Data Science Capstone Project

Internal Final Report

Supercomputer Job Statistics Analytics Project

Sponsored by 

Supervised by **Dr. Nathan**, *Scientist* at *Los Alamos National Laboratory*, New Mexico

Presented by:
Garima Badhan (634005973)
Jnana Preeti Parlapalli (234009464)
Ruthvik Kanumuri (734004406)

1. Introduction

Context and Background:
The Grizzly cluster at Los Alamos National Laboratory (LANL) is a high-performance computing (HPC) system designed to support computational tasks across various scientific domains. Efficient utilization of resources in such systems is critical for performance and cost management. This project focuses on analyzing memory usage patterns from Grizzly's logs to optimize resource allocation and identify inefficiencies.

Problem Statement:
Memory-intensive jobs often exhibit non-linear and unpredictable behavior, leading to resource contention, inefficiencies, and potential system failures. Identifying patterns and anomalies in such high-dimensional data presents significant challenges.

Objective:
To uncover insights from Grizzly cluster logs by analyzing memory usage patterns, and proposing strategies for optimized resource utilization.

Approach and Methodology:
This project employs data preprocessing, exploratory data analysis (EDA), clustering, and predictive modeling to analyze the dataset. By leveraging machine learning and visualization techniques, we aim to identify trends, inefficiencies, and optimization opportunities.

Significance and Applications:
Improved resource management will reduce costs, enhance job scheduling efficiency, and prevent bottlenecks, contributing to smoother HPC operations.

2. Data Summary

Data Source:
The dataset originates from Grizzly cluster logs captured during November 2018. The logs were converted from loosely JSON-formatted files into structured CSVs.

Dataset Description:
- **Size**: Originally 51GB, reduced to 5.75GB after preprocessing.
- **Features**: Memory usage, timestamps, job IDs, host information.
- **Statistics**:
    - Average memory usage: 2.5GB.
    - Range: 0.5GB–10.2GB.
    - Missing values (~5%) were imputed using median values.
- **Key Insights**:
    - Peaks in memory usage during specific hours.
    - Outliers exceeding 20GB flagged as anomalies.

3. Methods

3.1 Data Preprocessing

- **Loading and Cleaning**:

The preprocessing phase involved transforming the raw dataset into a structured format suitable for efficient analysis. The large JSON files, representing loosely formatted system logs, were divided into smaller, manageable chunks to streamline processing. A custom parser was designed to extract relevant fields from these files, converting the unstructured data into a well-organized CSV format, making it easier to manipulate and analyze.
Once the data was converted, individual CSV files were consolidated into a master dataset. This consolidated dataset was indexed and stored in a database to enable efficient querying and streamlined access for subsequent analysis. Python libraries such as Pandas, NumPy, and Dask were utilized to handle the large dataset, ensuring scalability and performance during the preprocessing phase.
The dataset, now in its CSV format, was loaded into a pandas DataFrame for further processing. An output directory was created to store visualizations and plots generated during the analysis. To enhance data quality, invalid rows, such as those with `jobid` values of `0`, were removed. The `timestamp` column, originally in Unix format, was transformed into a human-readable datetime format, facilitating temporal analysis and enabling exploration of time-based trends. These preprocessing steps ensured that the dataset was clean, structured, and ready for advanced analysis, including clustering and time-series modeling.

3.2 Exploratory Data Analysis (EDA)

- **Statistical Summaries**: Calculated the **mean, median, variance, and standard deviation** for memory usage across different hosts and jobs.

Statistic	Value
Count	1.302710e+08
Mean	1.595051e+07
Standard Deviation (std)	1.959913e+07
Minimum (min)	6.690320e+05
25th Percentile (25%)	3.799260e+06
Median (50%)	9.200812e+06
75th Percentile (75%)	1.677782e+07
Maximum (max)	1.271561e+08

- **Time Series Analysis**:
    - Line charts were used to visualize **memory usage trends over time** for specific jobs and hosts.
    - Peak usage periods were identified, showing busy hours between **2 PM and 5 PM**, which likely correspond to high workloads.

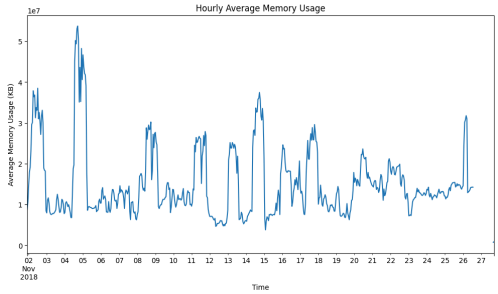

Fig 1: **Time Series Analysis**

- **Distribution Analysis**:
    - **Histograms** provided insights into the distribution of memory usage.
    - Identified nodes with consistently high memory usage, suggesting potential imbalances in workload distribution.

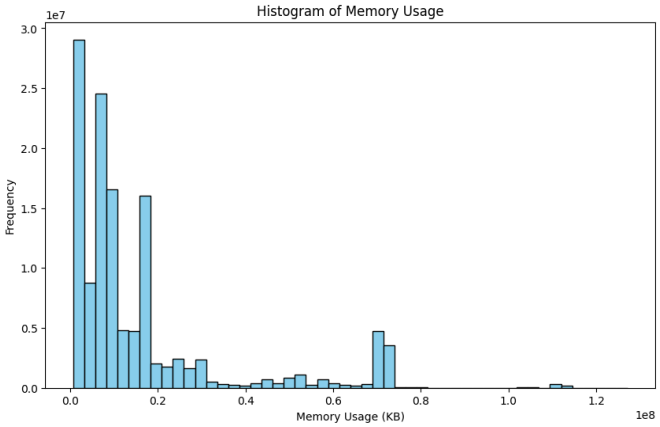

Fig 2: **Distribution Analysis**

- **Correlation and Temporal Analysis Summary**

    - **<u>Memory Usage Insights:</u>**
        - Time of Day: Peak memory usage occurs between 2 PM and 5 PM, suggesting the need for workload balancing during these busy hours.
        - Job IDs: Specific jobs (e.g., 15025738, 15026331) consume disproportionately high memory, exceeding 100 million KB on average, making them targets for optimization.
        - Hosts: Nodes like gr1264 and gr1261 consistently exhibit high memory usage, which could degrade performance. Load balancing and investigation into these nodes are recommended.
        - Job Duration: While there is no strong correlation (correlation coefficient: 0.0433), shorter jobs tend to consume higher memory, necessitating optimization of these resource-intensive tasks.

    - **<u>Temporal Patterns:</u>**
        - Daily Trends: Fridays show the highest memory usage, likely due to batch processing, while Saturdays have the lowest usage, presenting off-peak scheduling opportunities.
        - Weekly Trends: Memory usage fluctuates weekly, with Week 44 having the highest usage and Week 47 the lowest, providing insights for workload planning and maintenance scheduling.

3.3 Data Visualization

- **Line Charts:**
  Trends in overall memory usage over time reveal fluctuating patterns with distinct peaks, such as on November 5th and 26th, indicating high activity or resource-intensive workloads on those days. The memory usage curve highlights periods of increased activity, guiding resource planning and scheduling optimizations.

- **Heatmaps:**
  The heatmap visualization shows memory usage distribution across nodes (hosts) and dates. Specific hosts exhibit consistent high memory usage, indicating potential bottlenecks or uneven workload distribution. Bright spots on certain days suggest intensive cluster activity on those hosts, pointing to potential scheduling inefficiencies.

- **Interactive Dashboards:**
  Dashboards built using interactive tools like Plotly enable dynamic exploration of memory usage trends, allowing stakeholders to zoom in on specific dates, nodes, or patterns for deeper insights.

3.4 Advanced Analytics

Clustering Techniques

1. **K-Means Clustering**:
   - Raw data was grouped by `jobid` to calculate statistical features (`*mean*`, `*max*`, `*min*`, `*std*`) and job duration (difference between `*timestamp_min*` and `*timestamp_max*`), with infinite and missing values handled. Five key features—`*value_mean*`, `*value_max*`, `*value_min*`, `*value_std*`, and `*duration*`, standardized using **StandardScaler** to ensure equal contribution to clustering.
   - The *Elbow Method* was applied, where inertia values were plotted for cluster counts : k=1 to k=10. A *K-Means* model with k=3 (elbow point) was fitted to the standardized features, assigning each job to one of three clusters based on their similarities.
   - The clustering results were visualized using a **pair plot** to show relationships between features and cluster distributions, and an **elbow curve** was generated to validate the choice of k=3.

2. **Hierarchical Clustering:**
   - Hierarchical clustering (Bottom-Up approach) was employed to group jobs based on feature similarity. The ward linkage method was used, which iteratively merges clusters to minimize the increase in within-cluster variance. This ensures that the resulting clusters are compact.
   - Dendrograms visualize the hierarchy of clusters. The x-axis represents the job IDs or indices, and the y-axis shows the distance at which two clusters are merged.
   - The *truncate_mode='level'* parameter was used to display only the top 5 levels of the clustering process, showing the last few merged clusters for clarity.
   - Clusters were defined by "cutting" the dendrogram at a specific distance threshold (*max_d = 7*). This distance was chosen to identify distinct, meaningful groupings based on the structure of the dendrogram.

3. **Gaussian Mixture Models (GMM)**:
   - The dataset was filtered for valid *jobid*, *timestamps* were converted to datetime format, and job-level statistics (*job_duration*, *avg_memory_usage*, *std_memory_usage*, *max_memory_usage*, *num_records*) were calculated.
   - Standardized features using **StandardScaler** and applied a 3-component **Gaussian Mixture Model (GMM)** for probabilistic cluster assignment, allowing flexibility for overlapping and complex distributions.
   - GMM helped identify overlapping clusters, which are challenging to detect with traditional methods.

4. **HDBSCAN (Hierarchical Density-Based Spatial Clustering)**:
    - The dataset is sampled to 80,000 rows using *sample()* to reduce memory usage during processing.
    - **HDBSCAN** identifies clusters in data based on density, and parameters such as *min_cluster_size=100* (minimum points for a cluster) and *min_samples=50* (minimum neighbors for a core point) control how clusters are formed.
    - The algorithm assigns cluster labels to points, and *outliers* are labeled as *-1*. This is particularly useful for handling noisy or irregular data.

5. **Clustering into 9 buckets:**
    - This method processes job data by grouping it based on `*jobid*` to calculate each job's duration (in hours) and average memory usage.
    - Jobs are categorized into combinations of memory utilization (high, med, low) and duration (long, med, short) using quantile thresholds.

Predictive Modeling

1. **ARIMA and Prophet**:
    - The Prophet model was used for time series forecasting, resampling the data to hourly frequency and generating predictions for the next 24 hours.
    - ARIMA model was employed to forecast memory usage by using past values to predict future memory usage trends and was visualized by comparing actual vs. forecasted values.

2. **Long Short-Term Memory (LSTM)**:
    - Compared to traditional models, the LSTM demonstrated superior performance in predicting sudden spikes or declines in memory demand. This capability is particularly valuable for dynamic workloads, where abrupt changes in memory usage can significantly impact resource planning and allocation.

4. Results and Recommendations

4.1 Resource Bottleneck Identification

- **High-Memory Jobs**: Jobs such as 15025738 and 15026331 were identified as resource-intensive, with average memory usage exceeding **100 million KB**. These jobs may represent potential bottlenecks that require optimization.
- **Heavy Load Hosts**: Nodes like gr1264 and gr1261 exhibited consistently high memory usage, suggesting uneven workload distribution.

4.2 Temporal Patterns

- **Daily Trends**: Memory usage peaks on **Fridays**, indicating that batch jobs or maintenance tasks might be scheduled towards the end of the week.
- **Weekly Trends**: Usage fluctuations were observed, with **Week 44** showing the highest average memory usage and **Week 47** showing the lowest. Such insights can help in planning cluster maintenance.

4.3 Job Efficiency

- **Efficient Jobs**: Jobs like 15020060 were identified as efficient, consuming **2743.64 KB/s** on average, while **least efficient jobs** (e.g., 15008500) consumed over **1.5 billion KB/s**. This indicates potential inefficiencies that require further optimization.

4.4 k-Means Clustering

Plot explanation:

Cluster 0 (orange): Lower values across all variables. Cluster 1 (green): Higher values across all variables. Cluster 2 (blue): Intermediate values with moderate separation. Overall:Clusters are well-separated in most pairwise relationships, with overlap in some (e.g., `*value_std*` vs. `*duration*`).

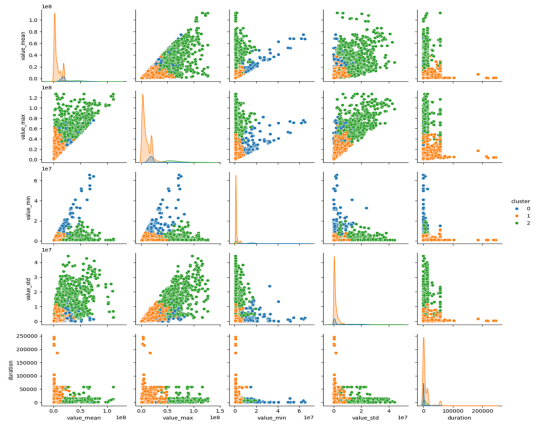

Fig 3: **k-Means clustering pairplot**

4.5 Hierarchical Clustering

Plot Explanation: **Clusters Formation**: The orange and green branches represent two main clusters. Sub-clusters within these branches merge at smaller distances, indicating higher similarity. **Threshold for Clusters**: By cutting the dendrogram at a certain distance (=7), distinct clusters can be identified. It provides a dendrogram, offering clear visualization of nested relationships and flexibility in determining cluster count, unlike K-Means which requires a predefined number of clusters.

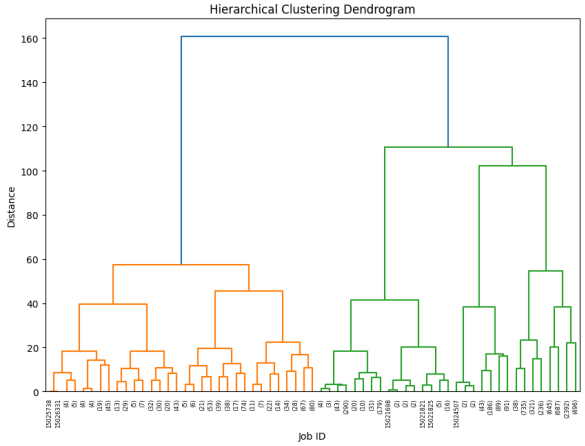

Fig 4: **Hierarchical clustering Dendrogram**

4.6 GMM Clustering

Plot Explanation

Bar and count plots show average memory usage and job count per cluster, showing differences in memory consumption and cluster sizes.

Clusters are well-separated in the t-SNE space, indicating that GMM effectively identifies meaningful groups. Cluster 0 (purple) appears more widespread, while Clusters 1 (yellow) and 2 (teal) are more compact and localized. GMM better captures varying cluster shapes and overlaps, as seen in the t-SNE plot, offering flexibility over hierarchical clustering's rigid structure.

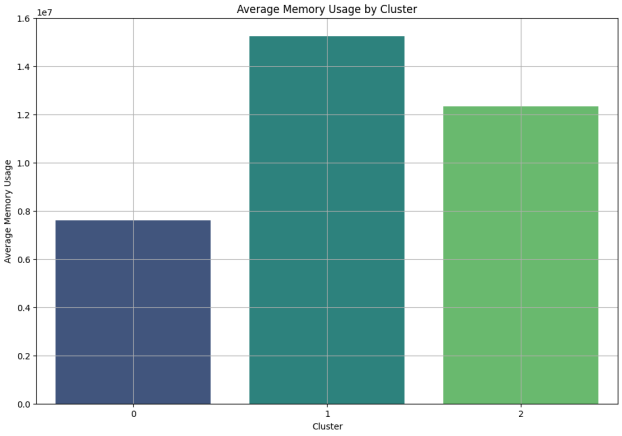

Fig 5: **GMM - Average memory usage cluster-wise**

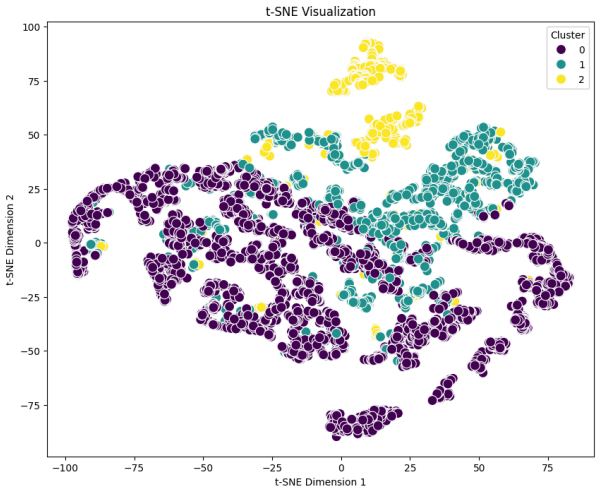

Fig 6: ***t-sne* visualization - GMM Clustering**

4.7 HDBSCAN Clustering

Plot Explanation

Points labeled *-1* are scattered throughout the plot, indicating unusual memory usage that deviates from the identified clusters. Clusters form horizontal bands, indicating that memory usage patterns are relatively stable over specific periods but vary between clusters. Also, the outlier cluster (= -1) has spikes in its behaviour in *hourly temporal patterns* plot depicting that it is an anomaly. It identifies outliers and handles clusters of varying shapes and densities without assuming Gaussian distributions, unlike GMM.

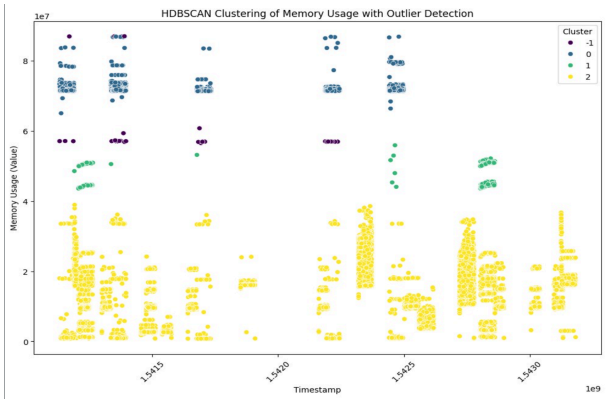

Fig 7: HDBSCAN Clustering scatterplot

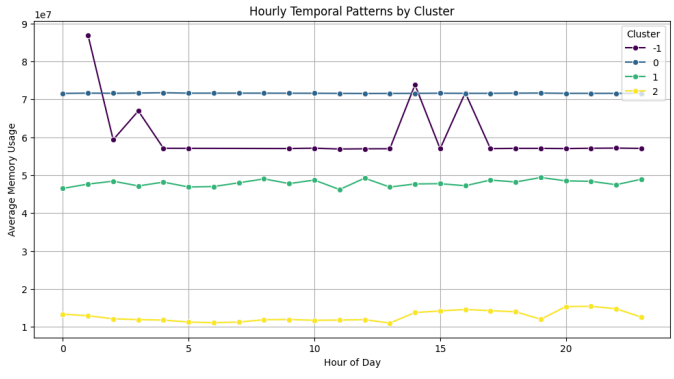

Fig 8: HDBSCAN- Hourly Temporal Trends

4.8 Clustering into 9 buckets:

Plot Explanation: The plot effectively groups jobs into specific categories based on memory utilization and duration, with clear color-coded clusters representing different utilization-duration combinations. Jobs with **high utilization** are concentrated in longer durations, suggesting intensive resource usage for extended periods, while **low utilization** jobs are spread across shorter durations.The clustering highlights potential areas to optimize, such as reallocating low-utilization short-duration jobs for better resource efficiency.

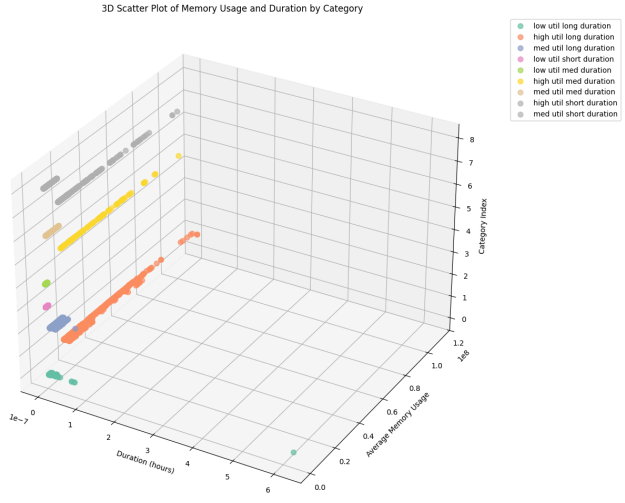

Fig 9: Clustering into 9 buckets- 3D scatterplot

4.9 ARIMA and Prophet forecasting:

Plots Explanation: Both the Prophet and ARIMA models capture general trends in memory usage but fail to predict sharp spikes accurately, indicating that their forecasting performance is insufficient for handling large fluctuations in the data.

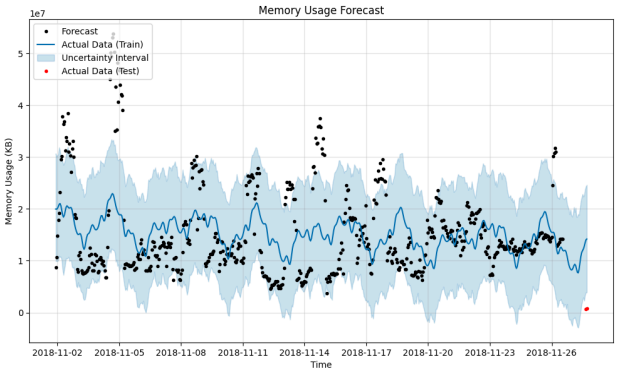

Fig 9: Prophet forecast visualization

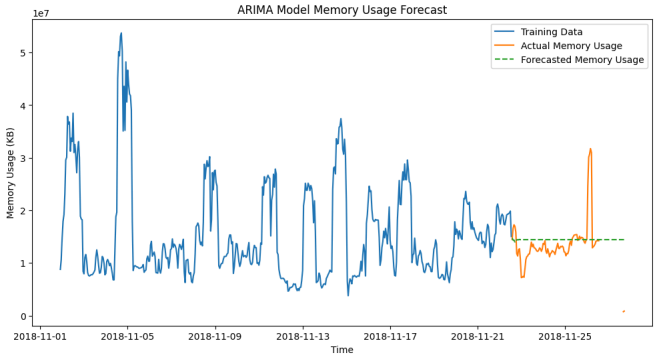

Fig 10: ARIMA forecast method visualization

4.10 LSTM forecasting:

Plot Explanation: The LSTM graph highlights forecasted memory usage alongside actual data, showcasing the model's predictive accuracy. Unlike ARIMA, which assumes stationarity and linear relationships, or PROPHET, which may underperform with irregular seasonality, LSTM leverages its sequential learning capabilities to handle intricate dependencies and long-term trends, resulting in more reliable and precise predictions.

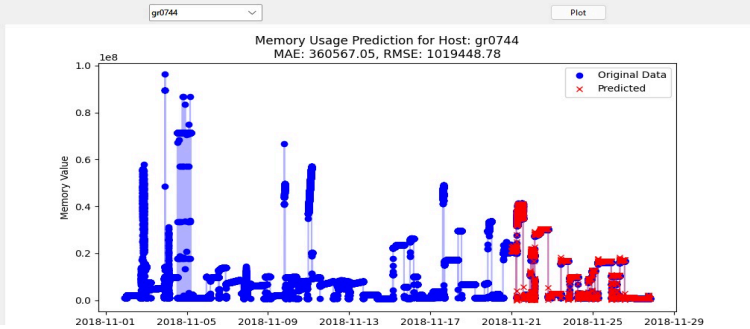

Fig 10: LSTM prediction method

4.11 Enhance Monitoring with Predictive Models

- Leverage **Temporal Time Series Transformer** models to predict memory usage for the next timestamps based on historical trends. This enables proactive management of potential bottlenecks.
- Use **Chronos Prediction Model** to establish a baseline for normal behavior. Deviations from the predicted behavior can help detect anomalies earlier, improving efficiency.

5. References

1. **Procstat Overview -** LinuxHowtos.org. (n.d.). *procstat*. - https://www.linuxhowtos.org/System/procstat.htm
2. **Memory Usage Overview -** Baeldung on Linux. (n.d.). *Understanding /proc/meminfo*. Retrieved from https://www.baeldung.com/linux/proc-meminfo
3. **Related Work -** Los Alamos National Laboratory. (2019). *System Log Files from Grizzly*. Retrieved from https://usrc.lanl.gov/data/LA-UR-19-28211.php
4. Halawa, M. S., Mohamed, S., & Goudarzi, M. (Year). *Unsupervised KPIs-Based Clustering of Jobs in HPC Data Centers*. Sensors Journal
5. Park, J.-W., Lee, S.-H., & Kim, H.-J. (Year). *Analyzing and Predicting Job Failures from HPC System Logs*. The Journal of Supercomputing
6. Mohammed, B., Zhao, X., & Barker, K. (Year). *Failure Prediction Using Machine Learning in a Virtualized HPC System*. Cluster Computing.