassiveMaster:~#
```

Prometheus Working and metrics:

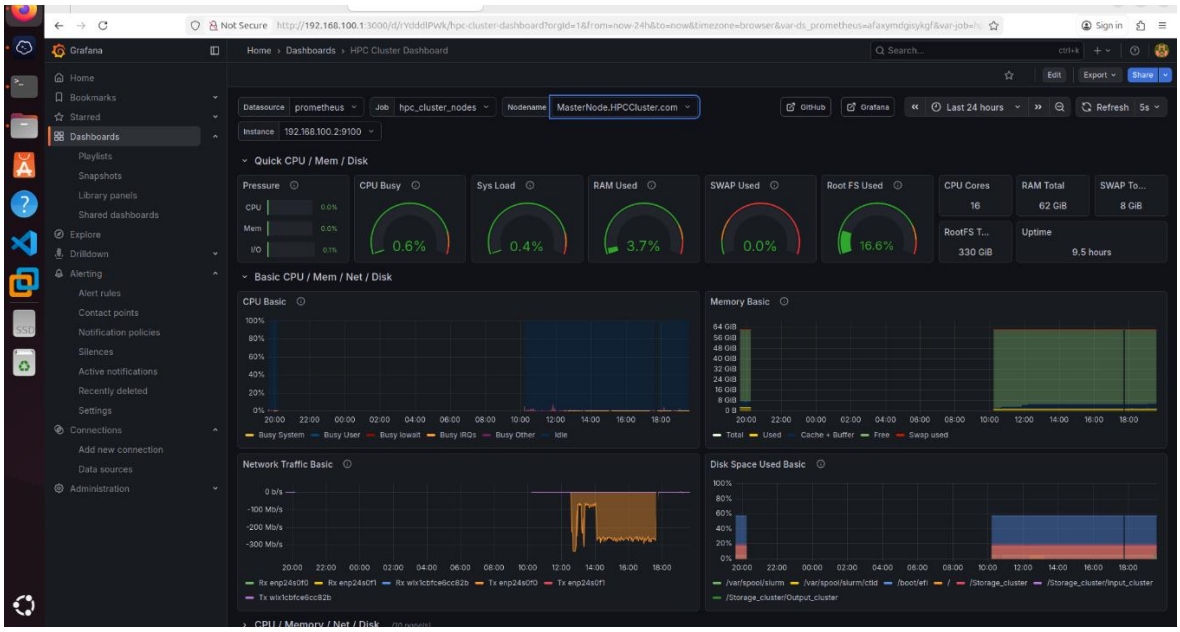


Grafana Working:

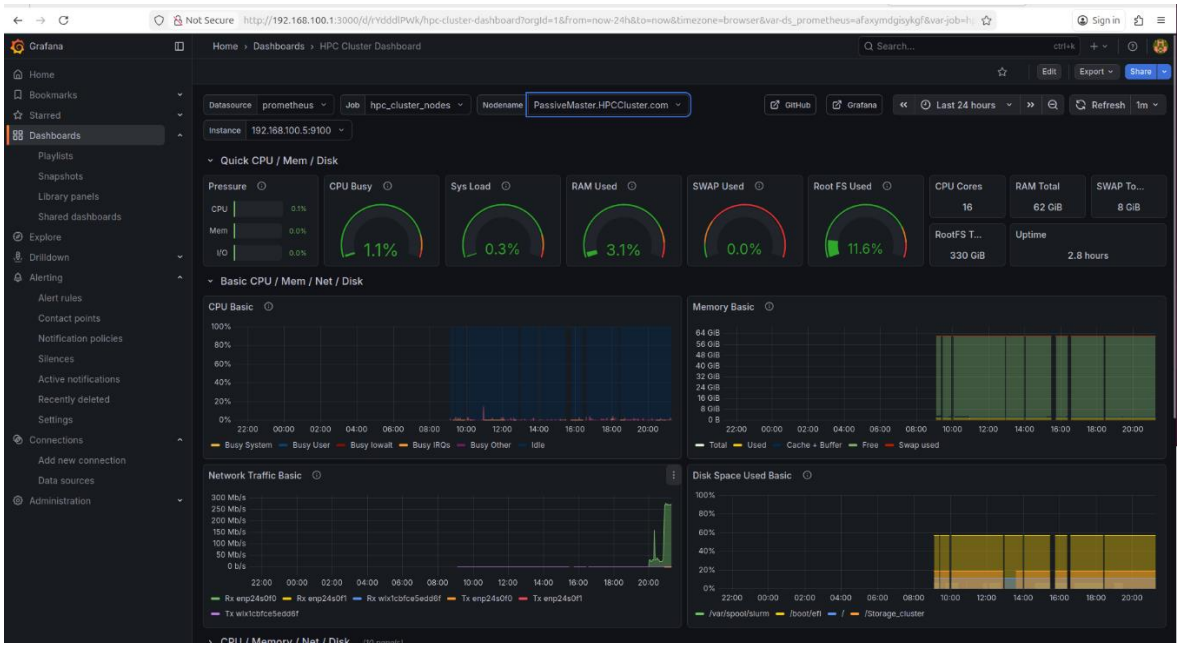
- ❖ Grafana dashboard in normal state:
  - Head node/Login node:



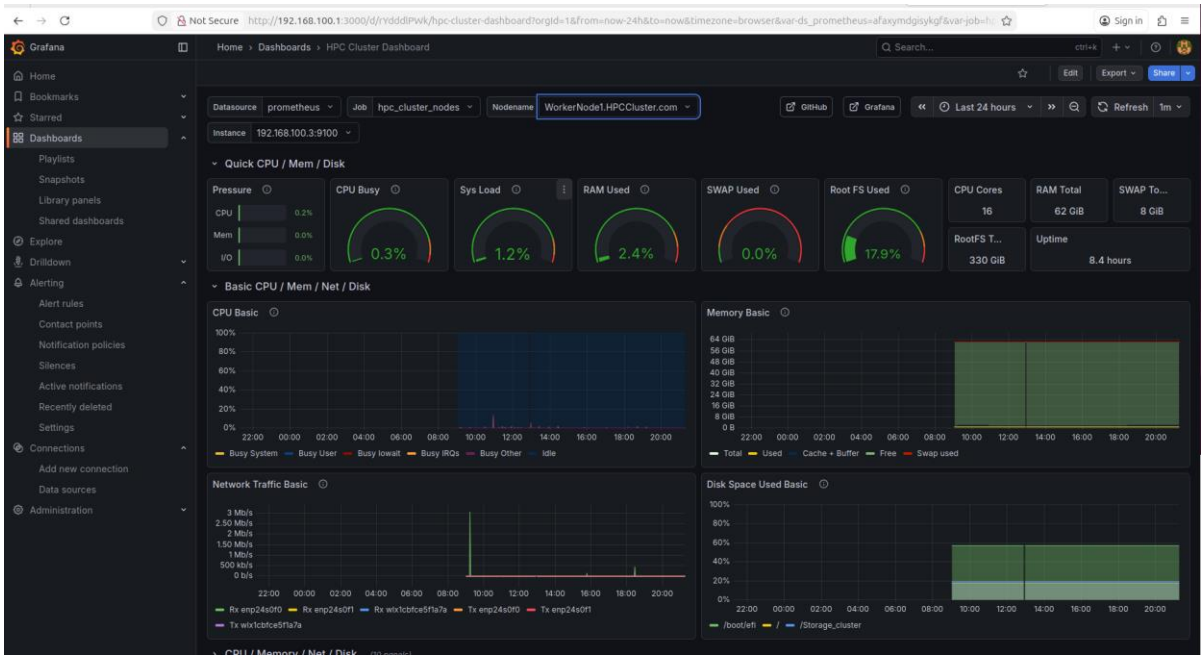
- Main Masternode:



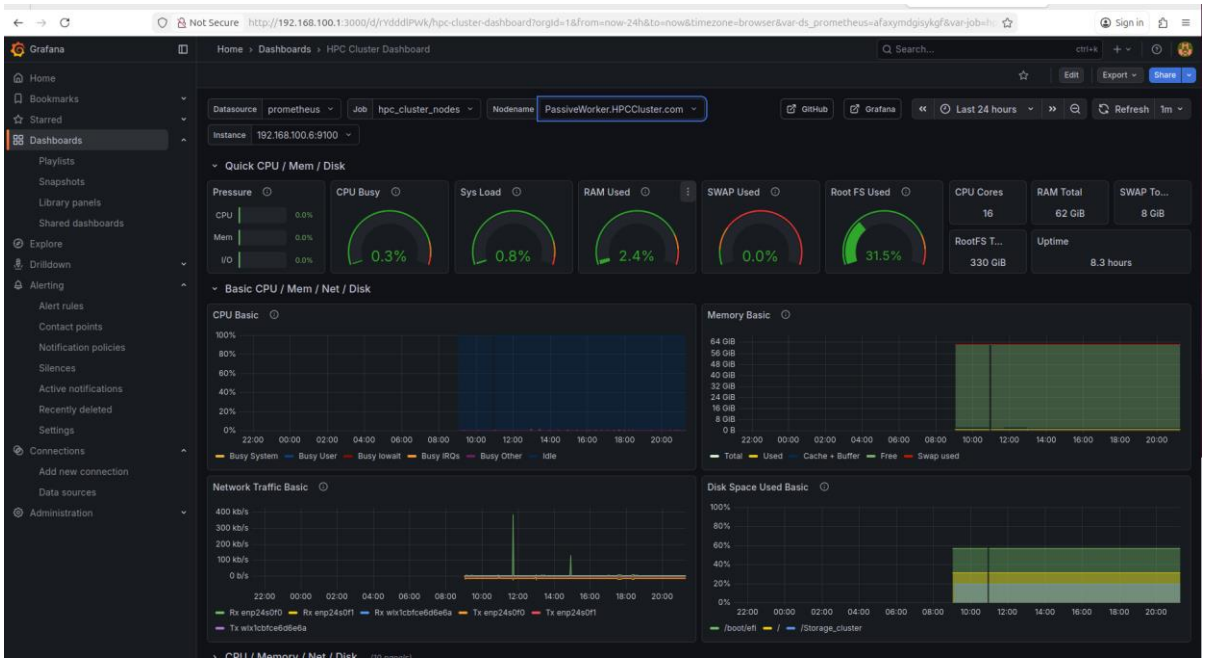
- Passive Master:



- Worker Node 1:

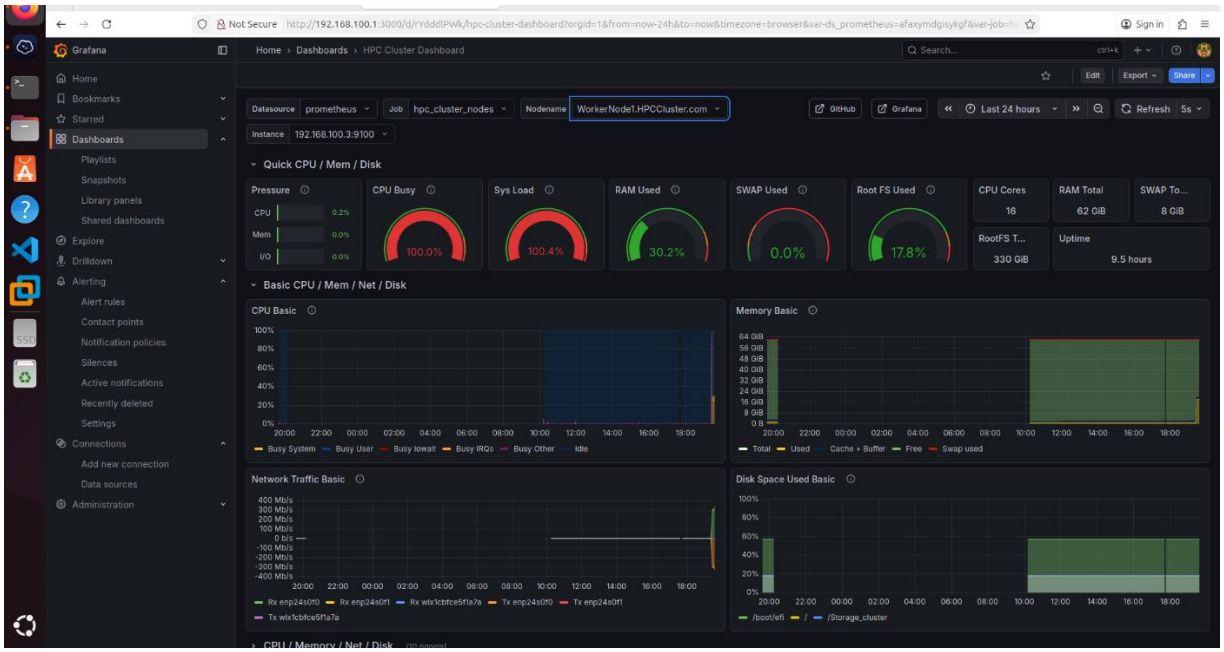


- Passive Node:

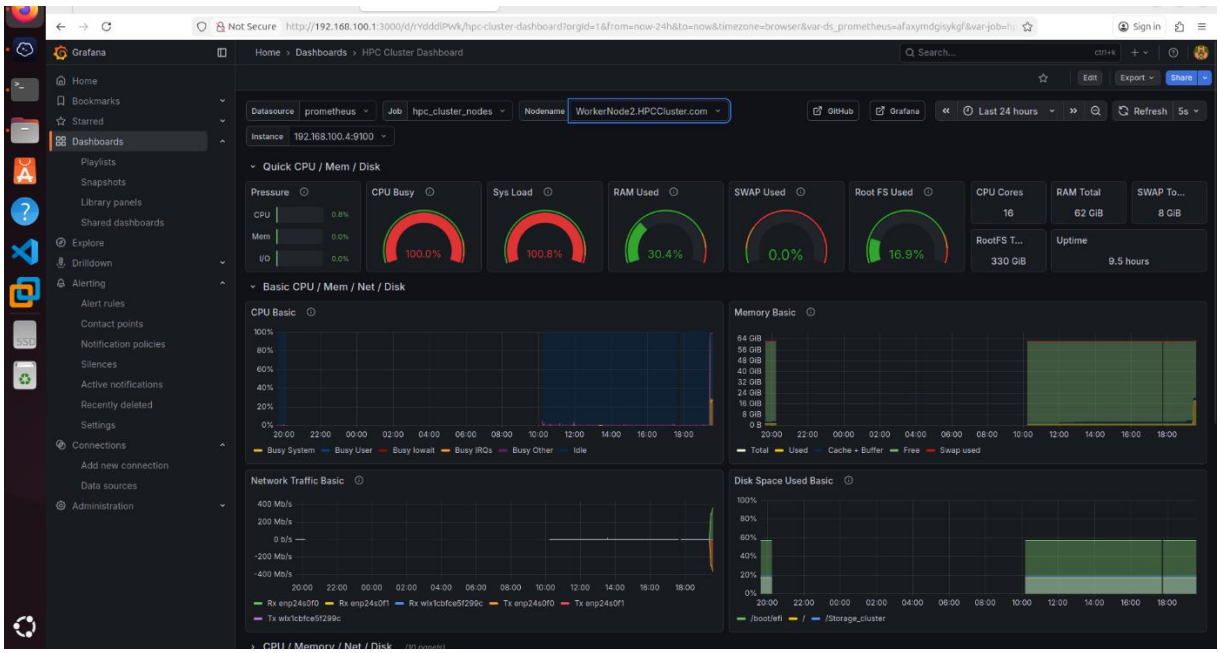


❖ Grafana dashboard during HPL Benchmarking:

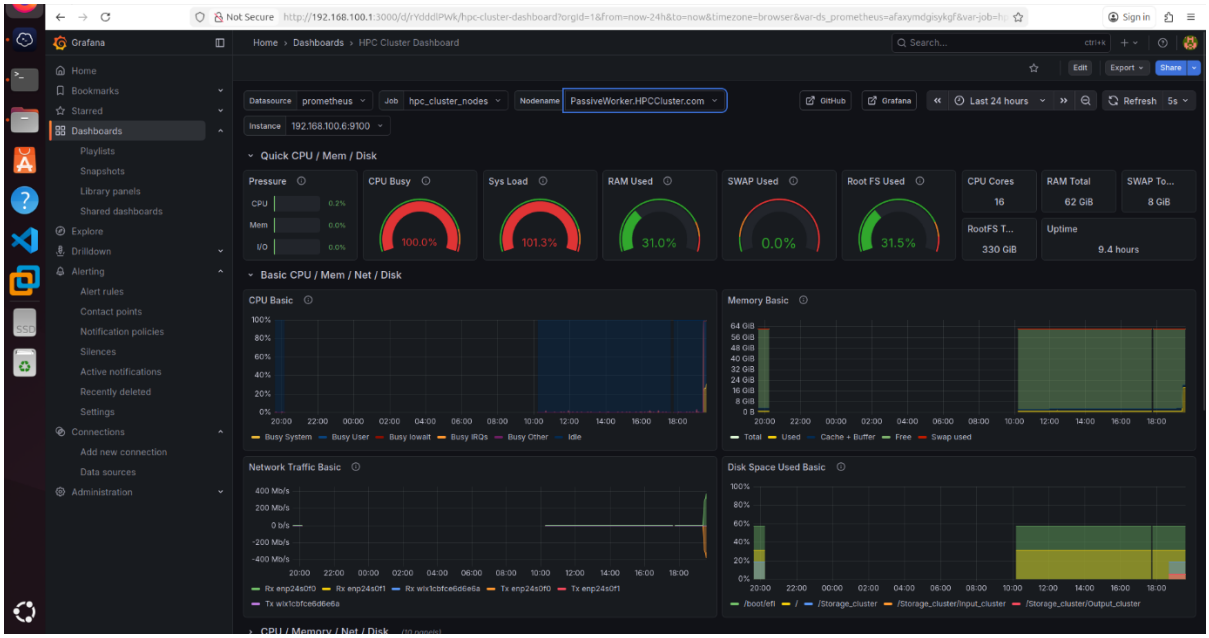
- Worker Node 1:



- Worker Node 2:



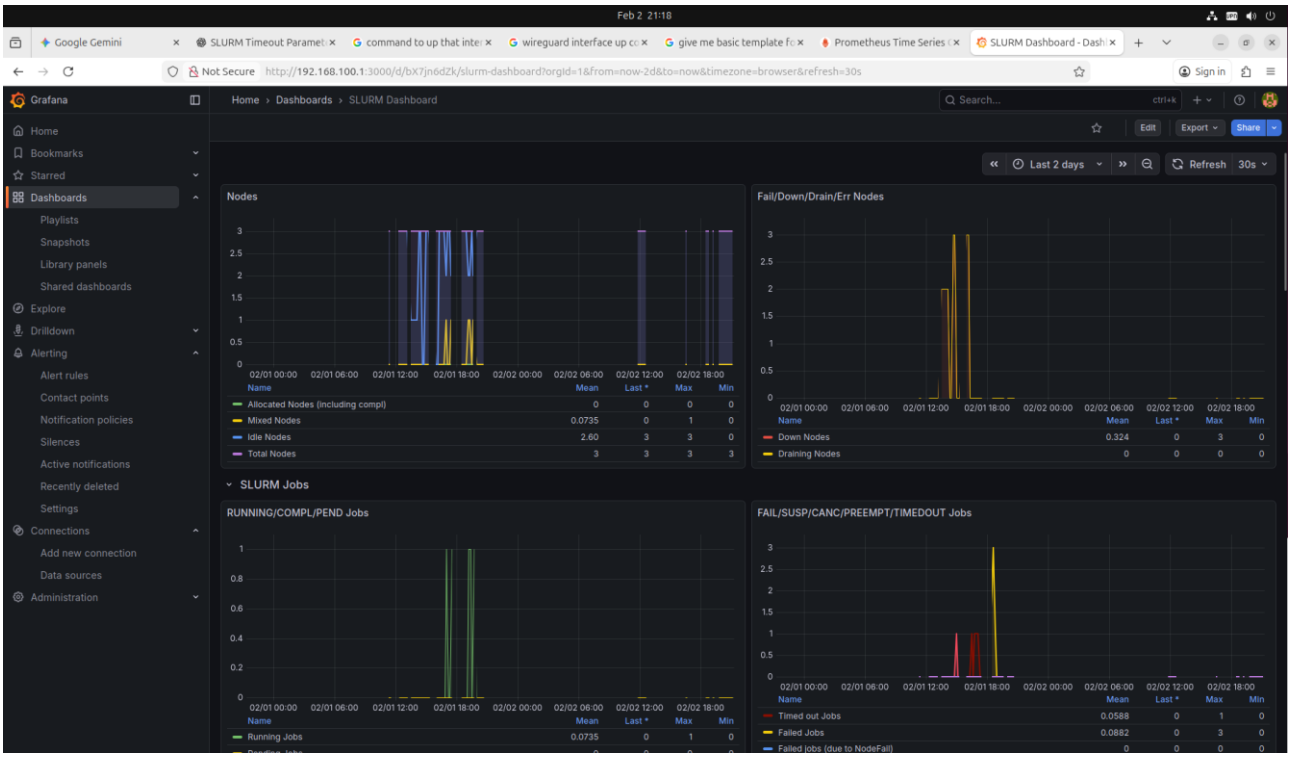
- Passive Node:



- HPL Benchmarking Result:

```
================================================================================
HPLinpack 2.3  --  High-Performance Linpack benchmark  --   December 2, 2018
Written by A. Petitet and R. Clint Whaley,  Innovative Computing Laboratory, UTK
Modified by Piotr Luszczek, Innovative Computing Laboratory, UTK
Modified by Julien Langou, University of Colorado Denver
================================================================================

An explanation of the input/output parameters follows:
T/V    : Wall time / encoded variant.
N      : The order of the coefficient matrix A.
NB     : The partitioning blocking factor.
P      : The number of process rows.
Q      : The number of process columns.
Time   : Time in seconds to solve the linear system.
Gflops : Rate of execution for solving the linear system.

The following parameter values will be used:

N      :   80000
NB     :     192
PMAP   : Row-major process mapping
P      :       6
Q      :       8
PFACT  :   Right
NBMIN  :       4
NDIV   :       2
RFACT  :   Crout
BCAST  :  1ringM
DEPTH  :       1
SWAP   : Mix (threshold = 64)
L1     : transposed form
U      : transposed form
EQUIL  : yes
ALIGN  : 8 double precision words

--------------------------------------------------------------------------------

- The matrix A is randomly generated for each test.
- The following scaled residual check will be computed:
      ||Ax-b||_oo / ( eps * ( || x ||_oo * || A ||_oo + || b ||_oo ) * N )
- The relative machine precision (eps) is taken to be               1.110223e-16
- Computational tests pass if scaled residuals are less than                16.0

================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR11C2R4       80000   192     6     8             925.91              3.6865e+02
HPL_pdgesv() start time Fri Jan 30 19:24:48 2026

HPL_pdgesv() end time   Fri Jan 30 19:40:13 2026

--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=    1.33652267e-03 ...... PASSED
================================================================================

Finished      1 tests with the following results:
              1 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
```

❖ Grafana Slurm dashboard:



CONCLUSION

The **High Availability HPC Cluster** project successfully demonstrates the design and implementation of a resilient, secure, and fault-tolerant high-performance computing environment.

The **Login Node** serves as a centralized access point, integrating key services including **LDAP** for user authentication, **DNS** and **DHCP** for network management, **NTP** for time synchronization, and **NFS/LVM-based shared storage**. Secure remote access is enforced through **SSH**, **WireGuard VPN**, **UFW**, **iptables**, private IP networking, and a **Virtual IP**, ensuring strong isolation and robust protection against unauthorized access.

SLURM effectively manages job scheduling and resource allocation across the cluster, while **PCS**, along with **DRBD**, provide high availability and seamless failover for master nodes. The integration of **Prometheus** and **Grafana** enables real-time performance monitoring and visualization, providing early detection of failures, resource bottlenecks, and system health anomalies. This comprehensive monitoring framework facilitates proactive maintenance, optimal resource utilization, and efficient operational management.

Overall, this project highlights the successful implementation of **automation, redundancy, secure access, and observability** in an HPC environment. The cluster provides **self-healing job access**, centralized control, and consistent storage management, delivering high availability, scalability, and reliability. The architecture aligns with modern HPC infrastructure and DevOps best practices, empowering users to focus on compute-intensive tasks while ensuring that failures are handled transparently and efficiently.

In conclusion, the **High Availability HPC Cluster** represents a robust and scalable solution for academic and enterprise computational workloads, combining fault tolerance, security, and operational efficiency to meet the demands of modern high-performance computing applications.

FUTURE SCOPE

The current implementation of the High Availability HPC Cluster provides a robust foundation for fault tolerance and secure computational access. To further evolve the system's capabilities, the following enhancements are proposed for future development:

7.1 Advanced Checkpointing and Distributed Computing

To protect long-running computational jobs from unexpected interruptions, dedicated checkpointing tools will be integrated.

- DMTCP Implementation: Utilizing Distributed Multi-Threaded Checkpointing to periodically save the state of running applications to the NFS-shared LVM storage.
- Fault Recovery: Enabling jobs to resume from the last saved state on a passive worker node if a primary node fails, rather than restarting the task from the beginning.

Resource Optimization: Improving the overall throughput of the cluster by minimizing wasted CPU cycles on failed processes**.**

7.2 Cloud-Based HPC Expansion (Hybrid Infrastructure)

To handle "burst" workloads that exceed the capacity of the current five-node on-premise setup, a hybrid cloud architecture is envisioned.

- Platform Integration: Extending the local Slurm controller to manage instances on AWS, Google Cloud, or Azure.
- Dynamic Scaling: Utilizing cloud elasticity to spin up virtual compute nodes during peak demand and de-provisioning them during idle periods to optimize costs.
- Disaster Recovery: Using cloud storage as a secondary site for DRBD data replication to ensure business continuity in case of total site failure.

7.3 Enhanced Alerting and Notification Mechanisms

To transition from reactive to proactive cluster management, the monitoring framework (Prometheus and Grafana) will be integrated with automated notification protocols.

- Real-time Alerts: Implementation of SMTP and SMS gateway integration to notify administrators of critical hardware or software failures instantly.
- Event Monitoring: Granular tracking of node heartbeats, Slurm job scheduling bottlenecks, and unauthorized security access attempts.
- Operational Efficiency: Reducing Mean Time to Repair (MTTR) by providing actionable insights through automated incident reports.

REFERENCES

1. Slurm Workload Manager SchedMD LLC. https://slurm.schedmd.com/documentation.html
2. ClusterLabs (Corosync & Pacemaker) ClusterLabs Project. https://clusterlabs.org/pacemaker/doc/
3. Prometheus Ecosystem Prometheus Authors. https://prometheus.io/docs/introduction/overview/
4. Grafana Labs Grafana Documentation. https://grafana.com/docs/grafana/latest/
5. OpenLDAP Project The OpenLDAP Foundation. https://www.openldap.org/doc/admin26/
6. The Linux Kernel Archives Linux Kernel Organization. https://docs.kernel.org/admin-guide/nfs/index.html
7. WireGuard VPN Protocol Jason A. Donenfeld. https://www.wireguard.com/quickstart/