Comparing Fault-tolerance in Kubernetes and Slurm in HPC Infrastructure

Mirac Aydin, Michael Bidollahkhani, Julian M. Kunkel

GWDG, University of Göttingen, Germany

DECICE Project, Horizon Europe

The Eighteenth International Conference on Advanced Engineering Computing and Applications in Sciences

Table of Content

- Motivation
- Background on Cluster Faults
- Architectures
- Methodology
- Results
- Conclusion

Motivation

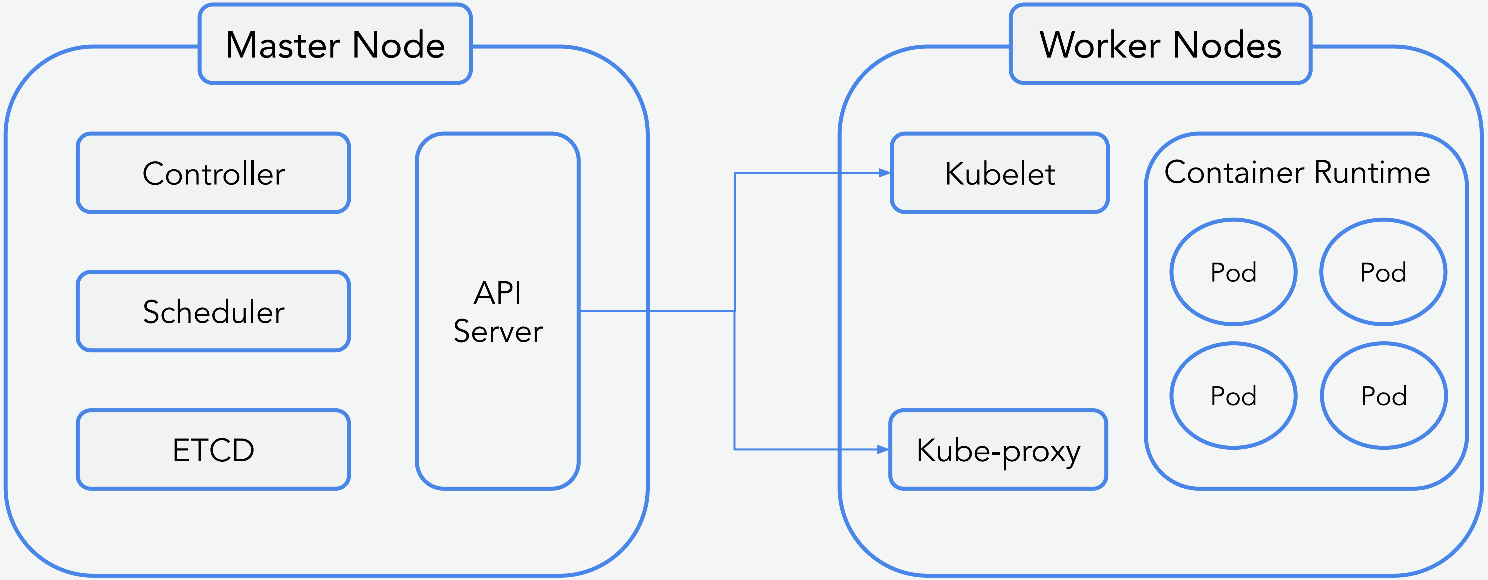
- Understanding the effectiveness of fault-tolerance mechanisms in handling hardware/software failures
- Analysing the logs to create error profiles
- Using the results in EU DECICE Project
    - Enhancing the anomaly detection capabilities of AI models
    - Feeding cluster data into the Digital Twin for real-time monitoring and diagnostics

Background on Cluster Faults

Fault-tolerance: The ability of a system to continue operating in the presence of failures and to automatically heal itself

- Common faults in clusters:
  - **Hardware Failures:** Node, network, and storage failures
  - **Software Failures:** Application crashes, operating system failures, middleware issues
  - **Human Errors:** Configuration errors, operational mistakes
  - **Environmental Factors:** Power outages, cooling failures

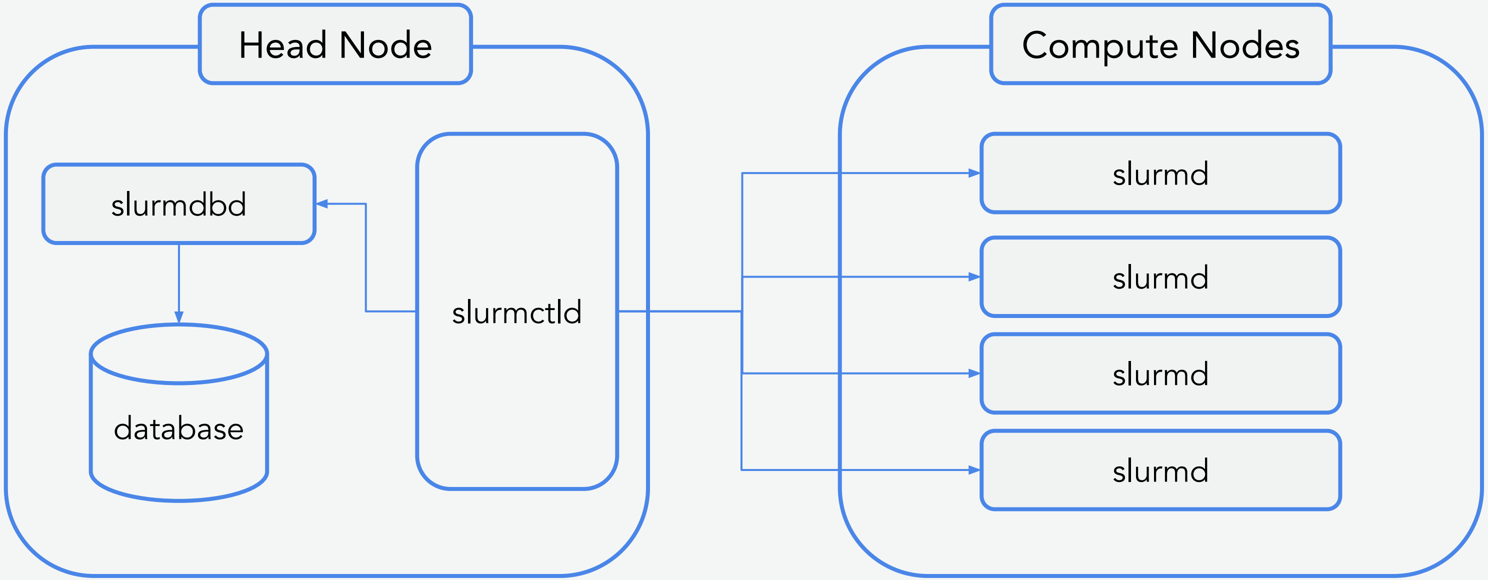
Kubernetes Architecture



Fault-Tolerance in Kubernetes

- Self-Healing: Automatic container restarts
- Replication: Ensures redundancy of pods
- Horizontal Pod Autoscaler (HPA): Adjusts the number of pods based on usage
- CRIU (Checkpoint/Restore): Facilitates live migration and rollbacks
- RAFT Protocol: Maintains state consistency with leader elections (ETCD)

Slurm Architecture



Fault-Tolerance in Slurm

- Node Failover: Reassigns jobs from failed nodes
- Job Checkpointing: Saves state for restart
- Health Checks: Monitors node health
- Job Requeueing: Failed jobs are requeued on healthy nodes

Methodology: Comparative Analysis

Kubernetes Cluster

Resources:
- 3 Master / 6 Worker nodes
- 52 Cores / 203 GB Memory

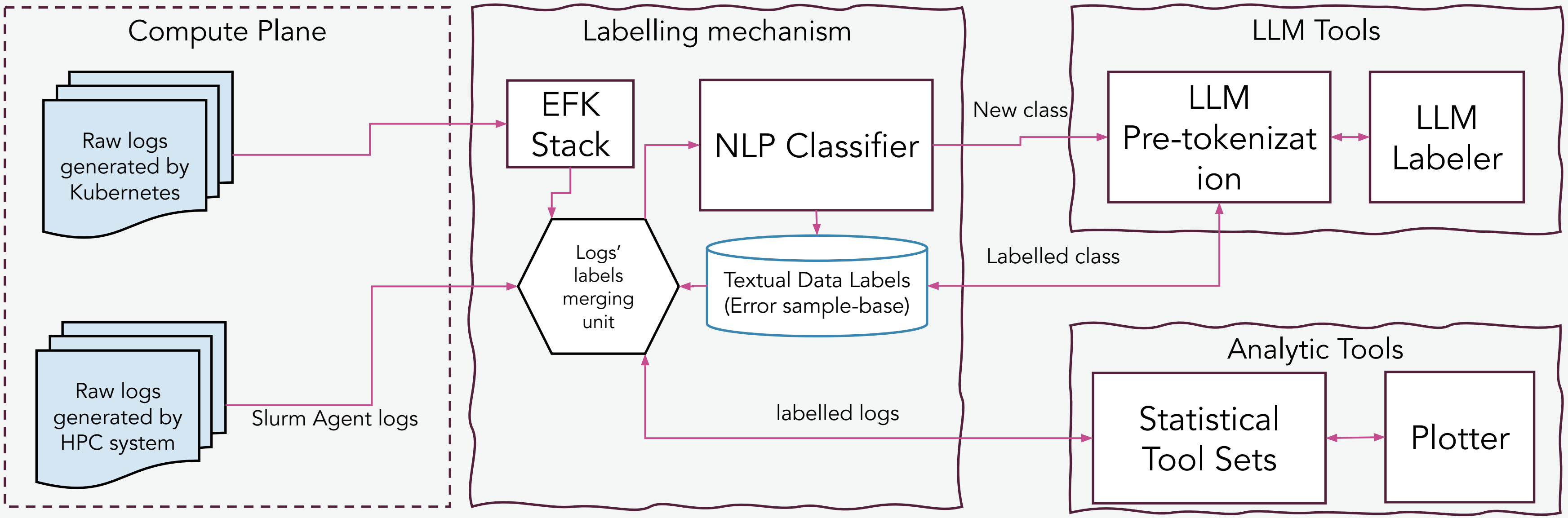
Data Collection:
- EFK Stack (Elasticsearch, Fluentd, Kibana)
- 1.8M log messages

HPC Cluster (SCC)

Resources:
- 410 Compute nodes
- 18.376 Cores / 99 TB Memory

Data Collection:
- Slurm agent logs
- 1.2M log messages

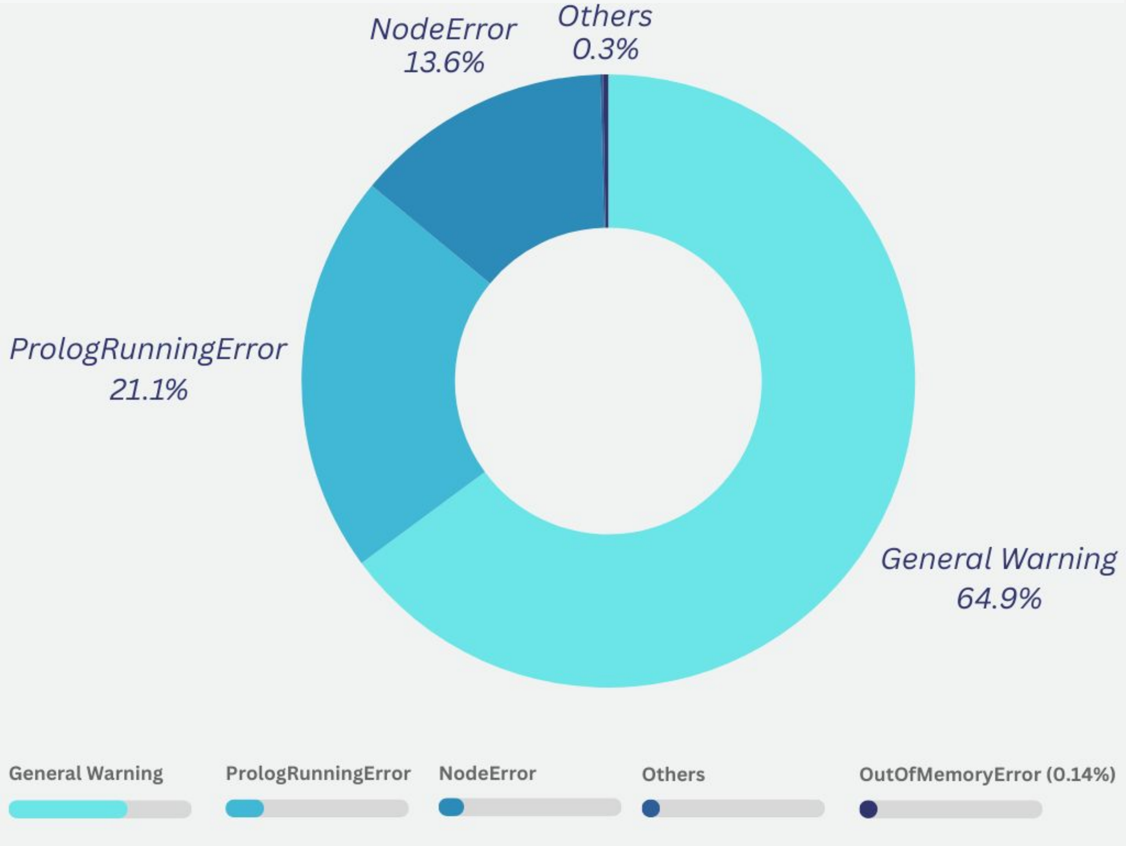
Methodology: Comparative Analysis



Error Distribution in Slurm

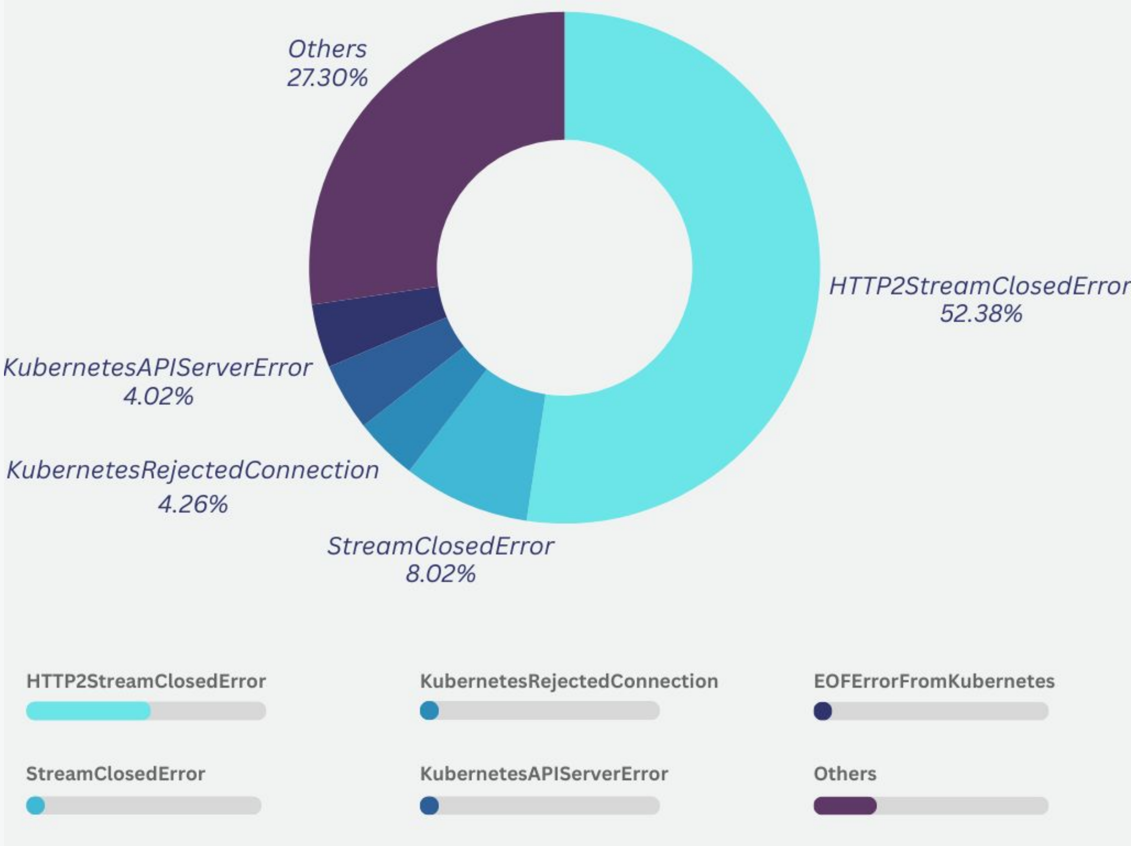
Impact: Highlights node initialization and resource management issues

Error messages that are less than 0.10% are neglected for presentation purposes.



Error Distribution in Kubernetes

Impact: Emphasizes network and API communication issues



Error messages that are less than 4% are neglected for presentation purposes.

Kubernetes vs Slurm

Recovery Time

- Kubernetes is faster due to self-healing and replication
- Slurm depends on node failover

Fault Detection

- Kubernetes robust with software health checks
- Slurm relies on node health checks

Overhead

- Kubernetes higher due to abstraction layers
- Slurm lower due to simpler fault detection mechanisms

Key Insights

Kubernetes excels in scalability, dynamic environments, quick recovery

Slurm is optimal for traditional HPC with efficient scheduling and resource management

Recommendation: Hybrid models leveraging both Kubernetes and Slurm strengths could enhance HPC resilience

Conclusion

- Fault-tolerance mechanisms of Slurm and Kubernetes were investigated
- Error distribution profiles were created for both platforms
  - API communication issues on Kubernetes
  - Job initialization issues on Slurm

Future Work

- Collecting more data to gain better insights
- Exploring AI-based general purpose predictive fault management, hybrid models for fault-tolerance in DECICE

Acknowledgments

Acknowledgment to the DECICE Project, GWDG resources, and supporting team members: Felix Stein, Mojtaba Akbari, Jonathan Decker

Project website: www.decice.eu

This project has received funding from the European Union's Horizon Europe Research and Innovation Programme under Grant Agreement No 101092582.

Quick Review

- Fault-Tolerance Importance: Ensures resilience and continuous operation in HPC systems
- Kubernetes:
  - Container orchestration platform
  - Fault-Tolerance Mechanisms: Self-healing (automatic pod restarts), replication, Horizontal Pod Autoscaler (HPA), RAFT protocol for state consistency
  - Best suited for dynamic, cloud-native environments with scalable workloads
- Kubernetes Strengths: Fast recovery, robust detection, self-healing mechanisms
- Slurm:
  - HPC workload manager designed for large-scale computational jobs
  - Fault-Tolerance Mechanisms: Node failover, job checkpointing, health checks, job requeuing
  - Optimized for traditional HPC systems focusing on resource scheduling and minimal overhead
- Slurm Strengths: Efficient resource use, tailored for traditional HPC, node and job management