**CDAC Project**

**Title**

**AI-Driven Resource Management and Scheduling Optimization in High-Performance Computing (HPC) Systems**

**Abstract**

This project explores the development of an AI-driven resource management system tailored to optimize job scheduling and resource utilization within high-performance computing (HPC) environments. By leveraging predictive modelling and adaptive scheduling algorithms, the project aims to address common inefficiencies in traditional HPC resource management, such as high job wait times, suboptimal resource allocation, and increased energy consumption. Through simulation and testing, the proposed system demonstrates improvements over traditional scheduling techniques, showcasing potential for scalable, efficient, and adaptable resource management in HPC clusters.

**Introduction**

High-performance computing systems are essential in research and industry for processing large-scale computations. Efficient resource management is critical to maximizing HPC cluster performance, reducing operational costs, and maintaining service quality. Traditional scheduling methods, such as FIFO or static round-robin, often fail to account for the varying resource demands of individual jobs, leading to inefficiencies. This project introduces an AI-driven approach to predict job requirements and dynamically allocate resources, aiming to improve overall HPC performance and reduce energy consumption.

**Objectives and Scope**

The primary objectives of this project are to:

1. Develop a predictive AI model to forecast resource needs for incoming jobs based on historical data.
2. Design an adaptive scheduling system that leverages these predictions to optimize job placement and resource allocation.
3. Assess the proposed system’s impact on key performance indicators (KPIs) such as job wait time, resource utilization, and energy efficiency.
4. Compare the effectiveness of this AI-driven scheduler with traditional HPC scheduling techniques.

The scope of this project is focused on simulating an HPC environment, utilizing open-source scheduling tools (e.g., SLURM or Torque), and applying AI models that are computationally feasible within a student budget.

**Literature Review and Theoretical Background**

Resource management and scheduling in HPC have been extensively researched, with classical algorithms often relying on heuristic or static methods. However, these methods are limited in handling dynamic and unpredictable workload patterns. Recent advancements in machine learning and AI have paved the way for more sophisticated, data-driven approaches that can predict resource needs and adaptively adjust resource allocation. Reinforcement learning, decision trees, and regression models have shown promise in handling job scheduling challenges by forecasting resource utilization and adapting to real-time cluster demands.

This project leverages predictive modelling (using methods such as Linear Regression and Random Forest) to estimate resource requirements and applies these estimates in a custom scheduling algorithm. By incorporating machine learning into HPC resource management, this approach aims to address the limitations of traditional methods and provide a more flexible and efficient scheduling solution.

**Methodology**

1. **Data Collection and Preprocessing**: Historical job logs from public HPC workload repositories, such as the Parallel Workloads Archive, were used to train the predictive model. These logs were cleaned, normalized, and prepared for model training.
2. **Predictive Model Development**: A predictive model was developed using machine learning algorithms to forecast CPU, memory, and job duration requirements based on job type, submission time, and other historical data points. Initial model iterations used Linear Regression and Random Forest, followed by hyperparameter tuning for performance improvements.
3. **Adaptive Scheduler Design**: A custom scheduling script was created to integrate with the predictive model. This scheduler uses the model’s forecasts to match jobs with available resources, dynamically adjusting resource assignments based on real-time demand and job priority levels.
4. **Simulation and Testing**: A simulated HPC environment was set up using SLURM on virtual machines. Various job scenarios were tested, including CPU-intensive and memory-heavy tasks, to evaluate the system’s adaptability and efficiency.
5. **Performance Benchmarking**: The AI-driven scheduler was benchmarked against traditional FIFO and round-robin schedulers using metrics such as job wait time, resource utilization, and energy consumption. Data from these tests were visualized and analyzed.
6. **Monitoring Dashboard**: A real-time dashboard was developed using Grafana to display job statuses, utilization rates, and performance metrics. This provided a user-friendly view of the scheduler’s effectiveness.

**Results and Analysis**

The AI-driven scheduler outperformed traditional methods in several key areas:

* **Reduced Job Wait Time**: The system reduced average queue wait times by efficiently allocating resources based on predicted job requirements.
* **Enhanced Resource Utilization**: The adaptive scheduling algorithm achieved higher resource utilization rates by matching jobs to resources based on need, reducing idle time.
* **Energy Efficiency**: Simulated energy savings were achieved by placing resources in low-power states during periods of low demand, guided by AI-based workload forecasts.

The results indicate that an AI-driven approach can significantly improve HPC scheduling performance, particularly in heterogeneous and dynamic workload environments.

**Conclusion**

The AI-driven resource management system developed in this project represents a novel approach to HPC scheduling, combining predictive modeling with adaptive scheduling algorithms. The system effectively improves upon traditional methods by reducing wait times, increasing resource utilization, and offering potential energy savings. These findings suggest that AI-based scheduling has the potential to significantly enhance HPC performance, making it a viable solution for large-scale computing environments with diverse workloads. Further research could explore integrating reinforcement learning techniques and expanding the system to accommodate a wider range of HPC workloads.

**Future Work**

1. **Reinforcement Learning**: Implementing reinforcement learning could enable the scheduler to adapt even more dynamically, learning from real-time interactions with the HPC environment.
2. **Scalability Testing**: Conduct larger-scale tests on actual HPC clusters to validate the model's scalability and generalizability.
3. **Energy Optimization**: Integrate more comprehensive energy-saving techniques, particularly for large, multi-node clusters, to improve energy efficiency further.