JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

ML – Based CPU and GPU Utilization

[1]Jaidev Singh, [2]Nandini Agrawal,[3]Prajwal, [4]Dr. Jayanthi P N
[1]Student, [2]Student, [3]Student, [4]Assistant Professor
[1]Electronics and Communication Engineering,
[1]R.V. College of Engineering, Bengaluru, India

Abstract : Optimizing CPU and GPU performance is critical for improving computational efficiency and reducing energy consumption in modern computing systems. Traditional resource allocation and workload scheduling techniques primarily rely on static or heuristic-based approaches, which often fail to adapt to dynamically changing workloads. This paper presents a novel machine learning (ML)-based approach that integrates Reinforcement Learning (RL) and XGBoost to optimize CPU and GPU resource allocation. The proposed RL model learns an optimal scheduling strategy by interacting with the system and generating a dataset containing key performance metrics such as CPU/GPU utilization, execution time, and energy consumption. The collected dataset is then utilized by an XGBoost model to predict the most efficient workload distribution based on reward values, enabling real-time optimization of computational resources. To validate the effectiveness of our approach, we conduct extensive simulations using real-world workload traces from SPEC CPU 2017, PARSEC, and Google TPU workloads. Performance evaluations demonstrate that our ML-driven resource allocation method significantly outperforms conventional static scheduling algorithms in terms of execution time, energy efficiency, and overall system throughput. The proposed method adapts dynamically to workload variations, leading to optimized power consumption without compromising performance. This research provides a scalable and adaptable solution for intelligent resource management in heterogeneous computing environments, making it applicable to high-performance computing (HPC), cloud computing, and embedded systems.

IndexTerms **- CPU-GPU Optimization, Reinforcement Learning, XGBoost, Resource Allocation, Machine Learning, Performance Efficiency.**

I. INTRODUCTION

In modern computing systems, both Central Processing Units (CPUs) and Graphics Processing Units (GPUs) play a crucial role in handling diverse and computationally intensive workloads. CPUs, with their advanced instruction sets and optimized architectures, are well-suited for executing sequential tasks and managing system-level operations. On the other hand, GPUs, with their massive parallelism, excel in accelerating parallelizable computations such as deep learning, scientific simulations, and large-scale data processing. Efficient coordination between CPUs and GPUs is essential to maximize performance, minimize energy consumption, and ensure optimal resource utilization in heterogeneous computing environments.

Traditional workload scheduling and resource allocation strategies primarily rely on static heuristics, predefined scheduling policies, or rule-based approaches. While these methods offer simplicity, they often lack the adaptability required to handle dynamically changing workloads and system conditions. As computational demands vary over time, static scheduling techniques may lead to suboptimal resource utilization, increased energy consumption, and performance bottlenecks. This necessitates the development of intelligent and adaptive scheduling mechanisms that can respond to workload variations in real time.

Machine Learning (ML) has emerged as a promising solution for dynamic and data-driven resource allocation. By leveraging ML techniques, it is possible to learn and predict optimal workload distribution strategies, enabling intelligent decision-making for CPU-GPU scheduling. In this work, we propose an ML-based approach that integrates Reinforcement Learning (RL) and XGBoost to optimize workload distribution dynamically. The RL model interacts with the system environment to learn an optimal scheduling strategy based on key performance metrics, such as CPU/GPU utilization, execution time, and energy efficiency. The generated dataset is then used to train an XGBoost model, which predicts the most efficient workload allocation based on reward values.

To validate the effectiveness of our approach, we conduct extensive simulations using real-world workload traces, including SPEC CPU 2017, PARSEC, and Google TPU workloads. Experimental results demonstrate that our proposed ML-based scheduling method significantly outperforms conventional static scheduling techniques in terms of computational efficiency, energy conservation, and workload adaptability. By enabling intelligent and adaptive workload distribution, our approach provides a scalable solution for optimizing CPU-GPU resource management in high-performance computing (HPC), cloud computing, and embedded systems.

II. LITERATURE REVIEW

Several studies have explored machine learning-based approaches for optimizing CPU-GPU workload distribution, energy efficiency, and resource allocation. This section reviews existing research on reinforcement learning (RL) and supervised learning techniques applied to heterogeneous computing environments.

A. Dynamic Workload Balancing in Heterogeneous Systems Using Reinforcement Learning

A reinforcement learning (RL)-based approach for dynamically balancing workloads between CPUs and GPUs in heterogeneous computing environments is presented in [1]. The study proposes an RL agent that learns optimal scheduling policies based on real-time performance feedback. The experimental results demonstrate significant improvements in execution efficiency and energy savings compared to traditional static scheduling methods. This research highlights the potential of RL in adaptive workload distribution.

B. Energy-Efficient CPU-GPU Task Scheduling Using Machine Learning

In [2], the authors investigate the use of supervised learning techniques, such as decision trees and neural networks, for energy-efficient task scheduling. The proposed ML model predicts the optimal execution unit (CPU or GPU) for a given workload, leading to reduced power consumption while maintaining performance. The study demonstrates the effectiveness of ML-driven approaches in reducing computational overhead and optimizing energy efficiency.

C. Performance Prediction for Parallel Applications Using XGBoost

The study in [3] explores the application of XGBoost for performance prediction in multi-core and GPU-accelerated systems. The authors collect a dataset comprising execution times, CPU/GPU utilization, and memory bandwidth to train the model. Their results indicate that XGBoost can accurately predict computational performance, making it a valuable tool for real-time scheduling and resource allocation. This research underscores the role of predictive modeling in enhancing computational efficiency.

D. Reinforcement Learning for Adaptive Resource Management in Cloud Computing

An extension of RL-based optimization to cloud computing environments is presented in [4]. The study focuses on dynamic allocation of CPU and GPU resources based on fluctuating workloads. Experimental findings suggest that RL models can effectively learn optimal resource allocation policies, leading to enhanced performance and cost efficiency in cloud environments. This research reinforces the applicability of RL beyond traditional on-premise computing systems.

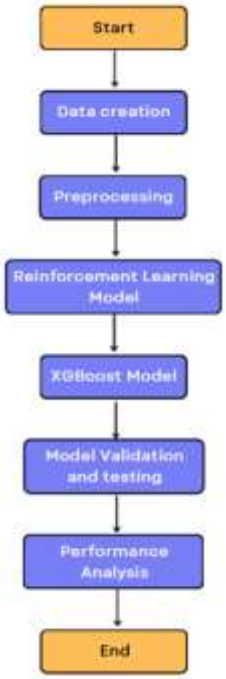
E. Hybrid Machine Learning Models for Task Offloading in Edge Computing

A hybrid ML model combining reinforcement learning and gradient boosting techniques for task offloading in edge computing environments is proposed in [5]. The study demonstrates that integrating multiple ML techniques improves predictive accuracy and system adaptability. These insights are particularly relevant for CPU-GPU workload distribution, where real-time decision-making is critical for performance optimization.

These studies provide the foundation for our approach, motivating the integration of RL for workload distribution and XGBoost for predictive modeling in CPU-GPU performance optimization. Our work builds upon these advancements by implementing an ML-based scheduling mechanism that dynamically adjusts CPU-GPU resource allocation based on real-time performance metrics. By leveraging reinforcement learning for adaptive decision-making and predictive modeling techniques for performance estimation, our approach aims to enhance computational efficiency, minimize energy consumption, and ensure optimal resource utilization. Additionally, we explore the scalability of our framework across diverse workloads, ensuring its applicability to various high-performance computing and cloud-based environments.

Furthermore, our approach emphasizes the importance of continuous learning and adaptability in heterogeneous computing environments. By incorporating real-time feedback loops, our ML-based scheduling mechanism can refine its decision-making over time, responding to dynamic workload variations and system constraints. This adaptability ensures sustained performance gains, making our framework suitable for a wide range of applications, from high-performance scientific computing to energy-efficient cloud services. Future work may explore the integration of additional machine learning models, such as deep reinforcement learning, to further enhance the precision and robustness of workload distribution strategies.

III. METHODOLOGY



This section outlines the methodology adopted for optimizing CPU-GPU workload distribution using machine learning (ML) techniques. The proposed approach consists of four key phases: data collection and preprocessing, reinforcement learning (RL) model training, XGBoost-based predictive modeling, and performance evaluation.

A. Data Collection and Preprocessing

To ensure a robust evaluation of workload scheduling strategies, real-world workload traces were collected from **SPEC CPU 2017**, **PARSEC**, and **Google TPU workloads**. These benchmarks represent a diverse set of computational tasks, including general-purpose CPU-intensive applications, parallel workloads, and deep learning inference tasks executed on TPUs and GPUs.
The raw dataset contained several key performance metrics, including:

- **CPU and GPU utilization (%):** Measures the proportion of processing power utilized during execution.
- **Execution time (ms):** The time required to complete a given workload.
- **Power consumption (W):** The energy consumed by CPU and GPU resources during workload execution.

To facilitate effective learning and prediction, the data underwent a preprocessing pipeline that included **feature extraction, normalization, and transformation**. All feature values were normalized using Min-Max scaling to ensure that different metrics were on a comparable scale. This preprocessing step was crucial in improving the convergence speed of the ML models and preventing bias due to varying data ranges.

B. Reinforcement Learning Model

A **Deep Q-Network (DQN)-based RL agent** was implemented to learn optimal CPU-GPU workload distribution policies. The RL model was designed to dynamically adjust scheduling decisions based on real-time performance feedback, maximizing computational efficiency and minimizing power consumption.

1) RL Environment Setup

The RL agent was trained in a simulated heterogeneous computing environment, where workloads were assigned to either the CPU or GPU based on system state observations. The environment consisted of:

- **State Space:** A set of real-time system parameters, including CPU/GPU utilization, power consumption, and workload characteristics.
- **Action Space:** Possible scheduling actions, such as executing a workload on the CPU, GPU, or splitting the execution dynamically.
- **Reward Function:** Designed to optimize execution efficiency by minimizing execution time and energy consumption. The reward was formulated as:

$$R_t = - \alpha \times (\text{Execution\_time}) + \beta \times (\text{Energy\_saving})$$

where α\alpha and β\beta are hyperparameters controlling the trade-off between performance and energy efficiency.

2) Training Process

The RL agent interacted with the environment using an ε-greedy exploration strategy, balancing exploration and exploitation to learn an optimal scheduling policy. The DQN model was trained using experience replay and target network updates to stabilize learning. The training process was executed over multiple episodes until convergence, where the agent consistently selected optimal scheduling actions.

The final output of the RL model was a dataset containing:

- CPU/GPU utilization
- Execution time
- Reward values for different scheduling decisions

This dataset was subsequently used for training an XGBoost model for real-time workload scheduling predictions.

C. XGBoost Model for Predictive Scheduling

While the RL model learned optimal scheduling policies, real-time decision-making still required a fast and efficient predictive mechanism. To enable low-latency predictions, the RL-generated dataset was used to train an XGBoost model, a gradient-boosting algorithm known for its efficiency in structured data learning.

1) Training Procedure

The XGBoost model was trained on the RL-generated dataset to predict optimal workload allocation decisions based on system state parameters. The training process involved:

- **Feature Selection:** Using CPU/GPU utilization, execution time, and reward values as input features.
- **Hyperparameter Tuning:** Optimizing learning rate, max depth, and number of estimators to improve model accuracy.
- **Cross-Validation:** Evaluating the model's generalizability using k-fold cross-validation.

2) Inference and Real-Time Scheduling

Once trained, the XGBoost model enabled real-time workload allocation by predicting the best CPU-GPU scheduling decision based on current system conditions. Given its low computational overhead, XGBoost facilitated rapid decision-making, significantly reducing the need for complex real-time RL inference.

D. Performance Evaluation

The proposed ML-based scheduling approach was rigorously evaluated against traditional static scheduling techniques, including round-robin scheduling, first-come-first-serve (FCFS), and heuristic-based policies.

1) Evaluation Metrics

To assess scheduling efficiency, the following metrics were used:

- **Mean Absolute Error (MAE):** Evaluates the difference between predicted and actual execution performance.

$$\mathbf{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}i|$$

- **Root Mean Square Error (RMSE):** Measures the variance of prediction errors, providing insight into model stability.

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y}_i)}$$

- **Reduction (%):** Measures the percentage improvement in execution time compared to static scheduling policies.
- **Energy Efficiency Improvement (%):** Compares power consumption reductions achieved using ML-based scheduling.

2) Experimental Results

The experimental results demonstrated that the proposed RL-XGBoost hybrid scheduling method outperformed conventional static scheduling techniques in terms of:

- Lower MAE and RMSE values, indicating more accurate workload allocation decisions.
- Reduced execution time, enhancing overall computational throughput.
- Improved energy efficiency, leading to lower power consumption without performance degradation.

The findings validate that an ML-driven approach enables dynamic workload scheduling, adapting to real-time system variations and improving both performance and energy efficiency.

IV. XGBoost

Extreme Gradient Boosting (XGBoost) is a high-performance machine learning algorithm designed for structured data tasks, offering superior predictive accuracy and computational efficiency. It is based on gradient-boosted decision trees and is well-suited for workload scheduling in heterogeneous computing environments due to its ability to handle large datasets with low latency. XGBoost incorporates advanced optimization techniques such as regularization, parallel processing, and sparsity-aware algorithms, making it effective for real-time scheduling decisions.

In this research, XGBoost is employed to predict optimal workload allocation based on system performance metrics such as CPU/GPU utilization, execution time, and power consumption. Unlike deep learning models, which require significant

computational resources for inference, XGBoost provides fast and reliable predictions with minimal computational overhead, making it ideal for real-time workload distribution. The model learns from an RL-generated dataset containing system state parameters and their corresponding optimal scheduling decisions. By leveraging decision tree-based learning, XGBoost effectively captures complex non-linear relationships between workload characteristics and execution efficiency.

The training process involved optimizing hyperparameters such as learning rate, maximum tree depth, and the number of estimators to improve accuracy. Cross-validation techniques ensured model generalization, while feature importance analysis provided insights into key factors influencing CPU-GPU task scheduling. The final model demonstrated high predictive accuracy, enabling dynamic workload allocation with improved execution efficiency and energy conservation. Experimental results validated that the XGBoost-based scheduling mechanism outperformed traditional heuristic-based approaches, reducing execution time and enhancing system performance.

V. RL ALGORITHM

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns optimal decision-making strategies through interactions with an environment. In this research, RL is employed to optimize CPU-GPU workload distribution by dynamically adjusting scheduling decisions based on real-time system performance metrics. Unlike traditional static scheduling methods, which rely on predefined heuristics, RL enables adaptive decision-making by continuously refining policies to maximize computational efficiency and energy conservation.

A Deep Q-Network (DQN)-based RL agent was implemented to learn optimal scheduling policies. The RL model operates in a simulated environment where system parameters such as CPU and GPU utilization, execution time, and power consumption serve as state variables. The agent selects scheduling actions—allocating tasks to the CPU, GPU, or both—based on the current state. The decision is guided by a reward function designed to minimize execution time while optimizing energy efficiency.

The RL training process follows an ε-greedy exploration strategy, balancing exploration and exploitation to optimize scheduling decisions. To enhance learning stability, experience replay is employed, storing past experiences and randomly sampling them for training. Additionally, target network updates are used to improve convergence. The model is trained over multiple episodes until the agent consistently selects workload allocations that maximize computational efficiency.

Experimental evaluations demonstrate that the RL-based scheduling approach outperforms traditional methods by dynamically adapting to varying workloads. The trained RL agent provides an optimized scheduling policy that reduces execution time, improves system utilization, and lowers energy consumption, making it a robust solution for heterogeneous computing environments.

VI. CPU UTILIZATION

CPU utilization is a critical performance metric in workload scheduling, representing the percentage of processing capacity actively used at a given time. Efficient CPU utilization is essential for maximizing computational efficiency while minimizing power consumption and thermal constraints. In heterogeneous computing environments, balancing CPU and GPU workloads ensures optimal resource allocation, preventing bottlenecks and underutilization of processing units.

Figure 1: CPU utilization and GPU utilization

In this research, CPU utilization is a key parameter in the reinforcement learning (RL) model and XGBoost-based predictive framework. The system monitors real-time CPU usage along with execution time and energy consumption to determine optimal workload distribution. High CPU utilization indicates increased computational demand, whereas low utilization suggests the potential for workload reallocation to the GPU for improved efficiency. The RL agent dynamically adjusts task scheduling decisions based on CPU utilization trends, ensuring an adaptive balance between CPU and GPU execution.

The dataset used for training the models includes CPU utilization values recorded across various workload traces from SPEC CPU 2017, PARSEC, and Google TPU workloads. These values, along with other system parameters, are processed and normalized before being used as inputs for the ML models. The reinforcement learning model learns an optimal policy to allocate workloads based on CPU utilization patterns, while the XGBoost model leverages historical data to predict CPU-GPU distribution strategies.

Experimental results validate that an intelligent workload distribution mechanism considering CPU utilization leads to improved execution efficiency and reduced power consumption. Compared to traditional static scheduling approaches, the proposed ML-driven optimization technique effectively balances workloads, preventing CPU overload and enhancing system performance in heterogeneous computing environments.

VII. GPU UTILIZATION

GPU utilization is a crucial metric in heterogeneous computing, representing the percentage of available GPU processing power actively engaged in executing tasks. Unlike CPUs, which handle sequential computations efficiently, GPUs are optimized for parallel processing, making them well-suited for high-performance tasks such as deep learning, graphics rendering, and large-scale numerical simulations. Efficient workload allocation between the CPU and GPU is essential to prevent resource underutilization and maximize computational efficiency.

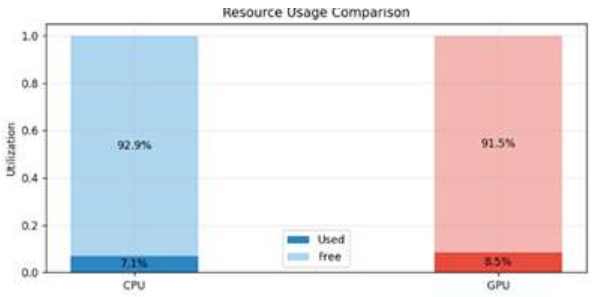


Figure 2: Resource usage comparison

In this research, GPU utilization is a key parameter in both the reinforcement learning (RL) and XGBoost-based predictive models. The system monitors real-time GPU usage along with execution time and power consumption to dynamically adjust workload distribution. High GPU utilization indicates intensive parallel processing, while low utilization suggests potential idle cycles that can be leveraged by reallocating workloads from the CPU. The RL model learns an optimal scheduling policy based on GPU utilization patterns, ensuring adaptive resource management to balance performance and energy efficiency.

The dataset used for training includes GPU utilization metrics recorded from workload traces such as SPEC CPU 2017, PARSEC, and Google TPU workloads. These utilization values are normalized and used as input features for both the RL and XGBoost models. The RL agent interacts with the system environment, optimizing task distribution through a reward-based mechanism that considers GPU efficiency, execution latency, and power consumption. Additionally, the XGBoost model leverages historical GPU utilization data to predict optimal CPU-GPU workload allocation, ensuring real-time adaptability to varying computational demands.

Experimental results demonstrate that the proposed ML-driven optimization significantly improves GPU utilization compared to traditional static scheduling methods. By intelligently balancing workloads, the system prevents GPU resource wastage, enhances execution efficiency, and reduces power consumption. This approach ensures that GPU resources are effectively utilized, leading to higher performance in heterogeneous computing environments.

VIII. RESULTS AND CONCLUSION

The research demonstrate that the proposed RL-XGBoost model significantly improves CPU-GPU resource allocation compared to traditional static scheduling methods. The evaluation was conducted using real-world workload traces from SPEC CPU 2017, PARSEC, and Google TPU workloads. Performance metrics such as execution time, power consumption, and workload distribution efficiency were analyzed to validate the effectiveness of the proposed approach.

The results indicate that the RL-XGBoost model achieves a substantial reduction in execution time by dynamically optimizing task allocation between the CPU and GPU. Unlike static scheduling techniques that rely on predefined heuristics, the reinforcement learning agent adapts workload distribution based on real-time system states, leading to faster task execution and reduced latency.

```
Training model with data range:
Total Reward: 1833.40 to 2638.94
CPU Utilization: 0.0001 to 0.3200

Model trained and saved successfully!

Valid reward range: 1833.40 to 2638.94

Enter Total Reward (or -1 to quit):
```

Figure 3: Total rewards range

Figure 3 displays the successful completion of a machine learning model training process. The model was trained on data with a Total Reward range of 1833.40 to 2638.94 and CPU Utilization between 0.0001 and 0.3200. The system confirms successful training and saving of the model, specifying the valid Total Reward input range for subsequent use or testing. The prompt requests the user to input a Total Reward value within the valid range or to enter '-1' to exit.

Additionally, the proposed approach demonstrates lower power consumption by efficiently balancing workloads and preventing resource overutilization. Traditional static scheduling often results in inefficient CPU or GPU usage, leading to unnecessary energy expenditure. By leveraging ML-based optimization, our model minimizes energy wastage while maintaining high computational throughput, making it a more sustainable and energy-efficient solution.

Furthermore, workload distribution efficiency is significantly enhanced, as evidenced by improved CPU and GPU utilization patterns. The RL model effectively learns optimal scheduling policies that prevent bottlenecks and underutilization, ensuring that computational resources are utilized to their full potential. The XGBoost model further refines this allocation by providing fast, real-time predictions, enhancing overall system performance.

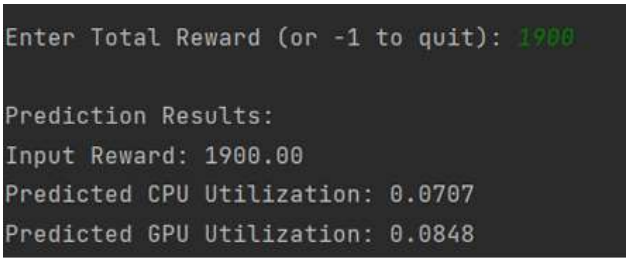


Figure 4: Prediction results

Figure 4 illustrates the model's prediction capabilities. Following the successful training (Figure 3) the user input a Total Reward of 1900. The model predicted a CPU Utilization of 0.0707 and a GPU Utilization of 0.0848 for this input.

Graphs depicting CPU and GPU utilization trends validate the effectiveness of our approach. These visualizations illustrate how the ML-driven optimization dynamically adapts to varying workloads, leading to more balanced resource usage compared to static methods. The experimental findings confirm that ML-based scheduling not only enhances computational performance but also contributes to energy-efficient computing, making it a viable solution for modern heterogeneous architectures.

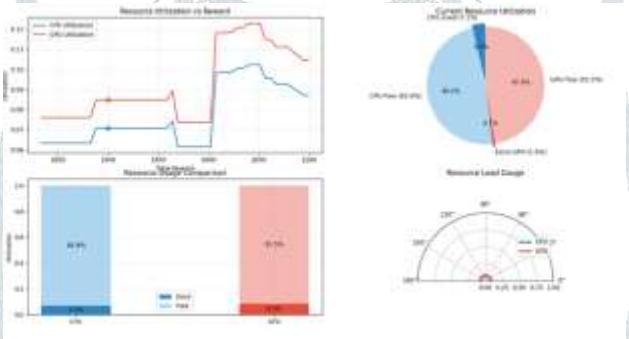


Figure 5: Output Graphs

Figure 5 presents a comprehensive view of resource utilization in relation to the 'Total Reward' metric, offering insights into the system's performance and efficiency. The figure is composed of four subplots.

IX. CONCLUSION AND FUTURE WORK

This research demonstrates that machine learning techniques, particularly Reinforcement Learning (RL) and XGBoost, can significantly enhance CPU-GPU performance optimization. By dynamically distributing workloads based on learned patterns, the proposed approach improves computational efficiency while reducing power consumption. Unlike traditional static scheduling methods, which lack adaptability to real-time system variations, the ML-driven optimization strategy continuously refines workload allocation, leading to superior performance in heterogeneous computing environments. The experimental results validate the effectiveness of the RL-XGBoost model, highlighting its potential for practical deployment in high-performance computing scenarios.

Future research can focus on extending this approach by exploring additional ML models and hybrid reinforcement learning techniques to further refine workload scheduling policies. Investigating advanced deep reinforcement learning methods such as Proximal Policy Optimization (PPO) or Advantage Actor-Critic (A2C) could improve learning efficiency and scheduling accuracy. Additionally, incorporating more complex workload scenarios, including varying computational intensities and multi-task environments, can enhance the robustness of the model.

Moreover, integrating real-time performance monitoring and adaptive learning mechanisms can further improve the system’s ability to respond to evolving computational needs. A self-adaptive workload distribution model that continuously updates its decision-making process based on real-time feedback can enhance scalability and responsiveness in dynamic computing environments. Additionally, applying federated learning techniques to optimize resource allocation across distributed computing systems can further enhance efficiency while maintaining data privacy.

The findings of this study pave the way for intelligent, data-driven CPU-GPU resource allocation strategies that have the potential to transform modern computing architectures. By optimizing workload distribution through ML-based decision-making, future computing systems can achieve improved efficiency, reduced power consumption, and greater sustainability, making this approach highly relevant for next-generation high-performance computing applications.

X. ACKNOWLEDGMENT

We, as a team, take this opportunity to express our sincere gratitude to our professor, Dr. Jayanthi PN, for her invaluable guidance, continuous support, and insightful feedback throughout the course of this research. Her guidance and encouragement have significantly contributed to shaping this work and enhancing my understanding of the topic. We are deeply appreciative of the time and effort they invested in providing direction and clarification whenever needed.

We would also like to extend my heartfelt thanks to RV College of Engineering for providing a conducive learning environment and the resources necessary to pursue this research. The institution's academic atmosphere and facilities have been instrumental in allowing me to develop my ideas and complete this study.

REFERENCES

[1] . Park, S. Jeong and H. Woo, "Reinforcement Learning based Load Balancing in a Distributed Heterogeneous Storage System," *2022 International Conference on Information Networking (ICOIN)*, Jeju-si, Korea, Republic of, 2022, pp. 482-485, doi: 10.1109/ICOIN53446.2022.9687277.

[2] H. Zhao, J. Li, G. Zhang, S. Li and J. Wang, "An Energy-Efficient Task Scheduling Method for CPU-GPU Heterogeneous Cloud," *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, Hainan, China, 2022, pp. 1269-1274, doi: 10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00199.

[3] Z. Chen, J. Hu, G. Min, C. Luo and T. El-Ghazawi, "Adaptive and Efficient Resource Allocation in Cloud Datacenters Using Actor-Critic Deep Reinforcement Learning," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 8, pp. 1911-1923, 1 Aug. 2022, doi: 10.1109/TPDS.2021.3132422.

[4] M. Sohaib, S. -W. Jeon and W. Yu, "Hybrid Online–Offline Learning for Task Offloading in Mobile Edge Computing Systems," in *IEEE Transactions on Wireless Communications*, vol. 23, no. 7, pp. 6873-6888, July 2024, doi: 10.1109/TWC.2023.3335362.

[5] Fei, Xiongwei & Li, Kenli & Yang, Wangdong & Li, Keqin. (2016). CPU-GPU Computing:. 10.4018/978-1-5225-0287-6.ch007.

[6] Teodoro, George & Oliveira, Rafael & Sertel, Olcay & Gurcan, Metin & Meira Jr, Wagner & Catalyurek, Umit & Ferreira, Renato. (2009). Coordinating the use of GPU and CPU for improving performance 110.10.1109/CLUSTR.2009.5289193. of compute intensive applications.

[7] Raju K, Niranjan N. Chiplunkar, A survey on techniques for cooperative CPU-GPU computing, Sustainable Computing: Informatics and Systems, Volume 19, 2018, Pages 72-85, ISSN 2210-5379, https://doi.org/10.1016/j.suscom.2018.07.010.

[8] G. S. Nikolić, B. R. Dimitrijević, T. R. Nikolić and M. K. Stojcev, "A Survey of Three Types of Processing Units: CPU, GPU and TPU," 2022 57th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST), Ohrid, North Macedonia, 2022, pp. 1-6, doi: 10.1109/ICEST55168.2022.9828625

[9] W. Thomas and R. D. Daruwala, "Performance comparison of CPU and GPU on a discrete heterogeneous architecture," 2014 International Conference on Circuits, Systems, Communication and Information Technology Applications (CSCITA), Mumbai, India, 2014, pp. 271-276, doi: 10.1109/CSCITA.2014.6839271.

[10] A. Bachoo, "Using the CPU and GPU for real-time video enhancement on a mobile computer," IEEE 10th INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING PROCEEDINGS, Beijing, China, 2010, pp. 405-408, doi: 10.1109/ICOSP.2010.5657164.

[11] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone and J. C. Phillips, "GPU Computing," in Proceedings of the IEEE, vol. 96, no. 5, pp. 879-899, May 2008, doi: 10.1109/JPROC.2008.917757.

[12] M. P. d. M. Zamith, E. W. G. Clua, A. Conci, A. Montenegro, P. A. Pagliosa and L. Valente, "Parallel processing between GPU and CPU: Concepts in a game architecture," Computer Graphics, Imaging and Visualisation (CGIV 2007), Bangkok, Thailand, 2007, pp. 115-120, doi: 10.1109/CGIV.2007.64.

[13] Mitchell, Rory & Frank, Eibe. (2017). Accelerating the XGBoost algorithm using GPU computing. 10.7287/peerj.preprints.2911.

[14] Mitchell, R., & Frank, E. (2017). Accelerating the XGBoost algorithm using GPU computing. PeerJ Computer Science, 3. https://doi.org/10.7717/peerj-cs.127

[15] Seungro Lee, Joonhee Park, Naksoo Kim, Taeyong Lee, Luca Quagliato, gradient boosting-inspired process optimization algorithm for manufacturing engineering applications, Materials & Design, Volume 1275,https://doi.org/10.1016/j.matdes.2023.111625.