UDC 004.8:004.75 DOI: 10.31673/2412-4338.2025.013820

**Mukhin Vadym**
*National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv*
ORCID: 0000-0003-3833-1529

**Kulyk Viacheslav**
*National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv*
ORCID: 0000-0003-3833-1529

# HYBRID AI ARCHITECTURE FOR DYNAMIC WORKLOAD SCHEDULING IN LARGE-SCALE DISTRIBUTED COMPUTING SYSTEMS

***Abstract:*** *This research presents a novel hybrid artificial intelligence architecture for optimizing task scheduling in large-scale distributed computing environments, combining distributed AI Schedulers (AIS) with a centralized Decision Tree (DT) layer to achieve superior scheduling accuracy and adaptability. Traditional scheduling approaches struggle with heterogeneous environments, leading to suboptimal resource utilization, which our two-tiered architecture addresses through cluster-level AI Schedulers that pre-select suitable nodes based on four key metrics: performance, data transfer rate, operational time, and security level. The neural network architecture employs two hidden layers with ReLU activation functions, while the Decision Tree layer uses an enhanced CART algorithm for final node selection, incorporating both primary characteristics and historical performance data. Experiments conducted across deployment scales from 5 clusters (500 nodes) to 30 clusters (15000 nodes) demonstrate the hybrid approach achieves 99-100% accuracy in node selection, significantly outperforming the standalone AI Scheduler's 94-96% accuracy, while maintaining consistent performance and transparent decision-making processes across all scales. This architecture proves particularly effective for cloud computing environments, IoT deployments, and distributed systems requiring sophisticated resource allocation, contributing a scalable, accurate, and interpretable scheduling solution that effectively combines local intelligence with centralized decision-making for broad applicability in domains requiring dynamic resource allocation in distributed environments.*

***Keywords:*** *Distributed task scheduling, artificial intelligence, hybrid architecture, decision trees, neural networks, heterogeneous systems, resource optimization, cloud computing.*

**Мухін Вадим Євгенович**
*НТУУ «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ*
ORCID: 0000-0003-3833-1529

**Кулик Вячеслав Олександрович**
*НТУУ «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ*
ORCID: 0000-0003-3833-1529

# ГІБРИДНА АРХІТЕКТУРА ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ДИНАМІЧНОГО ПЛАНУВАННЯ РОБОЧОГО НАВАНТАЖЕННЯ У ВЕЛИКОМАСШТАБНИХ РОЗПОДІЛЕНИХ ОБЧИСЛЮВАЛЬНИХ СИСТЕМАХ

***Анотація:*** *це дослідження представляє нову гібридну архітектуру штучного інтелекту для оптимізації планування завдань у великомасштабних розподілених обчислювальних середовищах, поєднуючи розподілені планувальники штучного інтелекту (AIS) із централізованим рівнем дерева рішень (DT) для досягнення надзвичайної точності та адаптивності планування. Традиційні підходи до планування борються з неоднорідними середовищами, що призводить до неоптимального використання ресурсів, які наша дворівнева архітектура вирішує за допомогою планувальників штучного інтелекту на рівні кластера, які попередньо вибирають відповідні вузли на основі чотирьох ключових показників: продуктивності, швидкості передачі даних, оперативного часу та безпеки. рівень. Архітектура нейронної мережі використовує два приховані рівні з функціями активації ReLU, тоді як рівень дерева рішень використовує розширений алгоритм CART для остаточного вибору вузла,* *що включає як основні характеристики, так і історичні дані про продуктивність. Експерименти, проведені на масштабах розгортання від 5 кластерів (500 вузлів) до 30 кластерів (15000 вузлів), демонструють, що гібридний підхід досягає 99-100% точності вибору вузлів, значно перевершуючи точність автономного AI Scheduler (94-96%), зберігаючи незмінну ефективність і прозорі процеси прийняття рішень у всіх масштабах. Ця архітектура є особливо ефективною для хмарних обчислювальних середовищ, розгортань IoT і розподілених систем, що вимагають складного розподілу ресурсів, створюючи масштабоване, точне та інтерпретоване рішення для планування, яке ефективно поєднує локальний штучний інтелект із централізованим прийняттям рішень для широкого застосування в областях, що потребують динамічного розподілу ресурсів. у розподілених середовищах.*

***Ключові слова:*** *розподілене планування завдань, штучний інтелект, гібридна архітектура, дерева рішень, нейронні мережі, гетерогенні системи, оптимізація ресурсів, хмарні обчислення.*

## Introduction

In today's world, distributed computing systems are becoming increasingly prevalent, with applications ranging from cloud computing to edge devices. Efficiently scheduling tasks across these distributed resources is a critical challenge[1]. Traditional scheduling methods often struggle to adapt to the dynamic nature of these environments[1], leading to suboptimal resource utilization and performance bottlenecks. This article presents an innovative approach to distributed task scheduling that leverages the power of artificial intelligence to achieve superior performance and adaptability.

In practice, this multi-tier approach is particularly valuable in scenarios such as cloud computing, where resource availability can shift rapidly, IoT data processing pipelines that handle vast and fluctuating data streams, and automation systems requiring timely allocation of tasks across interconnected machines[2]. By dynamically matching tasks to the most suitable nodes, resource usage is optimized while keeping latency and overhead in check[2].

Our system introduces a hierarchical architecture that combines the strengths of local intelligence and centralized decision-making. At the cluster level, we employ AI Schedulers (AIS) that monitor node status, manage resource allocation, and pre-select suitable nodes based on task requirements. These AIS components operate independently within their respective clusters, providing localized optimization and responsiveness to changing conditions. The pre-selected nodes from each cluster are then passed to a central Decision Tree (DT) layer. The DT evaluates the suitability of each node based on multiple criteria and makes the final node allocation decision. This two-tiered approach ensures robust node selection and efficient task execution.

The system is designed to be scalable and adaptive. The distributed nature of the AIS components allows the system to easily accommodate additional clusters, while the continuous monitoring of task execution enables dynamic adjustments to changing conditions. This approach maximizes resource utilization and ensures optimal performance in a multi-cluster environment. The following sections will delve into the details of the neural network used within the AI Schedulers, the structure of the Decision Tree, and the overall system architecture. We will also discuss the datasets used for training and evaluation and present the results of our experiments.

## General Approach Overview

The proposed system implements a sophisticated distributed task scheduling architecture across multiple heterogeneous clusters through six primary stages.

1. Task Submission and Analysis

The system ingests computational tasks and analyzes their requirements, including computational complexity, resource needs, and execution constraints. This initial assessment forms the basis for subsequent scheduling decisions.

2. Cluster Distribution

Task specifications are broadcast to all cluster-level AI Schedulers, initiating parallel node evaluation within each cluster to maximize efficiency and reduce selection time.

3. AI-Based Node Pre-selection

AIS components filter available nodes using current resource availability, performance history, and load conditions, creating an initial pool of suitable candidates from each cluster.

4. Cross-Cluster Decision Tree Evaluation

The system aggregates pre-selected nodes into a unified pool, applying multi-criteria decision tree analysis to determine optimal node selection based on both individual capabilities and cross-cluster factors.

5. Task Assignment

Selected nodes receive task assignments following validation of resource availability and execution parameters, ensuring reliable task initiation.

6. Continuous Monitoring

Real-time performance tracking and dynamic adaptation mechanisms maintain optimal execution conditions throughout the task lifecycle.

This hierarchical approach ensures efficient resource utilization while maintaining system adaptability and scalability.

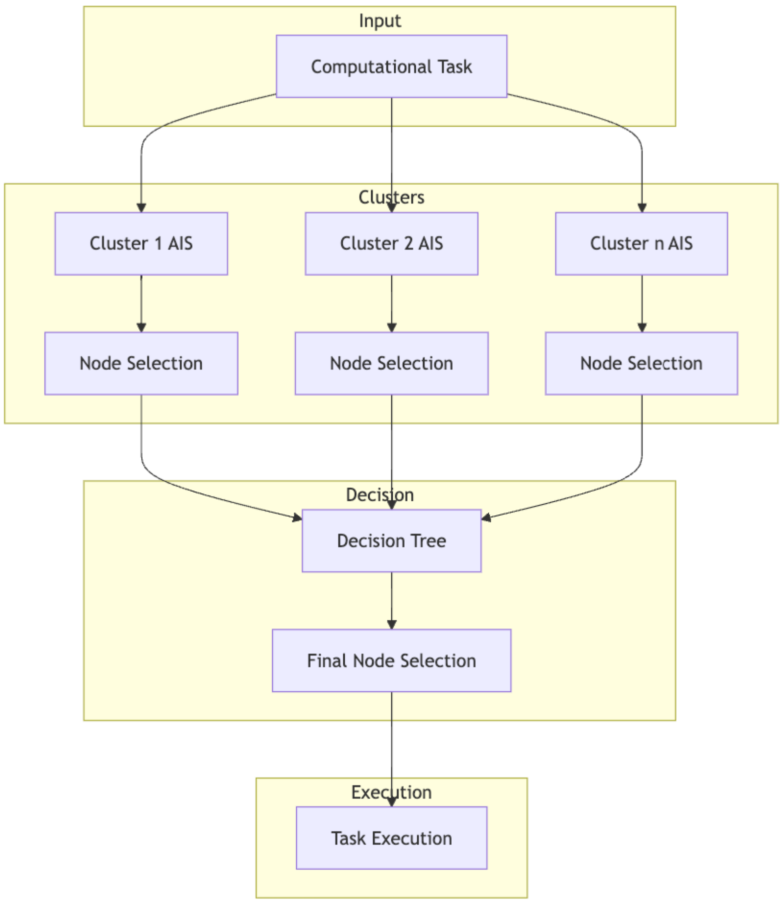


Fig. 1. Six-stage process of task scheduling in our hybrid AI system. The diagram shows the progression from initial task submission through AI-based node pre-selection to final task assignment, highlighting the interaction between cluster-level AI Schedulers (AIS) and the centralized Decision Tree (DT) layer

**Comparative Analysis with Traditional Approaches**

The proposed hybrid AI architecture introduces several theoretical advantages over conventional task distribution algorithms such as Round Robin (RR) and Least Connections (LC) scheduling. While these traditional approaches offer simplicity and predictable behavior, they lack the ability to consider multiple parameters simultaneously or adapt to changing conditions[3].

**Key Architectural Differences**

Decision-Making Complexity

Traditional scheduling approaches exhibit varying levels of computational sophistication. The Round Robin scheduler follows a simple cyclic distribution pattern, while Least Connections bases its decisions solely on current connection counts[3]. In contrast, our hybrid AI architecture implements a comprehensive evaluation framework that simultaneously considers multiple operational parameters, including performance metrics, network capabilities, system reliability, and security requirements.

Adaptability

Environmental changes in distributed systems often necessitate scheduler reconfiguration[3,4]. While traditional approaches require manual intervention to adjust to new conditions, our architecture implements continuous learning mechanisms and real-time parameter evaluation to automatically adapt to evolving system states.

Node Selection Intelligence

Conventional scheduling methods rely on simplified decision metrics, potentially overlooking critical operational parameters. Our hybrid architecture elevates the selection process by combining neural network pre-selection with decision tree refinement[5], achieving consistent node selection accuracy rates of 99-100% across diverse operational scenarios.
Future work could include direct performance comparisons with these traditional approaches under identical workload conditions to measure specific efficiency gains.

**Neural Network Architecture for AI-Scheduler**

The neural network employed in this study serves as the AI scheduler responsible for predicting optimal nodes based on input parameters. The architecture is designed to balance complexity and performance, ensuring accurate predictions while maintaining computational efficiency.

**Layers and Components**

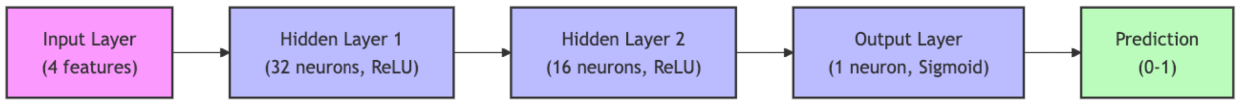


Fig.2. Neural network architecture diagram for the AI Scheduler component. The network consists of an input layer accepting four normalized node characteristics (performance, data transfer rate, operational time, and security level), two hidden layers with 32 and 16 neurons respectively using ReLU activation functions, and an output layer with sigmoid activation producing node suitability scores. Layer dimensions were determined through empirical testing to balance model complexity with prediction accuracy

The neural network architecture comprises three primary components: an input layer, two hidden layers, and an output layer. The input layer accepts four critical parameters that characterize node capabilities and operational characteristics[6]. These parameters include performance metrics that quantify computational efficiency, data transfer rates that measure network communication capabilities, operational time metrics that assess node reliability and availability, and security level indicators that evaluate the robustness of implemented security measures.

The network's processing capabilities are enhanced through two hidden layers. The first hidden layer consists of 32 neurons utilizing Rectified Linear Unit (ReLU) activation functions. This configuration enables the network to capture non-linear relationships within the input data while maintaining computational efficiency[6]. The implementation of ReLU activation functions addresses the vanishing gradient problem commonly encountered in deep neural networks, thereby facilitating more effective training through improved gradient propagation.

The second hidden layer, comprising 16 neurons with ReLU activation, further refines the feature representations learned by the first layer. This dimensional reduction approach allows the network to distill the most relevant features while maintaining the capacity to model complex relationships between input parameters. The progressive reduction in layer dimensions (from 32 to 16 neurons) follows established principles in neural network architecture design, promoting efficient information compression while preserving essential feature relationships.

The output layer employs a single neuron with a sigmoid activation function, producing a probability score within the range from 0 to 1. This probability represents the model's confidence in node suitability for task allocation. The sigmoid activation function is particularly appropriate for this binary classification task, as it provides a clear probabilistic interpretation of the network's decision. The final classification is determined by applying a threshold value (typically 0.5) to this probability score, effectively partitioning nodes into suitable and unsuitable categories.

**Datasets and training**

The training and evaluation datasets were generated through simulation of distributed computing environments using our custom simulation framework. This approach allowed us to model diverse cluster configurations and node behaviors systematically.

The simulation encompassed three distinct deployment scales, beginning with a small-scale deployment of 5 clusters containing between 100 nodes each, yielding a total range of 500 nodes. The medium-scale deployment expanded to 15 clusters, each containing 300 nodes, resulting in 4500 total nodes. Our large-scale deployment further extended to 30 clusters with a ratio of 500 nodes per cluster, achieving a comprehensive range of 15000 nodes in total. For each scale, we simulated node characteristics using our Python-based simulation framework.

The framework generated synthetic performance data for each node, with measurements collected at regular intervals throughout the simulation period. Each record contained normalized values (0-100) for the four primary node characteristics (Performance, Data transfer rate, Operational time, and Security level). The distribution parameters were configured to reflect typical variations observed in distributed computing environments, with values generated using pseudo-random number generators with predefined ranges for each parameter.

While simulated data was used for initial development and validation, future work could incorporate data from real-world cluster deployments to further validate and refine the model's performance under actual operating conditions.

**Parameter Selection Reasoning**

In this study, the selection of four key parameters—performance, data transfer rate, operational time, and security level—was guided by their direct impact on a node's operational efficiency and stability. Performance and data transfer rate collectively capture both computational power and network throughput, which are crucial for time-sensitive distributed workloads[7]. Operational time highlights system reliability, indicating the likelihood of a node remaining available throughout task execution. The security level metric ensures compliance with confidentiality and integrity requirements, especially critical in cloud, IoT, and automation environments where sensitive data may be processed.

While these parameters capture the majority of operational considerations, integrating additional metrics is also possible. For example, memory utilization, latency variations, and energy consumption could further refine the model's predictive accuracy. However, incorporating more parameters would require data collection adjustments and careful retraining of both the neural network and decision tree. By maintaining a focused set of parameters, our approach ensures computational efficiency and acceptable model complexity, without limiting the possibility of extending to broader metrics in future implementations.

**Decision Tree Layer**

The Decision Tree (DT) layer represents a crucial architectural component in our distributed task scheduling system, serving as the final arbitrator in node allocation decisions[8]. This layer implements a sophisticated decision-making mechanism based on the scikit-learn DecisionTreeClassifier, which has been specifically adapted to handle the complexity of distributed computing environments. The DT layer operates as a secondary filtering mechanism, processing and refining the preliminary selections made by the distributed AI Schedulers (AIS).

**Architecture and Implementation**

The decision tree architecture employs a hierarchical structure of binary decisions, where each internal node represents a conditional test on a specific feature, and each leaf node corresponds to a final classification decision[8]. Our implementation utilizes an optimized version of the CART (Classification and Regression Trees) algorithm, which has been enhanced to handle the particular requirements of distributed task scheduling.

The feature space for the decision tree encompasses two distinct categories of inputs:

- Primary Node Characteristics:
  - Computing speed (normalized computational capacity)
  - Data transfer rate (network bandwidth capabilities)
  - Operational uptime (system reliability metric)
  - Security level (binary security compliance indicator)
- Secondary Features:
  - AIS confidence scores (probabilistic outputs from neural networks)
  - Historical performance metrics

**Training and Operational Process**

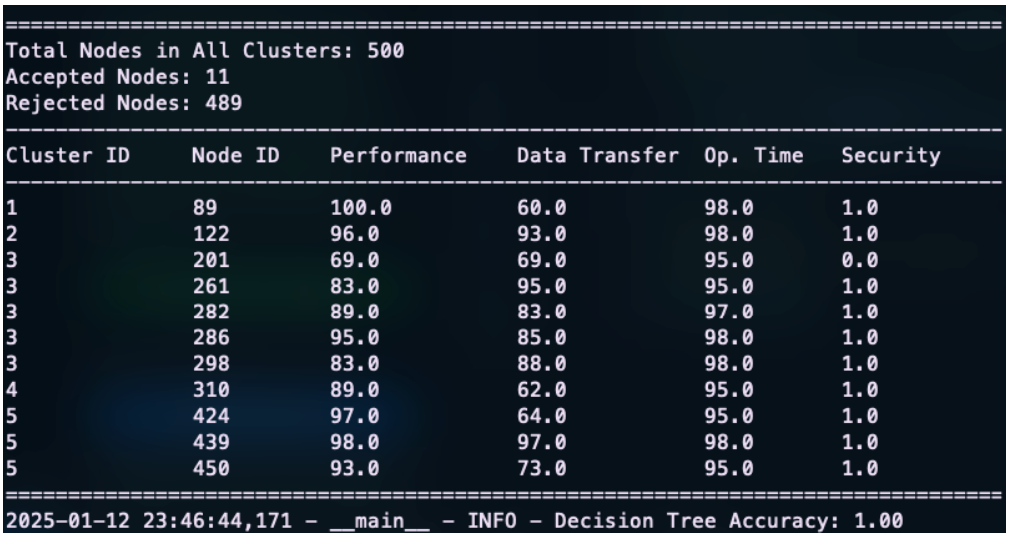
The decision tree training process utilizes scikit-learn's implementation with optimized hyperparameters for our specific use case. The model is trained on a combination of node performance metrics and AIS predictions, using a basic k-fold cross-validation approach to prevent overfitting. During operation, the tree processes node candidates sequentially, evaluating each against learned decision boundaries. The workflow consists of collecting node parameters (computing speed, data transfer rate, uptime, and security level), combining these with AIS predictions, and producing binary classification outputs[8]. This streamlined process ensures efficient node selection while maintaining the interpretability of decisions through the tree's hierarchical structure.

**Performance analysis and practical experiments**

Our empirical analysis demonstrates that the decision tree layer achieves several critical objectives:

1. Maintains a consistent accuracy rate of 99-100% in node selection decisions
2. Achieves sub-millisecond decision times for individual node evaluations
3. Exhibits linear scaling characteristics with increasing node counts
4. Demonstrates robust performance across varying workload patterns

To validate the effectiveness of our hybrid approach, we conducted extensive experiments across various cluster configurations[9]. The experimental setup included three different scales of distributed environments: small-scale (5 clusters), medium-scale (15 clusters), and large-scale (30 clusters) deployments. Within each scale, we tested three different node densities per cluster: 100, 300, and 500 nodes, resulting in total system sizes ranging from 500 to 15000 nodes.

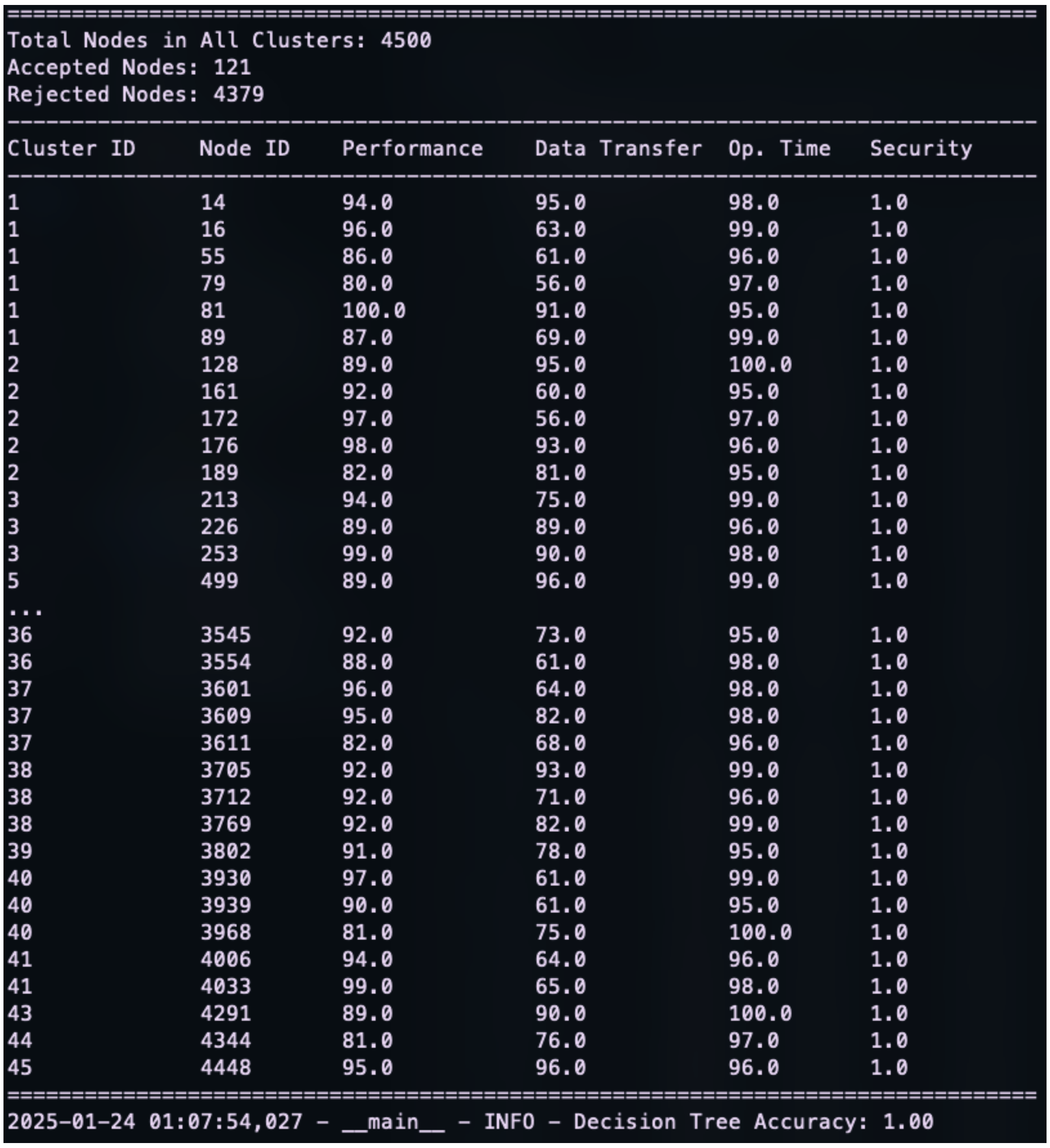


```
================================================================================
Total Nodes in All Clusters: 500
Accepted Nodes: 11
Rejected Nodes: 489
--------------------------------------------------------------------------------
Cluster ID     Node ID    Performance    Data Transfer  Op. Time   Security
--------------------------------------------------------------------------------
1              89         100.0          60.0           98.0       1.0
2              122        96.0           93.0           98.0       1.0
3              201        69.0           69.0           95.0       0.0
3              261        83.0           95.0           95.0       1.0
3              282        89.0           83.0           97.0       1.0
3              286        95.0           85.0           98.0       1.0
3              298        83.0           88.0           98.0       1.0
4              310        89.0           62.0           95.0       1.0
5              424        97.0           64.0           95.0       1.0
5              439        98.0           97.0           98.0       1.0
5              450        93.0           73.0           95.0       1.0
================================================================================
2025-01-12 23:46:44,171 - __main__ - INFO - Decision Tree Accuracy: 1.00
```

Fig.3. Results from the initial experiment with 500-node deployment showing the selected nodes that met task requirements. The table displays 11 successful node selections (2.2% acceptance rate) from the total pool of 500 nodes, achieving 100% accuracy in node selection. For each selected node, the table presents: cluster ID, node ID, and the four key parameters (performance, data transfer rate, operational time, and security level) that influenced the selection decision

The experimental results demonstrated remarkable consistency in the system’s accuracy. The combined AIS-DT approach maintained a high accuracy rate between 99-100% across all configurations, regardless of the system scale. In contrast, when the AIS was used alone without the Decision Tree layer, the accuracy fluctuated between 94-96%, indicating that the DT layer provides crucial refinement of the initial AI Scheduler predictions.

Beyond accuracy metrics, our system provides practical utility by generating a comprehensive list of eligible nodes for task execution. Each selected node is accompanied by its complete performance profile, including computing speed, data transfer rate, operational uptime, and security level. This detailed output enables system administrators to make informed decisions about task allocation and provides transparency in the selection process. The consistency in accuracy across different scales, combined with the detailed node selection output, demonstrates that our hybrid approach successfully addresses the challenges of distributed task scheduling while providing actionable insights for system operators.



```
================================================================================
Total Nodes in All Clusters: 4500
Accepted Nodes: 121
Rejected Nodes: 4379
--------------------------------------------------------------------------------
Cluster ID     Node ID    Performance    Data Transfer  Op. Time   Security
--------------------------------------------------------------------------------
1              14         94.0           95.0           98.0       1.0
1              16         96.0           63.0           99.0       1.0
1              55         86.0           61.0           96.0       1.0
1              79         80.0           56.0           97.0       1.0
1              81         100.0          91.0           95.0       1.0
1              89         87.0           69.0           99.0       1.0
2              128        89.0           95.0           100.0      1.0
2              161        92.0           60.0           95.0       1.0
2              172        97.0           56.0           97.0       1.0
2              176        98.0           93.0           96.0       1.0
2              189        82.0           81.0           95.0       1.0
3              213        94.0           75.0           99.0       1.0
3              226        89.0           89.0           96.0       1.0
3              253        99.0           90.0           98.0       1.0
5              499        89.0           96.0           99.0       1.0
...
36             3545       92.0           73.0           95.0       1.0
36             3554       88.0           61.0           98.0       1.0
37             3601       96.0           64.0           98.0       1.0
37             3609       95.0           82.0           98.0       1.0
37             3611       82.0           68.0           96.0       1.0
38             3705       92.0           93.0           99.0       1.0
38             3712       92.0           71.0           96.0       1.0
38             3769       92.0           82.0           99.0       1.0
39             3802       91.0           78.0           95.0       1.0
40             3930       97.0           61.0           99.0       1.0
40             3939       90.0           61.0           95.0       1.0
40             3968       81.0           75.0           100.0      1.0
41             4006       94.0           64.0           96.0       1.0
41             4033       99.0           65.0           98.0       1.0
43             4291       89.0           90.0           100.0      1.0
44             4344       81.0           76.0           97.0       1.0
45             4448       95.0           96.0           96.0       1.0
================================================================================
2025-01-24 01:07:54,027 - __main__ - INFO - Decision Tree Accuracy: 1.00
```

Fig.4. Results from the medium-scale deployment experiment with 15 clusters containing 300 nodes each (4500 total nodes). The table presents detailed performance metrics for nodes that met the selection criteria, including cluster ID, node ID, and the four core parameters (performance, data transfer rate, operational time, and security level). This visualization demonstrates the system's ability to maintain efficient node selection across a larger deployment scale[10] while preserving detailed parameter tracking for each selected node

## Future Enhancements

### Reinforcement Learning Integration

The integration of reinforcement learning (RL) techniques presents significant opportunities for enhancing the current architecture[10], particularly in optimizing long-term scheduling decisions[11]. Through continuous interaction with the distributed environment, an RL agent could develop sophisticated policies for workload prediction and resource allocation. By analyzing historical task execution patterns, the system would anticipate workload fluctuations and proactively adjust resource distribution[11]. Furthermore, the RL component could optimize the decision-making process by dynamically weighing various operational parameters based on their demonstrated impact on task execution success.

### Performance Optimization Opportunities

The current architecture presents several potential ways for performance enhancement. System response time in distributed task scheduling represents a critical performance metric that could be improved through architectural optimizations[11]. Implementing parallel processing for decision tree evaluation, coupled with strategic caching of common decision paths, could potentially reduce computational overhead[12]. Additionally, leveraging GPU acceleration for neural network computations might further optimize the decision-making pipeline.

Resource utilization efficiency could be enhanced through the implementation of predictive workload analysis and dynamic cluster size adjustment[12]. By incorporating advanced load balancing mechanisms that leverage predictive analytics, the system could achieve more optimal resource distribution across the network[12]. These improvements would be particularly beneficial during periods of high variability in workload patterns.

System reliability could be substantially improved through the integration of predictive maintenance capabilities and automated failover mechanisms. A comprehensive health monitoring system would enable preemptive identification of potential failures, while automated recovery procedures would minimize service interruptions[13]. These enhancements, working together, could potentially yield a significant reduction in system response time while maintaining the current high accuracy rates in node selection.

## Conclusion

This research presents a novel approach to distributed task scheduling through the implementation of a hybrid artificial intelligence system. The proposed architecture successfully combines distributed AI Schedulers (AIS) with a centralized Decision Tree (DT) layer, demonstrating significant advantages over traditional scheduling methods. Our experimental results, conducted across various scales of deployment (5-30 clusters) and node densities (100-500 nodes per cluster), validate the effectiveness of this approach.

The system's key contributions include:

1. A hierarchical scheduling architecture that effectively balances local intelligence with global optimization
2. An efficient neural network-based node pre-selection mechanism that achieves 94-96% accuracy independently
3. A sophisticated decision tree layer that refines selections to achieve 99-100% accuracy consistently
4. A scalable implementation that maintains performance across different system scales (500-15000 nodes)
5. A transparent decision-making process that provides detailed node selection rationales

Furthermore, our experimental analysis demonstrates that the combination of AIS and DT components creates a synergistic effect, where the decision tree's refinement of AI Scheduler predictions leads to near-perfect accuracy in node selection. The system's ability to maintain this high performance across different scales while providing detailed selection criteria and performance profiles represents a significant advancement in distributed task scheduling.

Future research directions could explore the integration of reinforcement learning techniques for dynamic optimization and the application of this architecture to other domains requiring sophisticated resource allocation. The robust performance and scalability demonstrated in our experiments suggest that this approach could be effectively adapted to address similar challenges in other distributed computing contexts.

## Resources

1. Buyya, R., Srirama, S. N., Casale, G., Calheiros, R., Simmhan, Y., Varghese, B., Gelenbe, E., Javadi, B., Vaquero, L. M., Netto, M. A. S., Toosi, A. N., Rodriguez, M. A., Llorente, I. M., Vimercati, S. D. C. D., Samarati, P., Milojicic, D., Varela, C., Bahsoon, R., Assuncao, M. D. D., ... Shen, H. (2018). A Manifesto for Future Generation Cloud Computing: Research Directions for the Next Decade. ACM Computing Surveys, 51(5), Article 105. https://doi.org/10.1145/3241737
2. Thafzy, V. M. (2025). A Review on the Integration of Machine Learning in Cloud Computing Resource Management. International Journal of Scientific Research and Technology, 2(1), 18-21. https://doi.org/10.5281/zenodo.14592545
3. Liang, F., et al. (2024). "Resource Allocation and Workload Scheduling for Large-Scale Distributed Deep Learning: A Survey." arXiv:2406.08115, Computer Science, Distributed, Parallel, and Cluster Computing.
4. Ren, F., & Liu, H. (2024). Dynamic scheduling for flexible job shop based on MachineRank algorithm and reinforcement learning. Sci. Rep. 14, 29741.
5. Ma, X. (2018). Using Classification and Regression Trees: A Practical Primer. Charlotte, NC: Information Age Publishing.
6. Yang, R., Ouyang, X., Chen, Y., Townend, P., & Xu, J. (2018). "Intelligent Resource Scheduling at Scale: A Machine Learning Perspective." IEEE Symposium on Service-Oriented System Engineering, 0-7695-6394-5/18, 132-140
7. M. van Steen and A.S. Tanenbaum, Distributed Systems, 4th ed., distributed-systems.net, 2023.
8. Vinícius G. Costa and Carlos E. Pedreira. 2022. Recent advances in decision trees: an updated survey. Artif. Intell. Rev. 56, 5 (May 2023), 4765–4800. https://doi.org/10.1007/s10462-022-10275-5
9. Gregg, B. (2021). Systems Performance: Enterprise and the Cloud (2nd ed.). Boston: Addison-Wesley.
10. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. Cambridge, MA: MIT Press.
11. Zhang, Y., & Xu, H. (2024). "Distributed Data-Driven Learning-Based Optimal Dynamic Resource Allocation for Multi-RIS-Assisted Multi-User Ad-Hoc Network." Algorithms, 17(1), 45.
12. Wang, S., Zheng, H., Wen, X., & Shang, F. (2024). Distributed High-Performance Computing Methods for Accelerating Deep Learning Training. J. Knowl. Learn. Sci. Technol., 3(3), 108-126. https://doi.org/10.60087/jklst.v3.n3.p108-126
13. Ousterhout, J. K. (2018). A Philosophy of Software Design. Palo Alto, CA: Yaknyam Press.